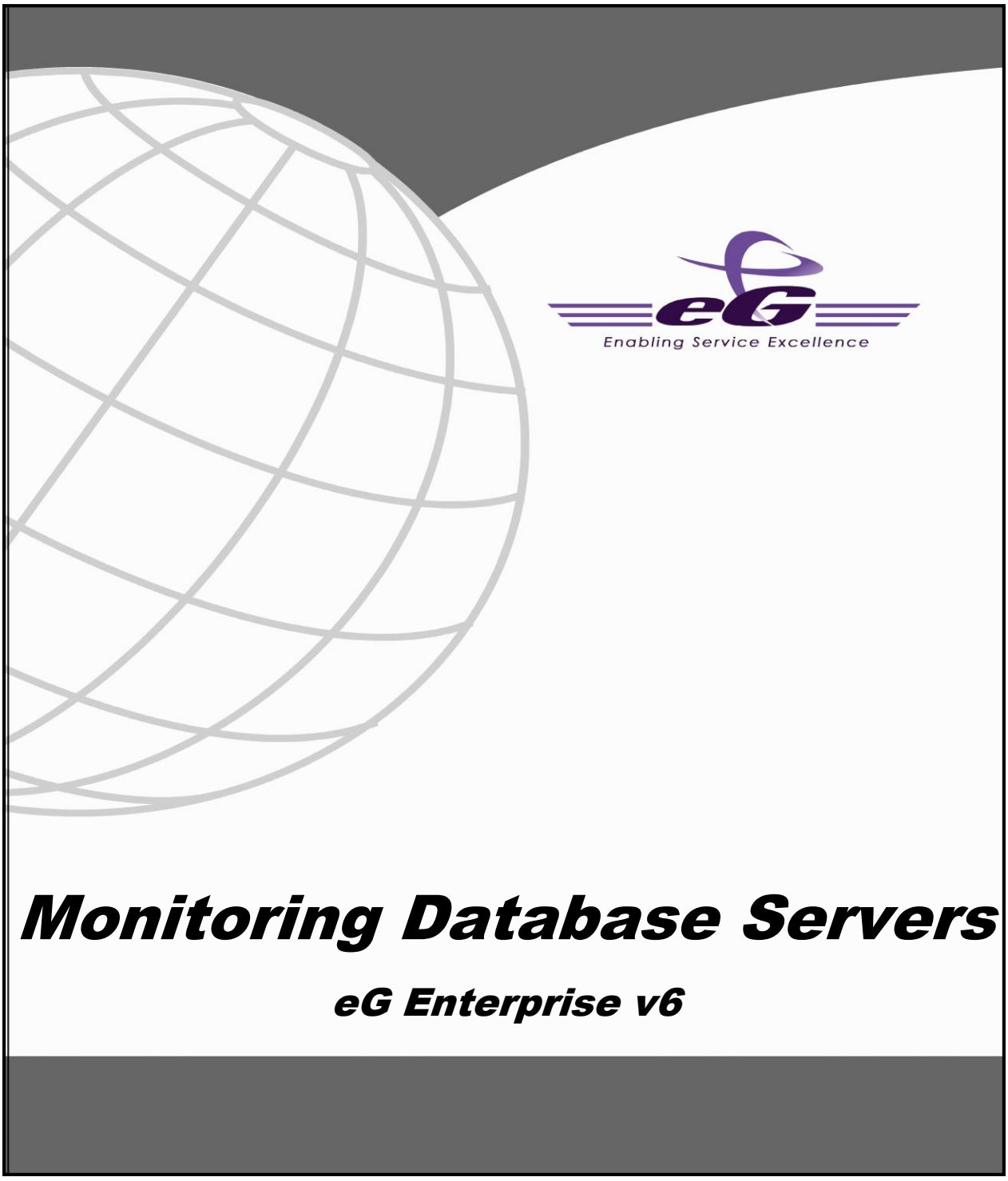


Monitoring Database Servers

eG Enterprise v6

Table of Contents

Table of Figures

1

Introduction

For storage and retrieval of persistent data in an IT infrastructure, application components rely on database servers. A database server is responsible for reliably managing a large amount of data in a multi-user environment so that many users can concurrently access the same data. At the same time, a database server must also prevent unauthorized access and provide efficient solutions for failure recovery.

For ensuring high availability, performance, and security, a database server includes a wealth of data storage, caching, and retrieval functions. To ensure peak performance, a database server needs to be continuously monitored and tuned. In an operational database, specific tables may grow in size, there by choking one or more of the database's tablespaces. Sometimes, there may be a sudden change in workload to the database, resulting in an increase in the number of simultaneously processed transactions. This scenario could result in a performance bottleneck at the database server. Continuous monitoring and optimization of the database server is essential for ensuring that the database server operates at its peak.

The eG Enterprise suite is programmed with a variety of tests that are designed to monitor the critical parameters of various database servers. This document describes how the eG Enterprise suite performs monitoring for database servers.

2

Monitoring Oracle Databases

An Oracle database server consists of many different components. These include internal memory structures, processes that execute the database server's tasks, the physical structures that include resources for storing application data and special resources that are designed to allow for recovering data from problems ranging from incorrect entry to disk failure. All three structures of the Oracle database server running together to allow users to read and modify data are referred to as an Oracle instance. Figure 2.1 demonstrates the various memory, process, and physical storage components of a typical Oracle instance.

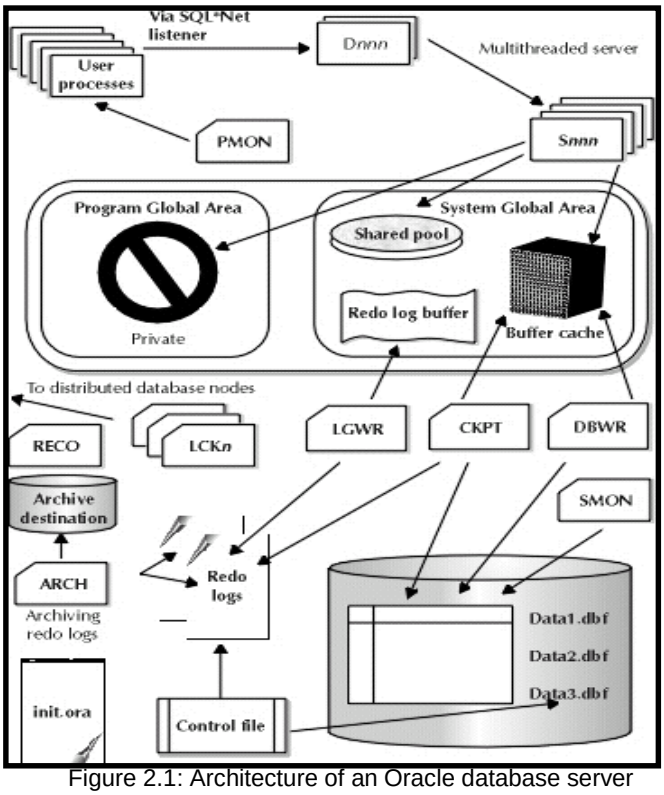


Figure 2.1: Architecture of an Oracle database server

The eG Enterprise Oracle Monitor includes extensive monitoring capabilities for Oracle databases. A single eG agent is capable of monitoring all of the Oracle database instances being executed on a system. Monitoring of the Oracle database instances is performed non-intrusively, with administrators having the option of configuring whether the monitoring is to be performed in an agent-based or agentless manner. eG Enterprise's 100% web-based architecture, allows geographically distributed database servers to be managed from a central manager. Administrators can view and analyze the performance of their database servers in real-time over the web. To avoid overwhelming the administrator with a ton of performance data, the eG Oracle Monitor includes a specialized model for an Oracle database

server. By viewing the layer model of a database server, an administrator can quickly determine which layer(s) of the database server is causing a problem.

Figure 2.2 depicts the layer model that the eG Enterprise suite uses to monitor an Oracle database server. The **Operating System** and **Tcp** layers have been already discussed in the earlier chapters. The **Oracle Processes** layer tracks the status of the individual Oracle processes that support a specific database instance. Above the **Application Processes** layer is the **SQL Network** layer. This layer provides information about the traffic flowing into and out of the database instance. As indicated above, to handle incoming requests, an Oracle database server uses logical memory structures that interact with the underlying physical structures to form the database. The **Memory Structures** layer in Figure 2.2 tracks the health of the Oracle server's memory structures. The memory structures of an Oracle database server include:

- The System Global Area (SGA), which is an area of memory that is designed to execute processes to obtain data for user queries as quickly as possible while also maximizing the number of concurrent users that can access the Oracle instance.
- Locks, which are synchronization mechanisms that prevent destructive interactions between transactions accessing the same resource (e.g., user objects such as tables, or system objects such as shared data structures in memory and data dictionary rows).
- Rollback segments, which represent an area where the before change image of the data is stored for undo purposes. The rollback data can be system rollback data or non-system rollback data.
- Above the **Memory Structures** layer, the eG Enterprise suite monitors the storage resources used by the database server. Tablespaces are logical storage resources used for storing tables, indexes, rollback segments, and the data dictionary. A tablespace can belong to one database instance only, and represents the place where the data is actually stored in the database. The **Tablespaces** layer in Figure 2.2 monitors the status of the different tablespaces used by a database server. A tablespace stores the data on disk in the form of one or more datafiles. The **Datafiles** layer of Figure 2.2 captures the health of the datafiles that form a major proportion of any database. These files contain user's data, Oracle's data dictionary and also include tables, indexes and clusters. Above the **Datafiles** layer, the **Oracle Service** layer tracks the overall health of the service offered by the Oracle database instance.

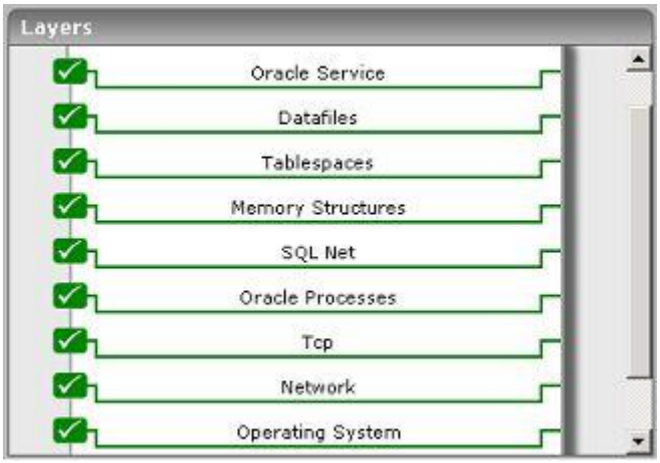


Figure 2.2: Layer model for Oracle database servers

Each of the layers in Figure 2.2 above is mapped to a wide variety of tests, which collect a wealth of performance data from the Oracle database. Using this data, the following questions can be answered:

Database service monitoring	▪ Is the database server available for servicing requests and what is the response time for a typical request?

Session monitoring	▪ How many users are accessing the Oracle database currently? Who are the active users?
Query monitoring	▪ What are the current top 10 SQL queries in terms of resource utilization?
Transaction monitoring	▪ What is the commit and rollback behavior of the applications using the database?
Alert log monitoring	▪ Have there been any recent errors/events in the Oracle alert log? What are they?
Rollback segment monitoring	▪ Is there heavy contention for the rollback segments?
Lock and latch monitoring	▪ Is there contention for locks? Is a specific application holding a lock for a long time? Which lock(s) are these?
Cache monitoring	▪ Are the library cache, dictionary cache, and the data buffer cache adequately sized?
Full table scan monitoring	▪ Is there any full table scan happening on the database? If so, how frequently?
Tablespace monitoring	▪ Are any of the tablespaces reaching their storage capacity? Is the load adequately balanced across the tablespaces?
Hot file monitoring	▪ Is the disk I/O (read/write) being balanced across the datafiles or is there a particular hot datafile that is handling all the requests?
Redo log monitoring	▪ Is the Oracle redo-log buffer sufficiently sized, or is there a large number of requests waiting for redo log space?
Object monitoring	▪ Is there any invalid object in the database? Which ones? Which objects have been recently modified and when? Are there objects that have reached their maximum extent? Which ones are these?

2.1 The Operating System Layer

Besides the **SystemDetails test**, **DiskActivity test**, **DiskSpace test**, **Uptime test**, and **MemoryDetails test** that have been already discussed, the **Operating System** layer of an Oracle server can be optionally enabled to execute the following tests. These tests are disabled by default. To enable a test, open the **AGENTS – TESTS CONFIGURATION** page using the Agents -> Tests -> Configure menu sequence, select *Oracle Database* from the **Select a component type** list, go to the **DISABLED TESTS** section, click on the check box preceding this test, and finally, click on the **Update** button.

2.1.1 Host Devices Test

The HostDevice test monitors the status of different devices accessible via a server.

Purpose	Monitors the status of different devices accessible via a server
Target of the test	A server that supports the Host Resources MIB

Agent deploying the test	A remote agent
Configurable parameters for 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. **SNMPPORT** - The port used to poll for SNMP statistics (default 161) 4. **SNMPVERSION** – By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the **SNMPVERSION** list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list. 5. **SNMPCOMMUNITY** – The SNMP community name that the test uses to communicate with the target host. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the **SNMPVERSION** chosen is **v3**, then this parameter will not appear. 6. **USERNAME** – This parameter appears only when **v3** is selected as the **SNMPVERSION**. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against the **USERNAME** parameter. 7. **AUTHPASS** – Specify the password that corresponds to the above-mentioned **USERNAME**. This parameter once again appears only if the **SNMPVERSION** selected is **v3**. 8. **CONFIRM PASSWORD** – Confirm the **AUTHPASS** by retyping it here. 9. **AUTHTYPE** – This parameter too appears only if **v3** is selected as the **SNMPVERSION**. From the **AUTHTYPE** list box, choose the authentication algorithm using which SNMP v3 converts the specified **USERNAME** and **PASSWORD** into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm 10. **ENCRYPTFLAG** – This flag appears only when **v3** is selected as the **SNMPVERSION**. By default, the eG agent does not encrypt SNMP requests. Accordingly, the **ENCRYPTFLAG** is set to **NO** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **YES** option. 11. **ENCRYPTTYPE** – If the **ENCRYPTFLAG** is set to **YES**, then you will have to mention the encryption type by selecting an option from the **ENCRYPTTYPE** list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard 12. **ENCRYPTPASSWORD** – Specify the encryption password here. 13. **CONFIRM PASSWORD** – Confirm the encryption password by retyping it here. 14. **TIMEOUT** – Specify the duration (in seconds) beyond which the SNMP query executed by this test should time out. The default is 10 seconds.

Outputs of the test	One set of results for every device being accessed via the server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Current status: This measure indicates the current status of a device that is accessible via the server.	Number	A value of 0 indicates that the device is operating normally. A value of 1 indicates that there is a warning associated with the device, whereas a value of 2 signifies an error.
	Errors: This measure indicates the number of errors associated with a device that occurred during the last measurement period.	Number	An unusually high number of device errors signifies a problem.

2.1.2 Host Storage Test

This test auto-discovers all the storage areas of a server and tracks the usage of each of these areas.

Purpose	Auto-discovers all the storage areas associated with a server
Target of the test	A server that supports the Host Resources MIB
Agent deploying the test	A remote agent

Configurable parameters for 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. **SNMPPORT** - The port used to poll for SNMP statistics (default 161) 4. **SNMPVERSION** – By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the **SNMPVERSION** list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list. 5. **SNMPCOMMUNITY** – The SNMP community name that the test uses to communicate with the target host. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the **SNMPVERSION** chosen is **v3**, then this parameter will not appear. 6. **USERNAME** – This parameter appears only when **v3** is selected as the **SNMPVERSION**. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against the **USERNAME** parameter. 7. **AUTHPASS** – Specify the password that corresponds to the above-mentioned **USERNAME**. This parameter once again appears only if the **SNMPVERSION** selected is **v3**. 8. **CONFIRM PASSWORD** – Confirm the **AUTHPASS** by retyping it here. 9. **AUTHTYPE** – This parameter too appears only if **v3** is selected as the **SNMPVERSION**. From the **AUTHTYPE** list box, choose the authentication algorithm using which SNMP v3 converts the specified **USERNAME** and **PASSWORD** into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm 10. **ENCRYPTFLAG** – This flag appears only when **v3** is selected as the **SNMPVERSION**. By default, the eG agent does not encrypt SNMP requests. Accordingly, the **ENCRYPTFLAG** is set to **NO** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **YES** option. 11. **ENCRYPTTYPE** – If the **ENCRYPTFLAG** is set to **YES**, then you will have to mention the encryption type by selecting an option from the **ENCRYPTTYPE** list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard 12. **ENCRYPTPASSWORD** – Specify the encryption password here. 13. **CONFIRM PASSWORD** – Confirm the encryption password by retyping it here. 14. **TIMEOUT** – Specify the duration (in seconds) beyond which the SNMP query executed by this test should time out. The default is 10 seconds.
Outputs of the test	One set of results for every storage area on the server being monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Storage size: Represents the total size of a storage area associated with a server.	GB	
	Usage of storage area: This metric denotes the percentage capacity of a storage area that is currently allocated.	Percent	A value close to 100% denotes a storage area that is highly used.
	Free space on storage area: This metric denotes the amount of storage of a storage area that is currently available for use.	GB	
	Allocation failures on storage area: The number of requests for storage represented by this entity that could not be honored in the last measurement period because there was not enough storage available to service application requests	Number	Ideally, there should be no allocation failures.

2.1.3 Host System Test

This test monitors the number of users accessing a server and the processes executing on a server.

Purpose	Monitors the number of users accessing a server and the processes executing on a server
Target of the test	A server that supports the Host Resources MIB
Agent deploying the test	A remote agent

Configurable parameters for 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. **SNMPPORT** - The port used to poll for SNMP statistics (default 161) 4. **SNMPVERSION** – By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the **SNMPVERSION** list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list. 5. **SNMPCOMMUNITY** – The SNMP community name that the test uses to communicate with the target host. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the **SNMPVERSION** chosen is **v3**, then this parameter will not appear. 6. **USERNAME** – This parameter appears only when **v3** is selected as the **SNMPVERSION**. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against the **USERNAME** parameter. 7. **AUTHPASS** – Specify the password that corresponds to the above-mentioned **USERNAME**. This parameter once again appears only if the **SNMPVERSION** selected is **v3**. 8. **CONFIRM PASSWORD** – Confirm the **AUTHPASS** by retyping it here. 9. **AUTHTYPE** – This parameter too appears only if **v3** is selected as the **SNMPVERSION**. From the **AUTHTYPE** list box, choose the authentication algorithm using which SNMP v3 converts the specified **USERNAME** and **PASSWORD** into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm 10. **ENCRYPTFLAG** – This flag appears only when **v3** is selected as the **SNMPVERSION**. By default, the eG agent does not encrypt SNMP requests. Accordingly, the **ENCRYPTFLAG** is set to **NO** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **YES** option. 11. **ENCRYPTTYPE** – If the **ENCRYPTFLAG** is set to **YES**, then you will have to mention the encryption type by selecting an option from the **ENCRYPTTYPE** list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard 12. **ENCRYPTPASSWORD** – Specify the encryption password here. 13. **CONFIRM PASSWORD** – Confirm the encryption password by retyping it here. 14. **TIMEOUT** – Specify the duration (in seconds) beyond which the SNMP query executed by this test should time out. The default is 10 seconds.
Outputs of the test	One set of results for each server being monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Current users: The current number of users logged in to the server being monitored.	Number	
	Current processes: The current number of processes executing on the server being monitored.	Number	

2.1.4 Host Processors Test

This test monitors the CPU usage of every processor on a server.

Purpose	Monitors the CPU usage of every processor on a server
Target of the test	A server that supports the HOST-RESOURCES MIB
Agent deploying the test	A remote agent

Configurable parameters for 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. **SNMPPORT** - The port used to poll for SNMP statistics (default 161) 4. **SNMPVERSION** – By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the **SNMPVERSION** list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list. 5. **SNMPCOMMUNITY** – The SNMP community name that the test uses to communicate with the target host. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the **SNMPVERSION** chosen is **v3**, then this parameter will not appear. 6. **USERNAME** – This parameter appears only when **v3** is selected as the **SNMPVERSION**. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against the **USERNAME** parameter. 7. **AUTHPASS** – Specify the password that corresponds to the above-mentioned **USERNAME**. This parameter once again appears only if the **SNMPVERSION** selected is **v3**. 8. **CONFIRM PASSWORD** – Confirm the **AUTHPASS** by retyping it here. 9. **AUTHTYPE** – This parameter too appears only if **v3** is selected as the **SNMPVERSION**. From the **AUTHTYPE** list box, choose the authentication algorithm using which SNMP v3 converts the specified **USERNAME** and **PASSWORD** into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm 10. **ENCRYPTFLAG** – This flag appears only when **v3** is selected as the **SNMPVERSION**. By default, the eG agent does not encrypt SNMP requests. Accordingly, the **ENCRYPTFLAG** is set to **NO** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **YES** option. 11. **ENCRYPTTYPE** – If the **ENCRYPTFLAG** is set to **YES**, then you will have to mention the encryption type by selecting an option from the **ENCRYPTTYPE** list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard 12. **ENCRYPTPASSWORD** – Specify the encryption password here. 13. **CONFIRM PASSWORD** – Confirm the encryption password by retyping it here. 14. **TIMEOUT** – Specify the duration (in seconds) beyond which the SNMP query executed by this test should time out. The default is 10 seconds.
Outputs of the test	One set of results for every processor on the IBM AS/400 server being monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Cpu utilization: The average, over the last minute, of the percentage of time that a processor was not idle.	Percent	A consistently high value of this measure indicates that there could be a CPU bottleneck on the server.

In addition to the above, the **Operating System** layer of an Oracle server can also be configured to run the following tests: VarAdmMessages test, UnixTables test, BufferCache test, ProcessState test, IOWaits test, InodeCache test, Paging test, ad Swap test. These tests have been dealt with extensively in the *Monitoring Unix and Windows Servers* document.

2.2 The Network Layer

The tests associated with the **Network** layer are shown by Figure 2.3.

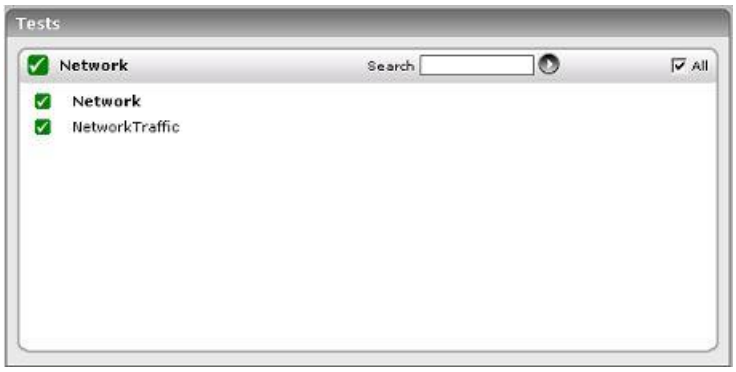


Figure 2.3: The tests associated with the Network layer

The tests associated with the **Network** layer have been elaborately discussed in the *Monitoring Unix and Windows Servers* document.

2.3 The TCP Layer

Apart from the **Tcp** test, a *TcpPort* test can also be optionally enabled for the **Tcp** layer of an Oracle database server. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

2.3.1 TCP Port Test

This operating system-specific test periodically tracks the status of TCP connections to and from a server. This test can be used in different ways. For instance, an administrator can use this test to determine the number of connections that currently exist to a specific TCP port on the server (e.g., the web server port). Alternately, the administrator can also determine the number of TCP connections established from one server to another server - for example, from a web server to a specific application server.

To understand the measures reported by this test, consider the state/transition diagram for the TCP protocol below. The different TCP protocol states are in round-ended boxes, and the transitions are the labeled arrows. The transitions

show how your program can make TCP move from one state to another. It also shows how the remote peer can make your stack change TCP states, and how you can recognize these changes at the application level.

A TCP client calls the **`connect()`** function (or similar), which causes TCP to send an empty packet with the SYN control bit set (SYN_SENT). The remote peer's stack sees this "synchronize" request, and sends back an empty packet with the SYN and ACK bits set (i.e. "I acknowledge your synchronize request"). When the client receives the SYN/ACK packet, it sends back an ACK packet, and reports a successful connection to the client program.

The TIME_WAIT state is a safety mechanism, to catch stray packets for that connection after the connection is "officially" closed.

FIN_WAIT_1 usually happens when the local program calls **`shutdown()`** with the "how" parameter set to 1 or SD_SEND, but the remote peer doesn't respond. Likewise FIN_WAIT_2 happens when the remote peer shuts down its sending half of the connection, and your program doesn't respond. Since FIN_WAIT states often last up to 10 minutes, it's well worth the effort to fix the problem that's causing these FIN_WAIT states.

To use the TcpPortTest, an administrator should specify a list of TCP source/port or destination/port combinations that he/she is interested in monitoring. For each such combination, the TcpPortTest reports the number of TCP connections in each of the TCP protocol states. Analysis of the results can point to scenarios that need attention - e.g., abnormally high established connections to a specific TCP port, unusually large number of connections in the FIN_WAIT state, etc. Often in multi-tier infrastructures that include a number of inter-dependent application tiers, it is also interesting to compare the number of connections established to each tier and correlate the increase/decrease of connections across tiers.

Purpose	To monitor the number of the TCP connections to different source/port and destination/port combinations on a server
Target of the test	An application being monitored
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST - THE HOST FOR WHICH THE TEST IS TO BE CONFIGURED
3. PORTNO - The port number on which the application being monitored is running
4. TARGETAPPPORTS - This parameter defines the source/port and/or destination/port combinations to be monitored. This parameter is specified in the format:
PatternName:LocalIP:LocalPort@RemoteIP:RemotePort
PatternName is a unique name by which the pattern being defined is to be identified. This is the name that would appear alongside the test name in the monitor interface.
When configuring this parameter, decide whether you are monitoring connections to the application or connections from the application to other applications. If the connections to an application running on the local system are to be monitored, the LocalIp and LocalPort become relevant. For example, if the number of connections to a web server on the local system have to be monitored, the LocalIp can be "*" (indicating that all the local IP addresses are to be considered), and the LocalPort can be "80", to monitor the web server running on port 80. On the other hand, if the web server is running on a specific IP address, specify this IP address in the LocalIp field.
The RemoteIP is the IP address of the remote end of the TCP connection. In the example above, TCP connections can be established from any remote address to the web server. Hence, the RemoteIP should be "*" in this example. Likewise, the RemotePort is the TCP port being used to connect to the application being monitored. In the example above, clients can use any TCP port to connect to the application, and hence, the RemotePort setting should be "*".
To conclude, to monitor all the connections to a web server running on port 80 and configured to use an IP address 192.168.10.8, the TARGETAPPPORTS specification should be WebUsage:192.168.10.8:80@*:*.
Suppose the administrator also wants to monitor the TCP connections going out of the web server to a J2EE application server that is listening on IP address 192.168.10.20 on port 6010, then the corresponding TARGETAPPPORTS configuration should be J2EE:*:*@192.168.10.20:6010. This indicates that the clients can be using any IP address/port to connect to the application server.
The complete TARTGETAPPPORTS specification for this example, will hence be:
WebUsage:192.168.10.8:80@*:*, J2EE:*:*@192.168.10.20:6010
As another example, you can also instruct the eG Enterprise system to monitor all TCP connections from IP addresses in the range 192.168.10.30-192.168.10.39, to IP addresses in the range 192.168.10.40-192.168.10.49. The pattern specification for this would be:
Connection34:192.168.10.3*:*@192.168.10.4*:*</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every pattern</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Syn-sent connections:** Indicates the number of connections that are in the process of being established by the host to other server(s)	Number	
	Syn-received connections: Indicates the number of connections that are in the process of being established by remote hosts to this host	Number	
	Established connections: Indicates the total number of TCP connections on this host for the port number(s) specified in the test arguments	Number	The number of TCP connections established to a server is one indicator of the server workload
	Close-wait connections: Indicates the current number of TCP connections to a port that are in the TCP CLOSE_WAIT state. Connections remain in the close wait state when they are waiting for a process to close the TCP socket.	Number	
	Fin-wait-1 connections: Indicates the number of TCP connections to a TCP port that are in the FIN_WAIT_1 state. A TCP connection moves to the FIN_WAIT_1 state when a local program closes a socket but the remote server does not respond.	Number	A large number of FIN_WAIT_1 connections can occur if clients are not properly closing down TCP connections. A connection may linger in this state for tens of minutes.
	Fin-wait-2 connections: Indicates the number of TCP connections to a TCP port that are in the FIN_WAIT_2 state. A connection moves to the FIN_WAIT_2 state when a remote server shuts down its side of a TCP connection and the local server does not respond to it.	Number	

	Time-wait connections: Indicates the number of connections in the TCP TIME_WAIT state. The TIME_WAIT state is a safety mechanism, to catch stray packets for that connection after the connection is "officially" closed. Since the maximum time that such stray packets can exist is 2 times the maximum round-trip time, the TIME_WAIT state lasts twice the round-trip period. Roughly, the duration is 30-120 seconds.	Number	
	TCP send queue: Send-Q is used to show the socket buffer status. This indicates the number of bytes that have been sent to the destination, and are awaiting acknowledgement. **(Available only for Solaris, Linux, HP-UX and AIX)**	Bytes/sec	A high value of this measure indicates a poor network response.
	TCP receive queue: Receive-Q is used to show the socket buffer status. The number indicates the number of bytes received from the source and copied. **(Available only for Solaris, Linux, HP-UX and AIX)**	Bytes/sec	A high value of this measure indicates a poor network response.

2.4 The Oracle Processes Layer

The **Oracle Processes** layer uses an OraProcessTest to track the health of the individual processes corresponding to the Oracle database server. The key processes associated with an Oracle database server are:

1. **The system monitor process (smon):** The usage and function of this Oracle background process is twofold. First, in the event of an instance failure—when the memory structures and processes that comprise the Oracle instance cannot continue to run—the smon process handles recovery from that instance failure. Second, the smon process handles disk space management issues on the database by taking smaller fragments of space and "coalescing" them, or piecing them together.
2. **The process monitor process (pmon):** This process watches the user processes on the database to make sure that they work correctly. If for any reason a user process fails during its connection to Oracle, pmon will clean up the remnants of its activities and make sure that any changes it may have made to the system are "rolled back," or backed out of the database and reverted to their original form.
3. **The log writer process (lgwr):** This background process handles the writing of redo log entries from the redo log

buffer to online redo log files on disk.

4. **The database writer process (dbw):** This background process handles all data block writes to disk. Working in conjunction with the Oracle database buffer cache memory structure, this process prevents users from ever accessing a disk to perform a data change such as update, insert, or delete.
5. **The checkpoint process (ckpt):** This process is used to handle writing log sequence numbers to the datafile headers and control file, alleviating the log writer process of that responsibility.
6. **The recoverer process (reco):** This background process handles the resolution of distributed transactions against the database.

Figure 2.4: The tests associated with the Oracle Processes layer

2.4.1 Oracle Processes Test

For each Oracle instance, this test measures statistics pertaining to the smon, pmon, lgwr, dbw, reco, and ckpt processes.

Purpose	To measure statistics pertaining to Oracle Processes executing on a host
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the specified **HOST** is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **PROCESS -** processName is a string that will be used for display purposes only. processPattern is an expression of the form - **expr** or expr or **expr or expr** or *expr1*expr2*... or expr1*expr2, etc. A leading '*' signifies any number of leading characters, while a trailing '*' signifies any number of trailing characters. The pattern(s) used varies from one application to another and must be configured per application. The default value that appears corresponds to the Unix platform. On Windows environment, this parameter does not require manual configuration. The default value taken is "Oracle*.exe". 8. **INSTANCEWISE** - By default, this test reports the resource usage of all the Oracle server instances that are currently running. For example, if 3 Oracle instances are currently operational, then the test will report the CPU and memory usage of all the three instances by default. Accordingly, the **INSTANCEWISE** parameter is set to **No** by default. On the contrary, if you want this test to report the CPU and memory usage of the monitored Oracle instance only, then set this flag to **Yes**. **Note:** Typically, while monitoring the 'Oracle.exe' process on Windows environments, you might want to set the **INSTANCEWISE** flag to **Yes**. However, on Windows 2000 in particular, before switching on the **INSTANCEWISE** flag, you will have to copy the **tlist.exe** file to the **{WINDOWS_HOME}\system32** directory. This file will be available in the Windows 2000 CD in the **\support\tools\support.cab** file.

<table>
<tr><td></td><td colspan="3">9. USEGLANCE - This flag applies only to Oracle database servers operating on HP-UX systems. HP GlancePlus/UX is Hewlett-Packards's online performance monitoring and diagnostic utility for HP-UX based computers. There are two user interfaces of GlancePlus/UX—Glance is character-based, and gpm is motif-based. Each contains graphical and tabular displays that depict how primary system resources are being utilized. In environments where Glance is run, the eG agent can be configured to integrate with Glance to pull out the process status and resource usage metrics of the configured Oracle processes. By default, this integration is disabled. This is why the USEGLANCE flag is set to No by default. You can enable the integration by setting the flag to Yes. If this is done, then the test polls the Glance interface of HP GlancePlus/UX utility to pull out the desired metrics.
10. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Processes running: Number of instances of a pmon, smon, lgwr, dbw, and reco processes currently executing.</td><td>Number</td><td>This value indicates if too many or too few processes corresponding to an application are executing on the host.</td></tr>
<tr><td>CPU utilization:
Total percentage CPU utilization of each process instance detected above.</td><td>Percent</td><td>A very high value could indicate that processes corresponding to the specified pattern are consuming excessive CPU resources.</td></tr>
<tr><td>Memory utilization:
The ratio of the resident set size of a process to the physical memory of the host system on which it executes, expressed as a percentage.</td><td>Percent</td><td>A sudden increase in memory utilization for a process may be indicative of memory leaks in the application.</td></tr>
</table>

2.4.2 Oracle Client Connections Test

This test reveals how much CPU and memory is utilized by the processes executed by the Oracle clients on the database server. In the event of excessive resource utilization on the database server, administrators can use this test and OracleProcesses test to determine which processes consume more server resources - the critical Oracle database server processes, or the client processes? **This test works only on Unix platforms.**

Purpose	To measure statistics pertaining to Oracle Processes executing on a host
Target of the test	An Oracle server on Unix
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the specified **HOST** is listening 4. **LISTENERNAME** - Specify the Oracle listener name. By default, this value will be the same as the **SID**. 5. **ISPASSIVE** - If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every Oracle database server monitored.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Processes running: Indicates the number of client processes currently executing.	Number	This value indicates if too many or too few processes are executing on the host.
	CPU usage: Indicates the total percentage CPU utilization of the client processes.	Percent	A very high value could indicate that the client processes are consuming excessive CPU resources.
	Memory usage: The ratio of the resident set size of the client processes to the physical memory of the host system on which they execute, expressed as a percentage.	Percent	A sudden increase in memory utilization for the client processes may be indicative of memory leaks in the corresponding application.

2.4.3 Host Processes Test

The HostProcess test monitors the specific processes executing on a server and reports the resource usage of the processes. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors the processes executing on a server and reports the resource usage of specific processes
Target of the test	A server that supports the Host Resources MIB
Agent deploying the test	A remote agent

Configurable parameters for 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. **SNMPPORT** - The port used to poll for SNMP statistics (default 161) 4. **SNMPVERSION** – By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the **SNMPVERSION** list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list. 5. **SNMPCOMMUNITY** – The SNMP community name that the test uses to communicate with the target server. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the **SNMPVERSION** chosen is **v3**, then this parameter will not appear. 6. **USERNAME** – This parameter appears only when **v3** is selected as the **SNMPVERSION**. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against the **USERNAME** parameter. 7. **AUTHPASS** – Specify the password that corresponds to the above-mentioned **USERNAME**. This parameter once again appears only if the **SNMPVERSION** selected is **v3**. 8. **CONFIRM PASSWORD** – Confirm the **AUTHPASS** by retyping it here. 9. **AUTHTYPE** – This parameter too appears only if **v3** is selected as the **SNMPVERSION**. From the **AUTHTYPE** list box, choose the authentication algorithm using which SNMP v3 converts the specified **USERNAME** and **PASSWORD** into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: a. **MD5** – Message Digest Algorithm b. **SHA** – Secure Hash Algorithm 10. **ENCRYPTFLAG** – This flag appears only when **v3** is selected as the **SNMPVERSION**. By default, the eG agent does not encrypt SNMP requests. Accordingly, the **ENCRYPTFLAG** is set to **NO** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **YES** option. 11. **ENCRYPTTYPE** – If the **ENCRYPTFLAG** is set to **YES**, then you will have to mention the encryption type by selecting an option from the **ENCRYPTTYPE** list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard 12. **ENCRYPTPASSWORD** – Specify the encryption password here. 13. **CONFIRM PASSWORD** – Confirm the encryption password by retyping it here.

	14. **PROCESS** - Should contain the specific processes to be monitored. Each process to be monitored is specified in the format "name:pattern". The regular expression pattern denotes patterns that will be used to match processes on the server. For instance, to monitor all the Java processes on a server, specify the argument "java_processes:*java*". 15. **USEPROCESSPATH** - In some operating systems (example, OpenVMS), the process name in the HOST RESOURCES MIB will be an empty string, and the process path will include the process name. In such cases therefore, the test should be explicitly instructed to search the process path strings for the configured process names/patterns. To ensure this, set the **USEPROCESSPATH** parameter to **true**. By default, this parameter is set to **false**. Operating systems where process name (in the HOST RESOURCES MIB) is not an empty string can go with this default setting. 16. **TIMEOUT** – Specify the duration (in seconds) beyond which the SNMP query executed by this test should time out. The default is 10 seconds.		
Outputs of the test	One set of results for every configured process pattern		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Processes running: The number of processes currently executing on the server that match the pattern specified as parameter.	Number	This value indicates if too many or too few processes corresponding to an application are executing on the host.
	Memory utilization: The total memory usage of all processes executing on the server that match the pattern specified as parameter. The memory usage is specified as a percentage of the total memory available on the server.	Percent	A very high value could indicate that processes corresponding to the specified pattern are consuming excessive memory resources.
	Memory size: The total memory usage(in MB) of all processes executing on the server that match the pattern specified as parameter.	MB	A sudden increase in memory utilization for a process(es) may be indicative of memory leaks in the application.
	CPU utilization: The total CPU utilization of all processes executing on the server that match the configured process pattern.	Percent	A high value could signify a CPU bottleneck. The CPU utilization may be high because a few processes are consuming a lot of CPU, or because there are too many processes contending for a limited resource. Check the currently running processes to see the exact cause of the problem.

2.4.4 Oracle Resource Usage Test

This test monitors how effectively the Oracle database server utilizes the session and process resources it is configured with. If the maximum limit to which the resource allocation can grow is violated, it is bound to deteriorate the performance of the server, as the server might not have the bandwidth to handle the additional sessions/processes.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	To identify the resources that occupied in Oracle server.
Target of the test	An Oracle server (9i and 10g)
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the session and process resources allocated to the Oracle server being monitored</td></tr>
<tr><td>Measurements made by the</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td></td><td>Initial allocation: Indicates the number of sessions/processes allocated at the time of creating the Oracle database instance.</td><td>Number</td><td></td></tr>
</table>

	Current utilization : This measure indicates the number of sessions/processes currently active on the Oracle database server.	Number	If the value of the *Cur_utilization* measure exceeds the value of the *Initial_allocation* measure, the additional required resources are allocated from the shared pool, where they must compete for space with other resources. During SGA reservation/initialization, a place is reserved in SGA for the **INITIAL_ALLOCATION** of resources. Based on usage, this allocation can later be changed using the **SESSIONS** and **PROCESSES** parameters in the init database parameter file. The *Config_Limit* measure of this test reports this new configuration only. For most resources, the **INITIAL_ALLOCATION** value and the *Config_Limit* will be the same. However, if the resource allocation is to be changed later, it is good practice to check the maximum utilization limit that Oracle prescribes for the database, and then make the change. This limit signifiies the maximum number of sessions and processes the database can handle, given its current memory capacity. The *Max_utilization_limit* measure reports this Oracle-recommended value. If the value of the *Cur_utilization* measure, exceeds the *Max_utilization_limit*, then the performance of your database is bound to deteriorate. Therefore, ensure that the *Config_Limit* is always well within the *Max_utilization_limit*.
	Maximum utilization limit: This measure indicates the maximum number of sessions/resources that can be allowed to run on the Oracle database server.	Number	If you want to allocate more sessions/processes to your database than what is recommended by Oracle, then its best that you enhance the memory capacity of your database, before altering the resource configurations.

	Configured limit: This measure indicates the number of sessions/processes the Oracle server is currently configured to handle.	Number	
	Percentage utilized: Indicates the percentage of the configured number of sessions/processes (i.e., the *Config_Limit*) that are currently utilized by the Oracle database instance.	Percent	Ideally, the value of this measure should be low. If this measure shows high value, then DBA should increase the configuration value of the **SESSIONS** & **PROCESSES** parameter in the database parameter file. Otherwise, DBA should identify idle sessions and terminate them, so as to make more space available for new sessions/processes.

2.4.5 Oracle Server Response Test

This test is used to measure the request processing ability of the database server.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Measures the request processing ability of the database server
Target of the test	An Oracle 10g server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. SID – The variable name of the Oracle instance.
8. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every Oracle server</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Avg. response time for queries**: This measure indicates time taken by the Oracle server to process SQL queries and send the result-set back to the user.	Secs	Ideally, the value of this measure should be low. If this measure shows high value then you must check which event takes more time. The response time for SQL queries can also increase due to any of the following factors: • A resource contention; • An object lock; • Inefficient SQL queries
	Avg. response time per transaction: Indicates the average time taken by the Oracle database server to process every transaction.	Secs	Ideally, the value of this measure should be low. If this measure, records a high value then it indicates that the Oracle database server takes more time to respond to user transactions. We suggest that you identify the queries that are responsible for this, and fine-tune them for efficiency.

2.4.6 Oracle Instance Status Test

This test monitors each instance of an Oracle server, and reports whether that instance is available or not. If a user complaints of an Oracle server being inaccessible, you can use this test to determine whether the server itself is down or only the target instance is down.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

- This test will report metrics only for an Oracle database server that is executing on a Windows platform.
- To make sure that this test reports metrics for an Oracle database server configured with multiple instances, you need to insert the following entry in the **listener.ora** file of each instance.

```
SID_LIST_LISTENER=
 (SID_LIST=
  (SID_DESC=
    (GLOBAL_DBNAME=orcl.us.acme.com)
    (ORACLE_HOME=/oracle10g)
    (SID_NAME=orcl))
```

Purpose	Monitors each instance of an Oracle server, and reports whether that instance is available or not
Target of the test	An Oracle server
Agent deploying the test	An internal agent
Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **SID** – The variable name of the Oracle instance. 8. **ORACLE HOME** – By default, this test auto-discovers the full path to the Oracle installation directory. This is why, the **ORACLE HOME** parameter is set to *none* by default. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

<table>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each instance of an Oracle server</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Instance availability:
This measure indicates whether this instance is available or not.</td><td>Percent</td><td>The value 100 denotes that the instance is availabe, and the value 0 indicates that the instance is unavailable.</td></tr>
<tr><td>Uptime :
Indicates how long the instance has been up and running.</td><td>Secs</td><td></td></tr>
<tr><td>Uptime since last measure :
Indicates the duration for which the instance has been up since the last measurement period.</td><td>Secs</td><td>If the value of this measure is lesser than the test frequency, it indicates that the instance was rebooted during the last measurement period.</td></tr>
<tr><td></td><td>Is rebooted?
Indicates whether this instance was rebooted or not.</td><td></td><td>This measure reports the value Yes if the instance was rebooted in the last measurement period, and the value No if it was not rebooted. The numeric values that correspond to these measure values have been listed in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>0</td></tr></table>
Note:
This test reports the Measure Values listed in the table above to indicate whether/not the instance was rebooted. In the graph of this measure however, the same will be represented using the numeric equivalents.</td></tr>
</table>

2.5 The Oracle Server Layer

SQL*Net/Net8 is Oracle's client-server middleware that provides transparent connection from client tools to the database. It works across multiple network protocols and operating systems. Using an OraSqlNet test, this layer tracks the amount of traffic that flows in and out of the database through the network.

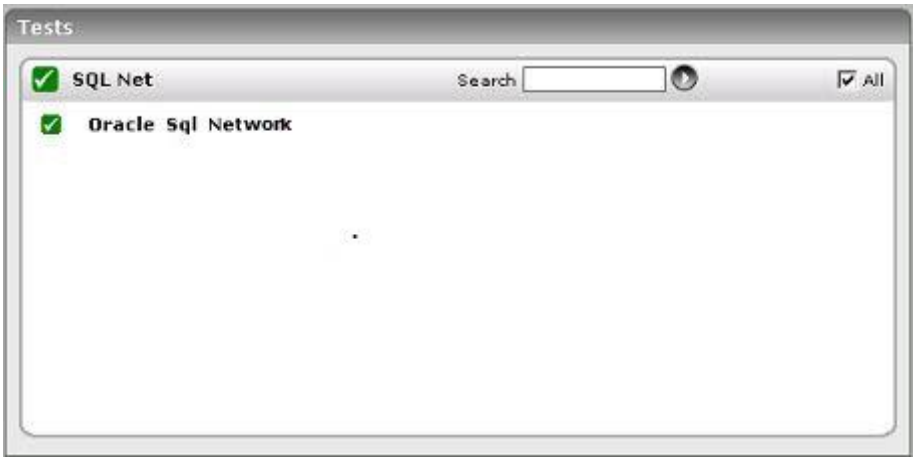


Figure 2.5: The tests associated with the SQL Net layer

2.5.1 Oracle SQL Network Test

Using the JDBC API, this test reports the availability and responsiveness of the server, and collects statistics pertaining to the traffic into and out of the database server.

Purpose	Reports the availability and responsiveness of the server, and collects statistics pertaining to the traffic into and out of the database server
Target of the test	An Oracle server
Agent deploying the test	An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the target Oracle server is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the target Oracle server is listening.

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. Additionally, to ensure that this test runs on Oracle 12c database servers, the special database user should also possess *select* permissions to the *sys.user$* table. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* *Grant select sys.user$ to **<user_name>*** The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **ORACLESID** - The variable name of the oracle instance. This parameter will not be available while configuring this test for an *Oracle Database* server. However, it will be available for this test when monitoring an *Oracle Cluster* service or an *External Oracle* server. 8. **TIMEOUT** - Specify the duration (in seconds) beyond which the test will timeout if no response is received from the server. The default value is 30 seconds. 9. **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:

<table>
<tr><td></td><td colspan="3">

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.

10. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID (instance) monitored.</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Oracle server availability:
Whether the database instance is responding to requests.</td><td>Percent</td><td>The availability is 100% when the instance is responding to a request and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the database instance, or because the instance is using an invalid user account. Besides the above, this measure will report that the server is unavailable even if a connection to the database instance is unavailable, or if a query to the database fails. In this case, you can check the values of the DB connection availability and Query processor availability measures to know what is exactly causing the database instance to not respond to requests - is it owing to a connection unavailability? or is it due to a query failure?
Although included as part of the OraSqlNet test, this measure maps to the Oracle Service layer.</td></tr>
<tr><td>Total response time:
The time taken by the database to respond to a user query. This is the sum total of the connection time and query execution time.</td><td>Secs</td><td>A sudden increase in response time is indicative of a bottleneck at the database server. This could even be owing to a connection delay and/or long running queries to the database. Whenever the value of this measure is high, it would be good practice to compare the values of the Connection time and Query execution time measures to zero-in on the root-cause of the poor responsiveness of the server - is it because of connectivity issues? or is it because of inefficient queries?
Although included as part of the Oracle SQL Network test, this measure maps to the Oracle Service layer.</td></tr>
<tr><td>Data transmit rate:
The rate of data being transmitted by the server in response to client requests.</td><td>KB/Sec</td><td>The data transmission rate reflects the workload on the server.</td></tr>
</table>

	Data receive rate: The rate of data received by the server from clients over SQL*Net.	KB/Sec	This measure also characterizes the workload on the server. As the data rate to a database server increases, consider tuning the Service Layer Data Buffer (SDU) and the Transport Layer Data Buffer (BDU) in the `TNSNames.ora` and `Listener.ora` files to optimize packet transfers across the network.
	DB connection availability: Indicates whether the database connection is available or not.	Percent	If this measure reports the value 100 , it indicates that the database connection is available. The value 0 on the other hand indicates that the database connection is unavailable. A connection to the database may be unavailable if the database is down or if the database is listening on a port other than the one configured for it in the eG manager or owing to a poor network link. If the *Oracle server availability* measure reports the value 0, then, you can check the value of this measure to determine whether/not it is due to the unavailability of a connection to the server.
	Query processor availability: Indicates whether the database query is executed successfully or not.	Percent	If this measure reports the value 100, it indicates that the query executed successfully. The value 0 on the other hand indicates that the query failed. In the event that the *Oracle server availability* measure reports the value *0*, check the value of this measure to figure out whether the failed query is the reason why that measure reported a server unavailability.
	Connection time to database server: Indicates the time taken by the database connection.	Secs	A high value could indicate a connection bottleneck. Whenever the *Total response time* of the measure soars, you may want to check the value of this measure to determine whether a connection latency is causing the poor responsiveness of the server.
	Query execution time: Indicates the time taken for query execution.	Secs	A high value could indicate that one/more queries to the database are taking too long to execute. Inefficient/badly designed queries to the database often take too long to execute. If the value of this measure is higher than that of the *Connection time* measure, you can be rest assured that long running queries are causing the respond slowly to requests.
	Records fetched: Indicates the number of records fetched from the database.	Number	The value 0 indicates that no records are fetched from the database.

The detailed diagnosis of the *Total response time* measure, if enabled, reveals the top ten resource consuming queries to the database. Resource consumption is reported in terms of disk reads, buffer gets, number of loads, execution cycles, rows processed, etc. Using this information, you can identify the non-optimal queries that could impact the database performance adversely (see Figure 2.6).

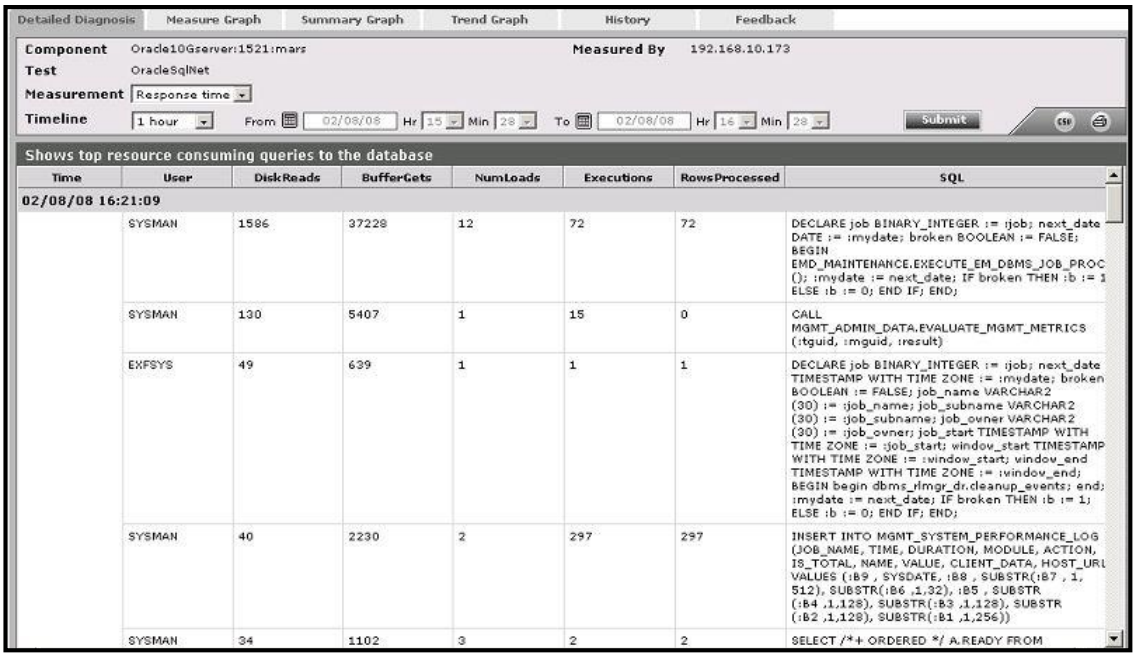


Figure 2.6: Detailed diagnosis of the Response time measure displaying the top 10 resource consuming queries

2.6 The Memory Structures Layer

This layer tracks the health of the SGA, the lock structures, and the rollback segments (see Figure 2.7).

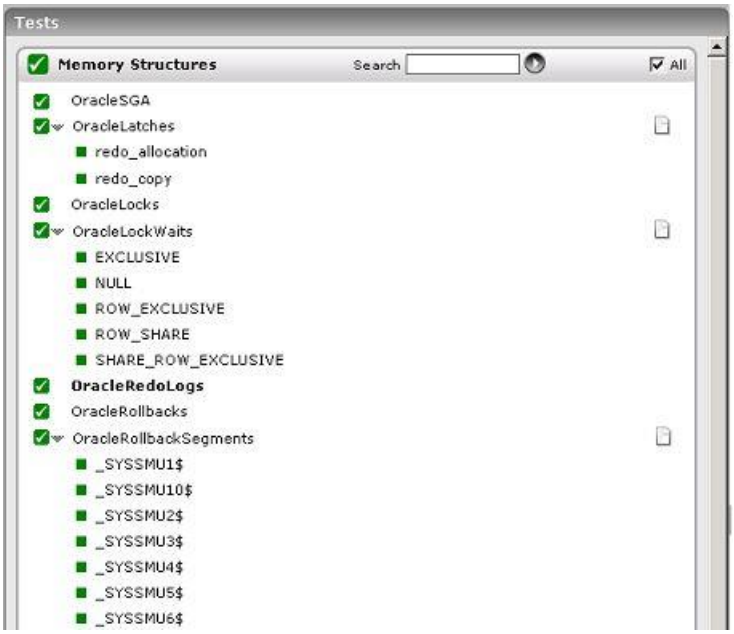


Figure 2.7: Tests mapping to the Memory Structures layer

2.6.1 Oracle Rollback Segments Test

Rollback segments are the undo records, which contain the picture before the database changes. These segments are a critical part of the recovery process and are applied during the rollback phase of the recovery process. The OracleRollbackSegments test measures the load and the efficiency of the rollback segments.

Purpose	Measures the load and the efficiency of the rollback segments
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Writes:
Indicates the total number of writes to a specific rollback segment during the last measurement period.</td><td>Number</td><td>This gives you an indication of the load on a specific rollback segment.</td></tr>
</table>

	Waits: Indicates the number of header waits during the last measurement period.	Number	This indicates the number of times the Oracle Server had to wait to acquire a rollback segment.
	Gets: Indicates the number of header gets during the last measurement period.	Number	This indicates the number of gets that have happened in a specific rollback segment.
	Current hit ratio: Indicates the waits to gets ratio during the last measurement period.	Percentage	This indicates the efficiency at which the rollback segments have performed during the last monitored interval. Ideally, the Hit Ratio should be >= 99%.
	Overall hit ratio: Indicates the waits to gets ratio from the time the instance was started	Percentage	Ideally, the Hit Ratio should be >= 99%. If not, consider adding additional rollback segments. Also, check the system undo header, system undo block, undo header, undo block statistics under "Wait Statistics", for additional information on rollback contention.

2.6.2 Oracle Redo Logs Test

Redo logs are applied during the roll forward phase of the recovery process. These logs hold information about the changes made to the database and whether they were committed. Each change is recorded in a redo record, which has information like the SCN of the change, changed data, commit flag, and information about which data block is changed. The OracleRedoLogs test monitors key performance metrics pertaining to the redo log buffer in an Oracle server instance.

Purpose	Monitors the redo log buffer in an Oracle server instance
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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. 8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every SID monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Redo buffer entries: This indicates the number of attempts to allocate space in the redo buffer. A value other then 0 indicates that the redo writer is falling behind. This could be caused by log switches or checkpoints.	Number	By adjusting the LOG_CHECKPOINT_INTERVAL and LOG_CHECKPOINT_TIMEOUT parameters in the init.ora, you will be able to minimize the number of checkpoints. **From Oracle 9i onwards however, the LOG_CHECKPOINT_INTERVAL parameter is supported only for ensuring backward compatability with previous versions of Oracle. The recommended equivalent in case of Oracle 9i therefore is FAST_START_MTTR_TARGET**. You can also increase the number of LGWR writers. These parameters are new in Oracle 8 and are defined in the init.ora parameters LGWR_IO_SLAVES and ARCH_IO_SLAVES. However, **note that both these parameters are obsolete from Oracle 8i onwards**.
	Redo log space requests: The active log file is full and Oracle is waiting for disk space to be allocated for the redo log entries. Space is created by performing a log switch.	Number	Small Log files in relation to the size of the SGA or the commit rate of the work load can cause problems. When the log switch occurs, Oracle must ensure that all committed dirty buffers are written to disk before switching to a new log file. If you have a large SGA full of dirty buffers and small redo log files, a log switch must wait for DBWR to write dirty buffers to disk before continuing.
	Redo entries: This statistic increments each time redo entries are copied into the redo log buffer. (ie. The number of attempts to allocate space in the redo)	Number	
	Log space requests: This indicates the percentage of log space requests.	Percentage	If the number is greater than 1%, you should increase the size of the Redo Log buffer. I would also check the checkpoint and size of the online redo log file.
	Log space waits: This measure indicates the number of times wait has happened to acquire a log buffer.	Number	If the Log Buffer space waits exist, consider increasing the size of the redo log. Also I would check the speed of the disk that the Online Redo Log files are in.

	Redo no wait: Indicates the percentage of redo entries for which there was space immediately available in the redo log.	Percent	A high value is typically desired for this measure. A low value indicates that many redo entries are waiting for space to become available in the redo logs. Frequent, or slow log switches may be contributing to waits for redo log space. If you are switching logs frequently (e.g. more than once every 15 minutes) this may be improved by increasing the size of the online redo logs. If the log switches are not frequent, check the disks the redo logs reside on to see if log switches are taking a long time due to a slow I/O system. If the I/O system is overloaded, either move the redo logs to disks with less activity, place the logs on dedicated disks or faster devices.

2.6.3 Oracle Cursor Usage Test

This test monitors the number of open cursors for a database instance. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors the number of open cursors for a database instance
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Current open cursors:** The number of cursors currently opened by applications using the database	Number	Many open cursors can exist if any application does not properly close the ResultSets before closing a connection. Alternatively, many simultaneous queries to the database can also result in many open cursors. A continuous increase in open cursors is an indicator of a problem in an application's use of the database.
	Percent open cursors: This metric reports the average percentage of open cursors with respect to the total allowed limit.	Percent	If the percentage of open cursors nears 100%, then this could invoke the "maximum open cursors exceeded" error message. If the percentage is consistently near 100%, consider increasing the value of the 'open_cursors' parameter in the init file.

2.6.4 Oracle Latches Test

Latches are mechanisms for protecting and managing SGA data structures and database objects being accessed concurrently. Unlike locks, latches provide exclusive access to protected data structures. Requests for latches are not queued. So, if a request fails, the requesting process may try later. Typically, latches are used to protect resources that are briefly needed.

An Oracle process can request a latch in one of the following two modes:

- **Willing-to-Wait Mode**: If the requested latch is not immediately available, the process will wait. When an attempt to get a latch in a willing-to-wait mode fails, the process will spin and try again. If the number of attempts reaches the value of the SPIN_COUNT parameter, the process sleeps. Sleeping is more expensive than spinning.
- **Immediate Mode (no-wait mode)**: In this case, the process will not wait if the requested latch is not available and it continues its processing.

Latch contention has a significant impact on performance when:

(i) Enough latches are not available

(ii) A latch is held for a relatively long time

Latch mechanisms most likely to suffer from contention involve requests to write data into the redo log buffer. To serve the intended purpose, writes to the redo log buffer must be serialized. There are four different groupings applicable to redo buffer latches: redo allocation latches and redo copy latches, each with immediate and willing-to-wait priorities.

The OracleLatches test is used to monitor latches in an Oracle database.

Purpose	Monitors the latches in an Oracle database
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Willing-to-wait misses:** This measures the latch contention for requests that were willing to wait to acquire a latch. The value of this metric represents the ratio of the number of requests that could not acquire a latch, to those that could acquire a latch.	Percent	Both the above metrics should be 1% or less. For redo allocation latches, if the Willing_to_wait_misses is high, consider decreasing the LOG_SMALL_ENTRY_MAX_SIZE parameter in the INIT.ORA file. By making the max size for a redo allocation latch smaller, more redo log buffer writes qualify for a redo copy latch instead, thus better utilizing multiple CPU's for the redo log buffer writes. Even though memory structure manipulation times are measured in nanoseconds, a larger write still takes longer than a smaller write. If the size for remaining writes done via redo allocation latches is small enough, they can be completed with little or no redo allocation latch contention. On a single CPU node, all log buffer writes are done via redo allocation latches. If log buffer latches are a significant bottleneck, performance can benefit from additional CPU's (thus enabling redo copy latches) even if the CPU utilization is not an operating system level bottleneck. If the values for redo copy latches is > 1%, consider increasing the LOG_SIMULTANEOUS_COPIES parameter in the INIT.ORA file. This initialization parameter is the number of redo copy latches available. It defaults to the number of CPU's (assuming a multiple CPU node). Oracle recommends setting it as large as 2 times the number of CPU's on the particular node, although quite a bit of experimentation may be required to get the value adjusted in a suitable manner for any particular instance's workload. Depending on CPU capability and utilization, it may be beneficial to set this initialization parameter smaller or larger than 2 X #CPU's. **Note that the LOG_SIMULTANEOUS_COPIES parameter obsolete from Oracle 8i onwards. Hence, if you are monitoring Oracle 8i (or higher), use the hidden parameter _LOG_SIMULTAENOUS_COPIES instead.** Recall that the assignment of log buffer writes to either redo allocation latches or redo copy latches is controlled by the maximum log buffer write size allowed for a redo allocation latch, and is specified in the LOG_SMALL_ENTRY_MAX_SIZE initialization parameter. Recall also that redo copy latches apply only to multiple CPU hosts. **Note that the LOG_SMALL_ENTRY_MAX-SIZE parameter is not supported from Oracle 9i onwards.**
	Immediate misses: This metric measures the latch contention for requests that were not willing to wait to acquire a latch. The value of this metric represents the percentage of "not willing to wait" latch requests that failed. In other words: **the number of "not willing to wait" request misses / the total number of "not willing to wait" requests**	Percent	

2.6.5 Oracle SGA Test

The SGA is the most important memory structure in Oracle. The SGA stores several different components of memory usage that are designed to execute processes to obtain data for user queries as quickly as possible while also maximizing the number of concurrent users that can access the Oracle instance. The main components of the SGA are

- **The buffer cache:** This area of memory allows for selective performance gains on obtaining and changing data. The buffer cache stores data blocks that contain row data that has been selected or updated recently. When the user wants to select data from a table, Oracle looks in the buffer cache to see if the data block that contains the row has already been loaded. If it has, then the buffer cache has achieved its selective performance improvement by not having to look for the data block on disk. If not, then Oracle must locate the data block that contains the row, load it into memory, and present the selected output to the user.
- **The shared pool:** The two main components of the shared pool are the shared SQL library cache and the data dictionary cache. The shared SQL library cache is designed to store parse information for SQL statements executing against the database. Parse information includes the set of database operations that the SQL execution mechanism will perform in order to obtain data requested by the user processes. This information is treated as a shared resource in the library cache. If another user process comes along wanting to run the same query that Oracle has already parsed for another user, the database will recognize the opportunity for reuse and let the user process utilize the parse information already available in the shared pool. The other component of the shared pool is the data dictionary cache, also referred to by many DBAs as the "row" cache. This memory structure is designed to store the data from the Oracle data dictionary in order to improve response time on data dictionary queries. Since all user processes and the Oracle database internal processes use the data dictionary, the database as a whole benefits in terms of performance from the presence of cached dictionary data in memory.

An Oracle database server brings in data into the SGA before doing any operation on it. So it is critical to monitor the various structures inside the SGA to ensure optimal database performance. The OracleSGA test collects a variety of statistics relating to the various SGA components.

Purpose	This test indicates the level of activity on the main components of System Global Area.
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Library cache hit ratio:** The library cache is a buffer that contains the shared SQL and PL/SQL areas. The library cache hit ratio indicates the percentage of shared SQL statements being reparsed.	Percent	For a well-tuned database, this ratio is 90% or more. A lower hit ratio may indicate that the memory allocation to the library cache is insufficient. A low value can significantly degrade the database performance. Increasing the value of the SHARED_POOL_SIZE initialization parameter will help in improving the hit ratio.
	Data buffer cache hit ratio: Indicates the percentage of time that the database server is able to satisfy a request with information that is already available in the memory.	Percent	Physical I/O takes a significant amount of time, and also increases the CPU resources required. The database configuration should be tuned to ensure that a required block will most likely be in memory. The extent to which this is achieved is measured using the buffer cache hit ratio. For a well-tuned database, this ratio should be 80% or higher. A lower value indicates insufficient memory allocation to the database buffer cache. Increasing the value of the DB_BLOCK_BUFFERS initialization parameter will help in improving the hit ratio. If you are monitoring Oracle 9i or higher, then, **note that the DB_BLOCK_BUFFERS parameter is not supported in Oracle 9i or above. It is therefore recommended that you use the equivalent DB_CACHE_SIZE parameter instead.**
	Dictionary cache hit ratio: Indicates the percentage of data dictionary information pertaining to the database, file space availability and object privileges being readily available in the memory.	Percent	As with the case of the library cache, the dictionary cache hit ratio should be at least 90%. A lower value may be due to the insufficient memory allocation to the dictionary cache. Increasing the value of the SHARED_POOL_SIZE parameter will help in improving the hit ratio.
	Redo log buffer misses: Indicates the percentage of requests that had to wait before the redolog buffer is allocated to it.	Percent	Before any transaction could occur, the before image of the data will be stored in the redo log buffer. It is crucial to make the redo log buffer available immediately to the transactions without any wait. The above is crucial to improve the overall performance. This measure indicates how many percentage of times it had to wait for a redo log buffer to be allocated. This can be improved by increasing the LOG_BUFFER parameter.

	Sorts on disk: Indicates the percentage of sorts that is happening on the secondary storage disk.	Percent	For best performance, most sorts should occur in memory; sorts written to disk adversely affect performance. If more than 10% of sorts happen on disk, the database performance could degrade. To improve the sorting performance of a database, consider tuning the parameters SORT_AREA_SIZE and SORT_AREA_RETAINED_SIZE. The dynamically modifiable initialization parameter called SORT_AREA_SIZE specifies the maximum amount of memory to use for each sort. If a significant number of sorts require disk I/O to temporary segments, an application's performance may benefit from increasing the size of the sort area. **Oracle 9i (or above) supports the SORT_AREA_SIZE and the SORT_AREA_RETAINED_SIZE parameters only to ensure backward compatibility with previous versions of Oracle. Therefore, while monitoring Oracle 9i or higher, it is recommended that you use the equivalent PGA_AGGREGATE_TARGET parameter instead.**
	Current size: Indicates the amount of space allocated to the SGA that is currently in use.	MB	A consistent and significant increase in the value of this measure is a cause for concern, as it indicates that SGA components are over-utilizing the available memory resources. In such a scenario, you can use the detailed diagnosis of the *Current size* measure to know the memory usage of the individual SGA components. In the process, you can identify the exact SGA component that is over-utilizing the memory resources.
	Buffer nowait: Indicates the percentage of requests a server process makes for a specific buffer where the buffer was available immediately.	Percent	If this ratio falls below 90%, it indicates that the server process has to wait for something before obtaining the buffer. In this case, determine which type of block is being contended for by examining the Buffer Waits Section of Statspack/ AWR report.

	Soft parse: Indicates the percentage of parse requests where the cursor was already in the cursor cache compared to the number of total parses.	Percent	A soft parse is recorded when the Oracle Server checks the shared pool for a SQL statement and finds a version of the statement that it can reuse. If the value of this measure falls below 90%, it indicates that very often server processes are unable to find SQL statements in the shared pool and are forced to perform hard parses for these statements. Soft parses consume less resources than hard parses, so the larger the value for this item, the better.
	Execute to parse: Is a measure of how many times you execute a sql statement versus parse it.	Percent	If this value is too low, it indicates that an application is parsing statements highly, but not executing properly. This could result in excessive CPU usage, increased shared pool latches, and serious performance degradations in the Oracle database server. The execute to parse ratio takes a hit when an application does not use shareable SQL or if the database has sub-optimal parameters that are reducing the effectiveness of **cursor sharing.** A problem like excessive parsing is likely to manifest itself as additional network traffic between the application server and clients. The additional parse activity may also show up as a marked increase in CPU consumption on the database server.
	Parse CPU to parse elapsed: Indicates the percentage of CPU time used when parsing.	Percent	Parse CPU time means amount of CPU time used for parsing. Parse Elapsed time means amount of clock time used for parsing – this is actually the sum of Parse CPU time and Parse Wait time. The *Parse CPU to parse elapsed ratio* is caculated using the formula: *(Parse CPU time /Parse elapsed time)*100* Ideally, Parse elsapsed must be equal to Parse CPU - i.e., only CPU time should be used for parsing. In that case the ratio will be 100%. However, if wait time is more then this ratio will be less. A low value for this ratio is an indicator of latching problems. Investigate the latch sections in AWR and Statspack report for contention on library cache and shared pool latches.
	CPU to non-parse: Indicates the percentage of CPU time spent for activities other than parsing the SQLs.	Percent	The closer the value of this measure is to 100, better will be the performance of the server. This is because, such a value means that your CPU works on executing your queries instead of parsing them.

	Hard parse ratio: Indicates the percentage of hard parses.	Percent	Hard parsing happens when the oracle server parses a query and cannot find an exact match for the query in the library cache. A hard parse is a very expensive operation both in terms of CPU used and in the number of latches that gets performed. This is why, the value of this measure should be very low. One of the common reasons for high hard parse ratio is the inefficient sharing of SQL statements.
	SGA usage: Indicates the percentage of the target SGA size that is in use currently.	Percent	The SGA_TARGET_SIZE is the total size of all SGA components. You can use this measure to know how much of the configured target SGA size is being used. If this value is close to 100%, it is a cause for concern, as it indicates that the SGA is about to run out of memory. This is turn can slow down user accesses and query execution. In such a scenario, you can use the detailed diagnosis of the *Current size* measure to know the memory usage of the individual SGA components. In the process, you can identify the exact SGA component that is over-utilizing the memory resources.

2.6.6 Oracle PGA Test

A PGA is a memory region that contains data and control information for a server process. It is nonshared memory created by Oracle Database when a server process is started. Access to the PGA is exclusive to the server process. There is one PGA for each server process. Background processes also allocate their own PGAs.

If the PGA runs out of memory, then critical server processes may not run. To avoid this, administrators can use the **Oracle PGA** test to keep an eye on the memory consumption by the PGA and be proactively alerted administrator if one/more server processes are draining memory from the PGA rapidly.

Purpose	Keeps an eye on the memory consumption by the PGA and be proactively alerted administrator if one/more server processes are draining memory from the PGA rapidly
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Current size:
Indicates the amount of PGA memory that is currently in use.</td><td>MB</td><td>Ideally, the value of this measure should be low. A steady rise in this value is a sign of excessive consumption of PGA memory by server processes.</td></tr>
</table>

	PGA hit ratio: Indicates the ratio of the total number of bytes processed in the PGA versus the total number of bytes processed plus extra bytes read/written in extra passes.	Percent	A value of 100% means that all work areas executed by the system since instance startup have used an optimal amount of PGA memory. If the value of this measure falls below 95%, it indicates that the work area cannot run optimal. As a result, one or more extra passes will be performed over the input data. In this case therefore, you can take one of the following actions: • When not using Automatic PGA memory, then increase SORT_AREA_SIZE init parameter. • When using Automatic PGA memory, then increase PGA_AGGREGATE_TARGET init parameter.
	PGA usage ratio: Indicates the percentage of PGA memory that is consumed by the server processes.	Percent	Ideally, the value of this measure should be low. If this value rapidly approaches 100%, it indicates that the PGA is about to run out of free memory. You may then want to consider resizing your PGA memory region by increasing the value for the PGA_AGGREGATE_TARGET init parameter.

2.6.7 Oracle Rollbacks Test

The immediate availability of rollback segments for the various activities that occur in a database server is very critical. Contention for rollback segments can adversely impact the performance of a database server and hence, needs to be detected and reported immediately. To detect contention for rollback segments, the OracleRollbacks test monitors the degree of contention for buffers that contain rollback segment blocks.

Purpose	This test monitors the waits that happen to acquire a rollback segment header or the rollback segments.
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**System segment waits:** Denotes the ratio of the number of waits for acquiring a header block or a block of the SYSTEM rollback segment to the total number of requests for data, measured over a period of time.	Percent	If the number of waits for any class of block exceeds 1% of the total number of requests, the size of the SYSTEM rollback segment needs to be increased.
	Non-system segment waits: Denotes the ratio of the number of waits for acquiring a header block or any other block of a non-SYSTEM rollback segment to the total number of requests for data, measured over a period of time.	Percent	If the number of waits for any class of block exceeds 1% of the total number of requests, the sizes of the existing rollback segments may need to be increased. Alternatively, additional rollback segments to may be created to reduce contention.

2.6.8 OracleLocks Test

An Oracle database server provides data concurrency and integrity between transactions using locking mechanisms. The locking activity of a database server must be monitored carefully because an application holding a specific lock for a long time could cause a number of other transactions relying on the same lock to fail. The OracleLocks test monitors the locking activity on a database server instance. The test looks for the following lock types: ROW SHARE, ROW EXCLUSIVE, SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE.

Purpose	This test indicates the level of locking activity on a database in terms of the number of total locks in different modes and total time since current lock mode was granted.
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **INCLUDELOCKS** – In this text box, provide a comma-separated list of lock types that require monitoring. By default, 'all' will be displayed here, indicating that locks of all types will be monitored. 8. **EXCLUDELOCKS** - Here, provide a comma-separated list of lock types that do not require monitoring. By default, MR, RT, TS, and KT will be displayed here. 9. **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.

<table>
<tr><td></td><td colspan="3">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.
10. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Locks:
Gives the total number of locks that are held.</td><td>Number</td><td>A high value may indicate one of the following:
Too many transactions happening
Locked resources not being released properly
Locks are being held unnecessarily.</td></tr>
<tr><td>Avg lock time:
Indicates the time for which these locks are held.</td><td>Secs</td><td>A high value may indicate one of the following:
Locked resources not being released properly
Locks are being held unnecessarily.</td></tr>
</table>

2.6.9 Oracle Lock Waits Test

An Oracle database server provides data concurrency and integrity between transactions using locking mechanisms. The locking activity of a database server must be monitored carefully because an application holding a specific lock for a long time could cause a number of other transactions relying on the same lock to fail. The OracleLockWaits test identifies the sessions that are waiting for acquiring a lock. The measures made by this test are as follows:

Purpose	Identifies the sessions that are waiting for acquiring a lock
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user 6. This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 7. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 8. **INCLUDELOCKS** – In this text box, provide a comma-separated list of lock types that require monitoring. By default, 'all' will be displayed here, indicating that locks of all types will be monitored. 9. **EXCLUDELOCKS** - Here, provide a comma-separated list of lock types that do not require monitoring. By default, MR, RT, TS, and KT will be displayed here. 10. **WAITTIME** - Specify the aggregate wait time in seconds for all the lock waits. The default is 10. 11. **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.

<table>
<tr><td></td><td colspan="3">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.
12. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Lock waits:
Indicates the number of lock waits.</td><td>Number</td><td>A high number of consistent lock waits in conjunction with high average lock time for a requested lock type may mean that lock(s) on specific object(s) are not being released by session(s) or are being held for a long time causing other sessions to wait for their release. The detailed diagnosis of this measure, if enabled, provides the details of the lock waits such as the object, session etc.</td></tr>
<tr><td>Avg lock wait time:
Indicates the duration for which sessions were waiting for this lock.</td><td>Secs</td><td>An high average lock wait time may mean sessions are having to wait for a long time to acquire locks on objects. The detailed diagnosis of this measure, if enabled, can be used to view the details of the sessions waiting for this lock.</td></tr>
</table>

2.6.10 Oracle Blocker Processes Test

One common problem encountered with databases is blocking. Suppose that process A is modifying data that process B wants to use. Process B will be blocked until process A has completed what it is doing. This is only one type of blocking situation; others exist and are common. What matters to a database administrator is identifying when blocking is a problem and how to deal with it effectively. When blocking is bad enough, users will notice slowdowns and complain about it. With a large number of users, it is common for tens or hundreds of processes to be blocked when slowdowns are noticed. Killing these processes may or may not solve the problem because 10 processes may be blocked by process B, while process B itself is blocked by process A. Issuing 10 kill statements for the processes blocked by B probably will not help, as new processes will simply become blocked by B. Killing process B may or may not help, because then the next process that was blocked by B, which is given execution time, may get blocked by process A and become the process that is blocking the other 9 remaining processes. When you have lots of blocking that is not resolving in a reasonable amount of time you need to identify the root blocker, or the process at the top of the tree of blocked processes. Imagine again that you have 10 processes blocked by process B, and process B is blocked by process A. If A is not blocked by anything, but is itself responsible for lots of blocking (B and the 10 processes waiting on B), then A would be the root blocker. (Think of it as a traffic jam. Figure 2.8 will help) Killing A (via kill) is likely to unblock B, and once B completes, the 10 processes waiting on B are also likely to complete successfully.

The Oracle Blocker Processes test reports the number of blocker processes in a database. The detailed diagnosis of

this test, provides the details of each of these blocker processes, thereby enabling you to identify the root blocker.

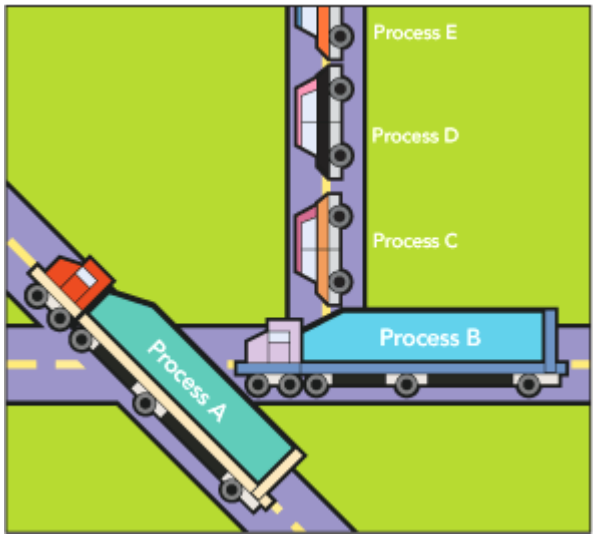


Figure 2.8: The traffic jam analogy representing blocking

Purpose	Monitors the number of blocker processes in a database
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Number of blockers:
Indicates the number of blocker processes.</td><td>Number</td><td>If this value increases suddenly, this is a cause for concern. Likewise, if a process has been blocking other processes for a long time, it is a reason for further investigation. The detailed diagnosis for this test, if enabled, will indicate which process is blocking which other processes. Killing a blocker process that has been running for a long while may get the database running well again. Also, by carefully observing the details of the blocker processes, you can quickly identify the root blocker, and investigate the reason why it is blocking other processes.</td></tr>
</table>

2.7 The Tablespaces Layer

Above the **Memory Structures** layer, eG Enterprise's database server model includes a **Tablespaces** layer that monitors the health of the individual tablespaces of a database server instance. A tablespace is a logical database structure that is designed to store other logical database structures. The objects that may be stored in a tablespace include tables, indexes, rollback segments, etc.

If a tablespace runs out of space then all the statements that try to acquire new space in that tablespace will fail. If there are too much read or write operations to a specific tablespace this could result in serious performance problems. Hence it is critical to monitor individual tablespaces of a database server instance.

To monitor the health of the different tablespaces of a database server instance, the eG Enterprise suite includes an OracleTableSpaces test.

Figure 2.9: Tests mapping to the Tablespaces layer

2.7.1 Oracle Tablespaces Test

This test tracks both the disk space usage per tablespace, as well as the rates at which data is written to and read from a tablespace.

Purpose	This test tracks the health of individual tablespaces associated with a database server instance
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **ALTERNATE VIEW** – In large environments, where the volume of transactions to the Oracle database server is generally very high, this test may take time to execute and retrieve the desired results. To ensure that the test is faster and is resource-efficient, administrators of such environments can create an alternate 'view' on the target Oracle database server, and grant *select* privileges to the view to the special database **USER** mentioned above. Once the view is created, the test should be configured to use the alternate view for metrics collection; to achieve this, specify the name of the view in the **ALTERNATE VIEW** text box. By default, this text box is set to *none*, which implies that the alternate view is not used by default.

This alternate 'view' should be created with the following structure:

```
CREATE OR REPLACE VIEW <VIEW_NAME> (
TABLESPACE_NAME,
FILE_ID,
BLOCK_ID,
BYTES,
BLOCKS,
RELATIVE_FNO
) AS
select /*+ use_hash (tsfi, fet2) */ tsfi.tablespace_name,
       tsfi.file_id,
       fet2.block_id,
       tsfi.blocksize * fet2.blocks,
       fet2.blocks,
       tsfi.relfile#

from   (select /*+ use_hash (ts, fi) */ ts.name tablespace_name,
               fi.file# file_id,
               ts.BLOCKSIZE,
               fi.relfile#,
               ts.ts#
        from   sys.ts$ ts,
               sys.file$ fi
        where  ts.ts# = fi.ts#
        and    ts.online$ in (1,4)) Tsfi,
       (select f.block# block_id,
               f.length blocks,
               f.file# file_id,
               f.ts#
        from   sys.fet$ f
        union all
        select f.ktfbfebno block_id,
               f.ktfbfeblks blocks,
               f.ktfbfefno,
               ktfbfetsn
        from   sys.x$ktfbfe f) Fet2
where  fet2.file_id = tsfi.relfile#
and    fet2.ts# = tsfi.ts# ;
```

8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

<table>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Current usage:
Indicates the current actual usage with respect to the current allocated size (Current_size)</td><td>Percent</td><td>As a rule of thumb, at any time, about 20% of the space allocated to a tablespace should be available. In case of auto-extensible tablespaces, even if this percentage touches 100%, there would be no cause for concern. However, if a tablespace is not auto-extensible, then when the percentage disk space usage reaches 100%, all statements that attempt to acquire new space in the tablespace will fail. Under such circumstances, the underlying datafiles of the tablespace may need to be resized or reorganized. Alternately, additional datafiles could be mapped to the tablespace.</td></tr>
<tr><td>Physical reads:
Indicates the rate of physical reads happening on a tablespace.</td><td>Reads/Sec</td><td>A sudden increase in the rate of data accesses may indicate a change in application characteristics. At any stage, if more than 50% of the total reads for a database instance happen to be on a particular tablespace, this may result in performance degradation.</td></tr>
<tr><td>Physical writes:
Indicates the rate of physical writes happening on a tablespace.</td><td>Writes/Sec</td><td>More than 50% of the total writes for a database instance happening on a particular tablespace may be indicative of a problem scenario that needs further investigation.</td></tr>
<tr><td></td><td>Auto extensible:
Indicates whether the tablespace has the capability to grow automatically or not</td><td></td><td>If the tablespace is auto-extensible, then this measure will report the value Yes. If it is not extensible, then the value of this measure will be No.

The numeric values that correspond to the measure values discussed above are as follows:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>0</td></tr></table>
Note:

By default, the measure reports the Measure Values listed in the table above to indicate whether/not a tablespace is auto-extensible. In the graph of this measure however, the same is represented using the numeric equivalents only.</td></tr>
</table>

	Max size: Indicates the maximum extent (in MB) upto which a tablespace can grow	MB	
	Current size: Indicates the current allocated size of the tablespace	MB	If a tablespace is not auto-extensible, then its Current size will be equal to the Max size. For auto-extensible tablespaces though, the values of the Current size and Max size measures could be different.
	Free space: Indicates the amount of unused space in the tablespace. This is computed using the formula: **Max size - Current actual usage**, where **Current actual usage** is arrived at by applying the Current usage percentage on the Current size (current allocated size) measure. For example, assume that the Max size of a tablespace is 2500 MB and its Current size is 1000 MB. Also, note that nearly 30% of the Current size has already been utilized. Therefore, the **Current actual usage** of the tablespace will be 30% of 1000MB, which is 300 MB. The available Free space will hence be, 2500-300, i.e. 2200 MB.	MB	If this value is very low, then it indicates over-utilization of the tablespace.
	Percent free space: Indicates the space available for overall growth expressed as a ratio of Free_space with respect to the Max_size of the tablespace. The formula used is: **Free_space/Max_size*100**	Percent	If this value is very low, then it indicates over-utilization of the tablespace.

	Biggest extent: Indicates the size of the biggest extent in the tablespace	MB	From both these values, you can figure out how space allocation, fragmentation, etc. have been performed on the tablespace.
	Smallest extent: Indicates the size of the smallest extent in the tablespace	MB	
	Remaining extents: Indicates the number of extents that can be added to a tablespace	Number	If this value is low and the tablespace is not auto-extensible, then it indicates that the tablespace requires resizing. In the case of auto-extensible tablespaces, this phenomenon is not a cause for concern. This measure is not applicable to tablespaces that have dictionary based extent management and allocation type is *user*.

2.7.2 Oracle Temporary Tablespace Test

A temporary tablespace, contrary to what the name might indicate, does exist on a permanent basis as do other tablespaces, such as the SYSTEM and SYSAUX tablespaces. However the data in a temporary tablespace is of a temporary nature, which persists only for the length of a user session. Oracle uses temporary tablespaces as work areas for tasks such as sort operations for users and sorting during index creation. Oracle does not allow users to create objects in a temporary tablespace. By definition, the temporary tablespace holds data only for the duration of the user's session, and the data can be shared by all users.

Sufficient free space should be available in the temporary tablespace, as critical operations such as sorting and execution of hash-intensive queries may otherwise fail. Periodically checking the space usage in the temporary tablespaces will provide you with early warning signals of potential space contentions.

This test helps you to track the space usage of the temporary tablespace. Using this test, you can figure out the following:

- The size of the temporary tablespace
- The size that is allocated from this temporary tablespace for user operations
- How much of the allocated space is currently utilized by the user operation?
- What percentage of free space is currently available in the temporary tablespace?

Purpose	Helps you to track the space usage of the temporary tablespace
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **ALTERNATE VIEW** – In large environments, where the volume of transactions to the Oracle database server is generally very high, this test may take time to execute and retrieve the desired results. To ensure that the test is faster and is resource-efficient, administrators of such environments can create an alternate 'view' on the target Oracle database server, and grant *select* privileges to the view to the special database **USER** mentioned above. Once the view is created, the test should be configured to use the alternate view for metrics collection; to achieve this, specify the name of the view in the **ALTERNATE VIEW** text box. By default, this text box is set to *none*, which implies that the alternate view is not used by default.

This alternate 'view' should be created with the following structure:

```
CREATE OR REPLACE VIEW <VIEW_NAME> (
TABLESPACE_NAME,
FILE_ID,
BLOCK_ID,
BYTES,
BLOCKS,
RELATIVE_FNO
) AS
select /*+ use_hash (tsfi, fet2) */ tsfi.tablespace_name,
       tsfi.file_id,
       fet2.block_id,
       tsfi.blocksize * fet2.blocks,
       fet2.blocks,
       tsfi.relfile#

from   (select /*+ use_hash (ts, fi) */ ts.name tablespace_name,
               fi.file# file_id,
               ts.BLOCKSIZE,
               fi.relfile#,
               ts.ts#
        from   sys.ts$ ts,
               sys.file$ fi
        where  ts.ts# = fi.ts#
        and    ts.online$ in (1,4)) Tsfi,
       (select f.block# block_id,
               f.length blocks,
               f.file# file_id,
               f.ts#
        from   sys.fet$ f
        union all
        select f.ktfbfebno block_id,
               f.ktfbfeblks blocks,
               f.ktfbfefno,
               ktfbfetsn
        from   sys.x$ktfbfe f) Fet2
where  fet2.file_id = tsfi.relfile#
and    fet2.ts# = tsfi.ts# ;
```

8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every SID monitored.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Total size: Indicates the total size of the temporary tablespace.	MB	
	Allocated size: Indicates the space that is currently allocated for user operations (for e.g., sorting) from the temporary tablespace.	MB	When a user operation is initiated in the Oracle database, the whole of the temporary tablespace is not utilized for that operation. Instead, the database allocates a considerable amount of tempfiles from the temporary tablespace to perform such operations.
	Used allocated space: Indicates the space that is currently utilized by user operations from the allocated size of the temporary tablespace.	MB	
	Free allocated space: Indicates the free space that is still available for use in the allocated size of the temporary tablespace.	MB	Ideally, the value of this measure should be high.
	Total free space: Indicates the percentage of free space that is currently available in the temporary tablespace.	Percent	The value of this measure is calculated using the formula: *(Total_Free_Size/Total_Size)*100*. If the value of this measure is low, it indicates that the temporary tablespace is running out of space which will eventually lead to the failure of critical operations such as sorting and execution of hash-intensive queries. To avoid such failures, the size of the temporary tablespace should be increased. You can increase the size by just adding datafiles to the temporary tablespace using the **ADD TEMPFILE** command or through the **Add Datafiles** option from the Oracle Enterprise Manager. If free space is not available in the temporary tablespace, an error message stating - "ORA-1652: unable to extend temp segment" will appear.

2.7.3 Oracle Database Growth Test

Periodic monitoring of the usage of the database is essential to ensure that the database is always adequately sized to handle current and future loads. The Oracle Database Growth test monitors the usage of a managed Oracle database instance, and indicates if it requires resizing. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors the usage of a managed Oracle database instance, and indicates if it requires resizing
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **ALTERNATE VIEW** – In large environments, where the volume of transactions to the Oracle database server is generally very high, this test may take time to execute and retrieve the desired results. To ensure that the test is faster and is resource-efficient, administrators of such environments can create an alternate 'view' on the target Oracle database server, and grant *select* privileges to the view to the special database **USER** mentioned above. Once the view is created, the test should be configured to use the alternate view for metrics collection; to achieve this, specify the name of the view in the **ALTERNATE VIEW** text box. By default, this text box is set to *none*, which implies that the alternate view is not used by default.

This alternate 'view' should be created with the following structure:

```
CREATE OR REPLACE VIEW <VIEW_NAME> (
TABLESPACE_NAME,
FILE_ID,
BLOCK_ID,
BYTES,
BLOCKS,
RELATIVE_FNO
) AS
select /*+ use_hash (tsfi, fet2) */ tsfi.tablespace_name,
       tsfi.file_id,
       fet2.block_id,
       tsfi.blocksize * fet2.blocks,
       fet2.blocks,
       tsfi.relfile#

from   (select /*+ use_hash (ts, fi) */ ts.name tablespace_name,
               fi.file# file_id,
               ts.BLOCKSIZE,
               fi.relfile#,
               ts.ts#
        from   sys.ts$ ts,
               sys.file$ fi
        where  ts.ts# = fi.ts#
        and    ts.online$ in (1,4)) Tsfi,
       (select f.block# block_id,
               f.length blocks,
               f.file# file_id,
               f.ts#
        from   sys.fet$ f
        union all
        select f.ktfbfebno block_id,
               f.ktfbfeblks blocks,
               f.ktfbfefno,
               ktfbfetsn
        from   sys.x$ktfbfe f) Fet2
where  fet2.file_id = tsfi.relfile#
and    fet2.ts# = tsfi.ts# ;
```

8. **USE MAX SIZE** – Set this flag to **Yes**, if you want the *Free space*, *Space usage*, and *Space free* measures of this test to be computed based on the maximum size upto which a database can grow. Set this flag to **No**, so that the aforesaid measures are computed based on the space allocated to a database.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every SID monitored.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Total size of database: Indicates the total size of this database instance.	MB	
	Used space in database: Indicates the amount of database space that has been currently utilized.	MB	
	Free space in database: Indicates the amount of free space in this database instance currently.	MB	If the **USE MAX SIZE** parameter of this test has been set to **Yes**, then the value of this measure will include the amount of allocated space that is still unused by the database and the amount of space that will be available to the database if more free space is added to it until its maximum size is reached. If the **USE MAX SIZE** parameter of this test has been set to **No**, then the value of this measure will only indicate the amount of allocated space that is still unused by the database. In this case, the database's growth capacity will be disregarded.
	Space usage: Indicates the percentage of database space that has been utilized.	Percent	If the **USE MAX SIZE** parameter of this test has been set to **No**, then the value of this measure will be computed using the following formula: *Used space/Total size of database * 100* If the **USE MAX SIZE** parameter of this test has been set to **Yes**, then the value of this measure will be computed using the following formula: *Used space/Maximum size upto which the database can grow * 100* Ideally, this value should be low. A value close to 100% is a cause for concern.

	Space free: Indicates the percentage of free space in this database instance.	Percent	If the **USE MAX SIZE** parameter of this test has been set to **No**, then the value of this measure will be computed using the following formula: *Free space/Total size of database * 100* If the **USE MAX SIZE** parameter of this test has been set to **Yes**, then the value of this measure will be computed using the following formula: *Free space/Maximum size upto which the database can grow * 100* Ideally, this value should be high. A sudden/consistent decrease in the value of this measure could indicate excessive utilization of the database caused by a sporadic/steady increase in database activity. Very low free space in a database instance could significantly deteriorate its performance. Under such circumstances therefore, you might want to check the measures reported by the Oracle Datafile GrowthTest to figure out which datafile is consuming too much space. You might then want to resize the datafile.

2.7.4 Tablespace Status Test

The TablespaceStatus test helps determine the current status (whether available or not) of each of the tablespaces of the monitored database instance. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Helps determine the current status (whether available or not) of each of the tablespaces of the monitored database instance
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Online status:
Indicates the current state of this tablespace.</td><td>Percent</td><td>The value 100 for this measure indicates that the tablespace is in an ONLINE state. The value 0, on the other hand, indicates that the tablespace is in an OFFLINE state.</td></tr>
</table>

2.8 The Datafiles Layer

Since the datafiles contain the user data and the data dictionary and represent a major component of the database, monitoring the activity on the different datafiles is critical for optimizing database performance. For monitoring datafile activity, the eG Enterprise suite uses an OracleDataFiles test (see Figure 2.10). This test is intended to identify the level of activity that is happening on each datafile of a database. The results of the test can be used to reorganize the data storage, so as to balance the activity across the different datafiles and among the different physical disks in use.

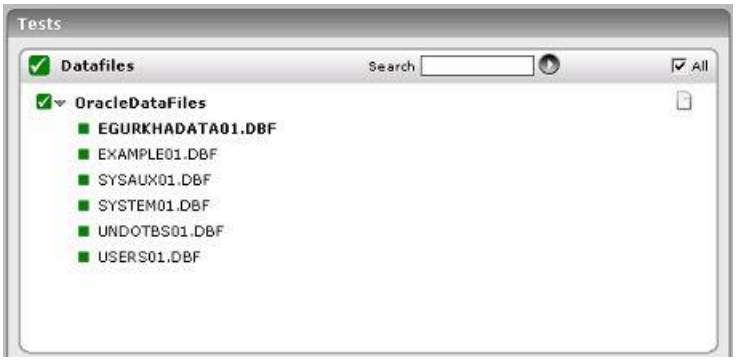


Figure 2.10: Tests mapping to the Datafiles layer

2.8.1 Oracle DataFiles Test

This test indicates the level of activity on a specific datafile in terms of the rate of physical reads and physical writes.

Purpose	This test indicates the level of activity on a specific datafile in terms of the rate of physical reads and physical writes.
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. INCLUDEPATH - This test reports a set of results for each datafile on the target Oracle database server. This means that every datafile is a descriptor of this test. By default, while displaying the descriptors of this test, the eG monitoring console does not prefix the datafile names with the full path to the datafiles. This is why, the INCLUDE PATH flag is set to No by default. If you want the data file names to be prefixed by the full path to the data files, then, set the INCLUDE PATH flag to Yes.
8. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Physical block read rate:** Indicates the rate at which disk blocks are being read from a specific datafile.	Blocks/Sec	A scenario in which more than 50% of blocks are being read from a single datafile could signify a problem.
	Physical block write rate: Indicates the rate at which disk blocks are being written to a specific datafile.	Blocks/Sec	A scenario in which more than 50% of blocks are being written to a single datafile could signify a problem. Too much activity to a specific datafile can result in reduced database performance. To improve performance, consider balancing I/O across disks, and reorganize tables across tablespaces to reduce activity to a specific datafile.
	Percent total I/O: Indicates the percentage of total I/O operations on the database server that were handled by a data file.	Percent	Disk reads and writes are expensive operations and all I/Os should be balanced across the different data files of an Oracle database for optimal performance. This metric reports the percentage of all I/O of an Oracle database that are happening on each of the data files of the Oracle database. This metric allows an Oracle administrator to determine which is/are the hot data file(s) (e.g., which data file is handling 80% of the total I/O).

2.8.2 Temporary Data Files Test

Temporary data files in Oracle are a special type of data file. Oracle uses temporary files to store the intermediate results of a large sort operation and hash operations, as well as to store global temporary table data, or result set data. If adequate space is not allocated or is not available to the temporary datafiles, it could cause abnormal termination of the key operations mentioned above, thereby rendering the database inaccessible.

This test periodically monitors the space usage of the temporary datafiles, and proactively alerts administrators to excessive space consumption by, or deficiencies in space allocations to, the temp datafiles.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Periodically monitors the space usage of the temporary datafiles, and proactively alerts administrators to excessive space consumption by the temp datafiles or deficiencies in space allocations to the temp datafiles
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every Oracle server being monitored</td></tr>
<tr><td>Measurements made by the</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

	Allocated size : This measure indicates the space allocated to temporary datafiles.	MB	
	Used size : This measure indicates the currently used space by temporary datafiles.	MB	
	Free space : This measure indicates the free space available to temporary datafiles.	MB	Ideally, the value of this measurement should be very high.
	Free space percentage: This measure indicates the percentage of space allocated to the temp datafiles, which is still unused.	Percent	Typically, a high percentage of free space is desired. A value close to 0 or a consistent decrease in the value of this measure could indicate excessive space consumption by the temporary datafiles or insufficient space allocation; lack of free space for temporary datafiles can severely affect database performance, and can even cause the database to hang! To avoid such adversities, you might want to consider allocating more space to the temporary datafiles.

2.8.3 Oracle DataFile Growth Test

Periodic monitoring of the usage of the database is essential to ensure that the database is always adequately sized to handle current and future loads. The OraDBFileGrowthTest monitors the usage of the datafiles that underlie a managed Oracle database instance, and indicates if any of the datafiles require resizing.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors the usage of the datafiles that underlie a managed Oracle database instance, and indicates if any of the datafiles require resizing
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **ALTERNATE VIEW** – In large environments, where the volume of transactions to the Oracle database server is generally very high, this test may take time to execute and retrieve the desired results. To ensure that the test is faster and is resource-efficient, administrators of such environments can create an alternate 'view' on the target Oracle database server, and grant *select* privileges to the view to the special database **USER** mentioned above. Once the view is created, the test should be configured to use the alternate view for metrics collection; to achieve this, specify the name of the view in the **ALTERNATE VIEW** text box. By default, this text box is set to *none*, which implies that the alternate view is not used by default.

This alternate 'view' should be created with the following structure:

```
CREATE OR REPLACE VIEW <VIEW_NAME> (
TABLESPACE_NAME,
FILE_ID,
BLOCK_ID,
BYTES,
BLOCKS,
RELATIVE_FNO
) AS
select /*+ use_hash (tsfi, fet2) */ tsfi.tablespace_name,
       tsfi.file_id,
       fet2.block_id,
       tsfi.blocksize * fet2.blocks,
       fet2.blocks,
       tsfi.relfile#

from   (select /*+ use_hash (ts, fi) */ ts.name tablespace_name,
               fi.file# file_id,
               ts.BLOCKSIZE,
               fi.relfile#,
               ts.ts#
        from   sys.ts$ ts,
               sys.file$ fi
        where  ts.ts# = fi.ts#
        and    ts.online$ in (1,4)) Tsfi,
       (select f.block# block_id,
               f.length blocks,
               f.file# file_id,
               f.ts#
        from   sys.fet$ f
        union all
        select f.ktfbfebno block_id,
               f.ktfbfeblks blocks,
               f.ktfbfefno,
               ktfbfetsn
        from   sys.x$ktfbfe f) Fet2
where  fet2.file_id = tsfi.relfile#
and    fet2.ts# = tsfi.ts# ;
```

8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every datafile monitored.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Datafile size: Indicates the current size of this datafile.	MB	
	Used space in datafile: Indicates the amount of database space that has been currently utilized by this datafile.	MB	
	Free space in datafile: Indicates the amount of free space currently available for this datafile.	MB	
	Space usage: Indicates the percentage of database space that has been utilized by this datafile.	Percent	Ideally, this value should be low. A value close to 100% is a cause for concern.
	Space free: Indicates the percentage of free space for this datafile.	Percent	Ideally, this value should be high. A sudden/consistent decrease in the value of this measure could indicate excessive utilization of the database caused by a sporadic/steady increase in database activity. Very low free space for a datafile could significantly deteriorate database performance. Under such circumstances therefore, you might want to resize that particular datafile or reorganize all the datafiles that are present in the managed database instance.

2.8.4 Oracle DataFile Activity Test

The average read and write time of Oracle metric is the amount of time spent for each read and write against the datafile. By comparing read and write times across multiple datafiles will show you which datafiles are slower than others and you can identify the hot files among them.

Note : The test should configure for run every 10 mins or more

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors the read and write time of datafiles

Target of the test	An Oracle 10g server
Agent deploying the test	An internal agent
Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query the Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

	8. **SHOW DATAFILE PATH** - This test reports a set of results for each datafile on the target Oracle database server. This means that every datafile is a descriptor of this test. By default, while displaying the descriptors of this test, the eG monitoring console does not prefix the datafile names with the full path to the datafiles. This is why, the **SHOW DATAFILE PATH** flag is set to **No** by default. If you want the data file names to be prefixed by the full path to the data files, then, set the **SHOW DATAFILE PATH** flag to **Yes**.		
Outputs of the test	One set of results for every datafile on the Oracle server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Average read time: This measure indicates the average time taken to read each datafile.	Secs	Disk read times might be high due to the following reasons. ▪ Executing inefficient queries for retrieving data; this could increase the frequency of full table scans and disk sorts, and can delay reading considerably ; ▪ Frequent insert and update operations on datafiles could cause data fragmentation Building efficient SQL queries can significantly increase the speed of your read operations. If fragmented data is the cause for the consistent slow-down in the read operations, then you might want to consider re-organizing the database objects to address this issue.
	Average write time: This measure indicates the average time taken to write each datafile.	Secs	

2.8.5 Oracle Database File Status Test

This test reports the status of each datafile in each database of an Oracle instance and the current access mode of every datafile.

Purpose	Monitors the read and write time of datafiles
Target of the test	An Oracle 10g server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query the Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every datafile on the Oracle server</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

<table>
<tr>
<td></td>
<td>File status:
Indicates the current status of this datafile.</td>
<td></td>
<td>The table below indicates the values that this measure can report and their corresponding numeric equivalents:

<table>
<tr><th>Numeric Value</th><th>Measure Value</th></tr>
<tr><td>1</td><td>System</td></tr>
<tr><td>2</td><td>Online</td></tr>
<tr><td>3</td><td>Recover</td></tr>
<tr><td>4</td><td>Unknown</td></tr>
</table>

If a datafile is part of the SYSTEM tablespace, its status is SYSTEM (unless it requires recovery).

If a datafile in a non-SYSTEM tablespace is online, its status is ONLINE. If a datafile in a non-SYSTEM tablespace is offline, its status can be either OFFLINE or RECOVER.

Note:

By default, this measure reports the above-mentioned Measure Values while indicating the current status of a datafile. However, in the graph of this measure, data file states will be represented using the corresponding numeric equivalents only - i.e., 1 to 4.</td>
</tr>
<tr>
<td></td>
<td>File access mode:
Indicates the current access mode of this datafile.</td>
<td></td>
<td>The table below indicates the values that this measure can report and their corresponding numeric equivalents:

<table>
<tr><th>Numeric Value</th><th>Measure Value</th></tr>
<tr><td>0</td><td>Disabled</td></tr>
<tr><td>1</td><td>Read Only</td></tr>
<tr><td>2</td><td>Read Write</td></tr>
<tr><td>3</td><td>Unknown</td></tr>
</table>

Note:

By default, this measure reports the above-mentioned Measure Values while indicating the mode through which this datafile can be accessed. However, the graph of this measure will be represented using the corresponding numeric equivalents i.e., 0 to 3.</td>
</tr>
</table>

2.8.6 Oracle Data File IO Statistics

If an Oracle datafile is able to process I/O requests to it quickly, it is a sign of the good health of the Oracle database server. On the other hand, any slowdown in IOPS could indicate a serious processing bottleneck on the server, probably caused by a poor indexing engine or badly structured tables in a datafile. Administrators should hence continuously track the I/O requests to every datafile on the Oracle database server, identify the type of requests received – i.e., whether single block or multi-block I/O requests - and measure the time taken by that datafile to process each type of request. For this purpose, you can run the **Oracle Data File IO Statistics** test.

This test auto-discovers the datafiles on the Oracle database server and reports the time taken by each datafile to process single block and multiblock I/O requests. In the process, I/O processing bottlenecks can be detected and the datafiles affected can be identified.

Purpose	Auto-discovers the datafiles on the Oracle database server and reports the time taken by each datafile to process single block and multiblock I/O requests. In the process, I/O processing bottlenecks can be detected and the datafiles affected can be identified
Target of the test	An Oracle 12c server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query the Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. LISTENER NAME – Specify the Oracle listener name. By default, this will be the same as the Oracle SID.
8. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every datafile on the Oracle server</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

	Multiblock read time: Indicates the time taken by this datafile to service multiblock I/O requests during the last measurement period.	Secs/Read	Multiblock I/O read means reading multiple database blocks with a single operating system READ call. Typically, a database block is 8 KB. A single block read call results in one of these 8 KB blocks read from the datafile. Where a lot of data is to be read, it would be less efficient and more resource-intensive to read single blocks of data of 8KB each when the underlying I/O system is capable of reading say, 1 MB in one read. Oracle therefore issues a multiblock I/O and requests 1MB worth of block (128 8kb blocks) in one system READ call rather than 128 individual requests and therefore speeds up performance of the I/O requests. A very high value of this measure could indicate a bottleneck when processing multiblock read requests to a particular datafile. Compare the value of this measure across files to accurately identify that datafile from which multiple blocks of data were read from most slowly. In the event of high latency when processing read requests, you can do one/more of the following to clear the processing bottleneck: • Create tables and indexes in separate tablespaces. • Create datafiles across multiple disks. • Create table partitions across multiple datafiles.
	Singleblock read time: Indicates the time taken for singleblock reads from this datafile during the last measurement period.	Secs/read	Typically, a database block is 8 KB. A single block read call results in one of these 8 KB blocks read from the datafile. Where a lot of data is to be read, it would be less efficient and more resource-intensive to read single blocks of data of 8KB each when the underlying I/O system is capable of reading say, 1 MB in one read. A very high value of this measure could indicate a bottleneck when processing single block write requests to a particular datafile. Compare the value of this measure across files to accurately identify that datafile from which a single block of data was read most slowly.

	Multiblock write time: Indicates the time taken for multiblock writes into this datafile during the last measurement period.	Secs/write	Multiblock I/O write means writing multiple database blocks to a datafile with a single operating system WRITE call. A very high value of this measure could indicate a bottleneck when processing multiblock write requests to a particular datafile. Compare the value of this measure across files to accurately identify that datafile to which multiple blocks of data were written most slowly. If the write latency is very high, you can do one/more of the following to clear the processing bottleneck: • Create tables and indexes in separate tablespaces. • Create datafiles across multiple disks. • Create table partitions across multiple datafiles.
	Singleblock write time: Indicates the time taken for singleblock writes to this datafile during the last measurement period.	Secs/write	In case of a singleblock write, a write call results in a single 8KB block being written into the datafile. If a lot of data is to be written to a datafile, single block writes can significantly increase I/O processing overheads and related resource costs. A very high value of this measure could indicate a bottleneck when processing single block write requests to a particular datafile. Compare the value of this measure across files to accurately identify that datafile to which a single block of data was written most slowly.
	Sync read latency: Indicates the average latency for singleblock synchronous reads for single request since last test cycle on each datafile.	Msecs/request	If there is a high latency for critical data files, you may want to consider relocating these files to improve their service time.
	IO time since last measure: Indicates the time taken by IOPS on this datafile during the last measurement period.	Secs	A high value could indicate a processing bottleneck with the datafile. Compare the value of this measure across datafiles to identify that datafile the read/write requests to which take too long to be serviced.

2.8.7 Oracle Dead Kill Processes Test

If one/more sessions or processes on the Oracle server are obstructing the execution of a few other sessions/processes, then, it is quiet natural for administrators to want to kill the blocking sessions/processes to ensure the smooth execution of critical database transactions. Typically, these 'dead' sessions/processes continue to consume resources, until the

PMON process automatically cleans up these sessions/processes. If cleanup is delayed, then the Oracle instance will not be able to release those objects and resources that have been locked by the dead sessions/processes for long time periods. In such situations, administrators often resort to killing these dead sessions/processes at the operating system-level, so as to hasten the release of valuable resources. Before attempting the OS-level kill, administrators should first figure out which sessions/processes are 'dead' presently and how long they have been 'dead'. This can be ascertained using the **Oracle Dead Kill Processes** test. This test auto-discovers the dead processes/sessions and reports the current cleanup state of each process/session. In addition, the test reveals the duration for which each process/session remained dead and the count of processes that are being blocked by that dead process/session. This way, administrators can determine whether/not cleanup is occurring as per schedule, and if not, how badly the delay in cleanup is affecting other processes. Alongside, administrators can figure out whether an OS-level process kill is justified or not.

Purpose	Auto-discovers the dead processes/sessions and reports the current cleanup state of each process/session. In addition, the test reveals the duration for which each process/session remained dead and the count of processes that are being blocked by that dead process/session. This way, administrators can determine whether/not cleanup is occurring as per schedule, and if not, how badly the delay in cleanup is affecting other processes. Alongside, administrators can figure out whether an OS-level process kill is justified or not.
Target of the test	An Oracle 12c server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query the Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. LISTENER NAME – Specify the Oracle listener name. By default, this will be the same as the Oracle SID.
8. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for deadprocessaddress_deadsessionaddress on the Oracle instance monitored</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Process state:

Indicates the current cleanup state of this process.

The values that this measure can report and their corresponding numeric values have been discussed hereunder:

Measure Value	Description	Numeric Value
UNSAFE TO ATTEMPT	Occurs for a killed session that has not been moved, so no cleanup can occur on it yet	1
CLEANUP PENDING	Occurs for a dead process / killed session that can be cleaned up, but PMON has not yet made an attempt	2
RESOURCES FREED	Occurs for a dead process / killed session where all children have been freed, but the process / killed session itself is not yet freed	3
RESOURCES FREED – PENDING ACK	- Occurs for a killed session where all children have been freed, but the session itself cannot be freed until the owner has acknowledged it	4
PARTIAL CLEANUP	Occurs if some of the children have been cleaned up	5

			Note: By default, this measure reports the above-mentioned **Measure Value**s while indicating the current cleanup state of a dead process. However, in the graph of this measure, the same will be represented using the corresponding numeric equivalents only.
	Dead time: Indicates how long it has been since this process was marked dead or this session was marked killed.	Secs	A consistent increase in the value of this measure is a cause for concern as it indicates that auto-cleanup has not occurred. This can cause the dead process/session to continue consuming resources and blocking object, thereby degrading server performance.
	Number blocked: Indicates the count of processes that are blocked by this process.	Number	A high value indicates that the dead process is impeding the execution of many other processes, some of which may also be mission-critical. If the *Dead time* of such a process is also very high, it is a matter of great concern, and must be looked into immediately. In such circumstances, you may want to consider killing the process at the OS-level. On a Unix system, you can issue the **KILL -9 <PID>** command at the Shell prompt to kill the process at that level.

2.8.8 Oracle IO Latency Test

Functions such as direct reads, direct writes, buffer cache reads, DBWR etc., often generate a high level of I/O activity on the storage sub-system of Oracle. If the Oracle storage is not sized right to handle the I/O load, then one/more of these mission-critical functions may significantly slowdown (i.e., take more than 500 milliseconds to complete), thus delaying critical database operations. In the event of such a slowdown therefore, administrators must be able to quickly and accurately pinpoint the latent function and determine what is ailing that function, so that the configuration of the Oracle server or its storage sub-system can be fine-tuned to avoid such anomalies. This is where the **Oracle IO Latency** test helps. This test automatically identifies those functions that are taking more than 500 milliseconds to complete. For each of these functions, this test reports the size of the I/O generated by that function and exactly how much time Oracle's storage sub-system takes to process this I/O load. In the process, the test turns the spotlight on the latent functions and reveals if the high I/O latency is due to a poorly configured storage system.

Purpose	Automatically identifies those functions that that are taking more than 500 milliseconds to complete. For each of these functions, this test reports the size of the I/O generated by that function and exactly how much time Oracle's storage sub-system takes to process this I/O load. In the process, the test turns the spotlight on the latent functions and reveals if the high I/O latency is due to a poorly configured storage system.
Target of the test	An Oracle 12c server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query the Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **LISTENER NAME** – Specify the Oracle listener name. By default, this will be the same as the Oracle SID. 8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
Outputs of the test	One set of results for every I/O function that is taking more than 500 milliseconds to complete

Measurements made by the	Measurement	Measurement Unit	Interpretation
	IO size: Indicates the current size of the I/O generated by this function.	Bytes	This is a good indicator of the current I/O workload on the Oracle storage.
	IO latency: Indicate the total I/O latency of this function.	Msecs	Compare the value of this measure across functions to know which function is the most latent.
	Average IO latency: Indicates the average I/O latency of this function.	Msecs	This represents the time taken by the storage sub-system to process a single byte of I/O requests for the function. A very high value is indicative of the inability of the storage system to process requests for a function quickly. This could be because the storage system is not configured with adequate space. You may want to consider resizing the storage system to ensure better I/O throughput.
	Max IO latency: Indicates the maximum I/O latency for a single byte of requests for this function.	Msecs	A high value is a cause for concern, as it indicates a potentially latent function.

2.8.9 Oracle Other File IO Statistics Test

An Oracle database can typically consist of data files, control files, redo log files, temporary files, archive log files, and files of many other types. If I/O requests to any of these files experience processing bottlenecks, it is bound to adversely impact user experience with the Oracle database server. If this is to be avoided, administrators should closely track read/write requests to each of these files, measure how quickly the server handles these requests, and initiate pre-emptive action upon the first sign of a processing latency. This is where the eG agent helps. The eG agent periodically runs the **Oracle Datafile IO Statistics** test and points administrators to latencies in I/O requests to datafiles. Likewise, at configured intervals, the eG agent runs the **Oracle Temporary File IO Statistics** test to enable administrators to spot latencies when processing requests to temporary files.

Similarly, to determine whether/not requests for any of the other files in an Oracle database are processed slowly, administrators can configure the eG agent to run the **Oracle Other File IO Statistics** test at regular intervals. This test auto-discovers the archive log files, redo log files, control files, etc., in the Oracle databases and reports the time taken by the server for processing single block and multiblock I/O requests to each file. This way, the test points you to current/probable latencies when processing I/O requests to archive, redo log, control files, and others. The exact file, when reading from/writing to which, the latency was maximum can also be identified.

Purpose	Auto-discovers the archive log files, redo log files, control files, etc., in the Oracle databases and reports the time taken by the server for processing single block and multiblock I/O requests to each file. This way, the test points you to current/probable latencies when processing I/O requests to archive, redo log, control files, and others. The exact file, when reading from/writing to which, the latency was maximum can also be identified

Target of the test	An Oracle 12c server
Agent deploying the test	An internal agents
Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query the Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **LISTENER NAME** – Specify the Oracle listener name. By default, this will be the same as the Oracle SID. 8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
Outputs of the test	One set of results for every file other than datafiles and temp files on the Oracle server

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Multiblock read time: Indicates the time taken by this file to service multiblock I/O requests during the last measurement period.	Secs/Read	Multiblock I/O read means reading multiple database blocks with a single operating system READ call. Typically, a database block is 8 KB. A single block read call results in one of these 8 KB blocks read from the datafile. Where a lot of data is to be read, it would be less efficient and more resource-intensive to read single blocks of data of 8KB each when the underlying I/O system is capable of reading say, 1 MB in one read. Oracle therefore issues a multiblock I/O and requests 1MB worth of block (128 8kb blocks) in one system READ call rather than 128 individual requests and therefore speeds up performance of the I/O requests. A very high value of this measure could indicate a bottleneck when processing multiblock read requests to a particular file. Compare the value of this measure across files to accurately identify that file from which multiple blocks of data were read from most slowly.
	Singleblock read time: Indicates the time taken for singleblock reads from this file during the last measurement period.	Secs/read	Typically, a database block is 8 KB. A single block read call results in one of these 8 KB blocks read from the datafile. Where a lot of data is to be read, it would be less efficient and more resource-intensive to read single blocks of data of 8KB each when the underlying I/O system is capable of reading say, 1 MB in one read. A very high value of this measure could indicate a bottleneck when processing single block read requests to a particular file. Compare the value of this measure across files to accurately identify that file from which a single block of data was read from most slowly.
	Multiblock write time: Indicates the time taken for multiblock writes into this file during the last measurement period.	Secs/write	Multiblock I/O write means writing multiple database blocks to a file with a single operating system WRITE call. A very high value of this measure could indicate a bottleneck when processing multiblock write requests to a particular file. Compare the value of this measure across files to accurately identify that file to which multiple blocks of data were written most slowly.

	Singleblock write time: Indicates the time taken for singleblock writes to this file during the last measurement period.	Secs/write	In case of a singleblock write, a write call results in a single 8KB block being written into the file. If a lot of data is to be written to a file, single block writes can significantly increase I/O processing overheads and related resource costs. A very high value of this measure could indicate a bottleneck when processing single block write requests to a particular file. Compare the value of this measure across files to accurately identify that file to which a single block of data was written most slowly.
	Sync read latency: Indicates the average latency for singleblock synchronous reads for single request since last test cycle on this file.	Msecs/request	If there is a high latency for critical files, you may want to consider relocating these files to improve their service time.
	IO time since last measure: Indicates the time taken by IOPS on this file during the last measurement period.	Secs	A high value could indicate a processing bottleneck. Compare the value of this measure across files to identify that file, the reads and writes to which take the maximum time.

2.8.10 Oracle PDB Status Test

The multitenant architecture enables an Oracle database to function as a multitenant container database (CDB) that includes zero, one, or many customer-created pluggable databases (PDBs). A PDB is a portable collection of schemas, schema objects, and nonschema objects that appears to an Oracle Net client as a non-CDB. All Oracle databases before Oracle Database 12c were non-CDBs.

A container is either a PDB or the root container (also called the root). The root is a collection of schemas, schema objects, and nonschema objects to which all PDBs belong.

Every CDB has the following containers:

- Exactly one root

  The root stores Oracle-supplied metadata and common users. An example of metadata is the source code for Oracle-supplied PL/SQL packages. A common user is a database user known in every container. The root container is named `CDB$ROOT`.

- Exactly one **seed PDB**

  The seed PDB is a system-supplied template that the CDB can use to create new PDBs. The seed PDB is named `PDB$SEED`. You cannot add or modify objects in `PDB$SEED`.

- Zero or more user-created PDBs

  A PDB is a user-created entity that contains the data and code required for a specific set of features. For example, a PDB can support a specific application, such as a human resources or sales application. No PDBs exist at creation of the CDB. You add PDBs based on your business requirements.

If a user is experiencing errors when attempting to open a PDB, administrators must be able to quickly check the status of the PDB to figure out the reason for the error. For this purpose, administrators can use the **Oracle PDB Status** test. This test automatically discovers the PDBs and reports the current status and mode of every PDB.

Purpose	Automatically discovers the PDBs and reports the current status and mode of every PDB
Target of the test	An Oracle 12c server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query the Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. LISTENER NAME – Specify the Oracle listener name. By default, this will be the same as the Oracle SID.
8. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every PDB on the Oracle server</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

<table>
<tr>
<td></td>
<td>Status:
Indicates the current status of this PDB.</td>
<td></td>
<td>The values that this measure can report and their corresponding numeric values are discussed hereunder:
<table>
<tr><th>Measure Value</th><th>Description</th><th>Numeric Value</th></tr>
<tr><td>NEW</td><td>The PDB has never been opened since it was created. It must be opened in<code>READ WRITE</code> mode for Oracle to perform processing needed to complete the integration of the PDB into the CDB and mark it <code>NORMAL</code>. An error will be thrown if an attempt is made to open the PDB read only</td><td>1</td></tr>
<tr><td>NORMAL</td><td>The PDB is ready to be used</td><td>2</td></tr>
<tr><td>UNPLUGGED</td><td>The PDB has been unplugged. The only operation that can be performed on it is <code>DROP PLUGGABLE DATABASE</code>.</td><td>3</td></tr>
<tr><td>NEEDS UPGRADE</td><td>A PDB needs to be upgraded to the version of the CDB into which it was plugged</td><td>4</td></tr>
<tr><td>CONVERTING</td><td>A non-CDB was plugged into the CDB and is undergoing conversion required to make it behave like a real PDB.</td><td>5</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>
<table>
<tr><th>Measure Value</th><th>Description</th><th>Numeric Value</th></tr>
<tr><td>UNUSABLE</td><td>The PDB is being created or an unrecoverable error was encountered during its creation. The PDB cannot be opened while its state is set to UNUSABLE. If the PDB remains in this state because of an error encountered during its creation, it can only be dropped. The alert log can be checked to determine if there was an error during PDB creation.</td><td>6</td></tr>
</table>
Note:
By default, this measure reports the above-mentioned Measure Values while indicating the current state of a PDB. However, in the graph of this measure, the same will be represented using the corresponding numeric equivalents only.
</td>
</tr>
<tr>
<td></td>
<td>Mode:
Indicates the mode in which this PDB has been opened currently.</td>
<td></td>
<td>
The values that this measure can report and their corresponding numeric values are discussed hereunder:
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>MOUNTED</td><td>1</td></tr>
<tr><td>READ WRITE</td><td>2</td></tr>
<tr><td>READ ONLY</td><td>3</td></tr>
<tr><td>MIGRATE</td><td>4</td></tr>
</table>
Note:
By default, this measure reports the above-mentioned Measure Values while indicating the mode in which the PDB is opened. However, in the graph of this measure, the same will be represented using the corresponding numeric equivalents only.
</td>
</tr>
</table>

2.8.11 Oracle Temp File IO Statistics Test

Temp files are a special class of data files that are associated only with temporary tablespaces. Locally managed temporary tablespaces use **temp files**, which do not modify data outside of the temporary tablespace or generate any redo for temporary tablespace data. Because of this, they enable you to perform on-disk sorting operations in a read-only or standby database.

If IOPS performed on the temp files take too much time, administrators must be able to quickly and accurately identify the exact temp file to which read/write operations are most latent and the type of I/O operation that was performed on that file (i.e., whether a multiblock read/write or a single block read/write) when latency peaked. This will enable administrators to determine the course of action that needs to be taken to ensure that the I/O latency does not aggravate. This insight is provided by the **Oracle Temp File IO Statistics** test. This test automatically discovers the temp files, and for each temp file reports the time taken to read and write single and multiple blocks of data in the file. This will point administrators to that temp file on which read/write operations take longer than normal. From this test, you can also infer when read/write latency is maximum – when reading a single block of data? Or when reading multiple blocks of data? When writing a single block of a data to the file? Or when writing multiple blocks of data to the file?

Purpose	Automatically discovers the temp files, and for each temp file reports the time taken to read and write single and multiple blocks of data in the file. This will point administrators to that temp file on which read/write operations take longer than normal. From this test, you can also infer when read/write latency is maximum – when reading a single block of data? Or when reading multiple blocks of data? When writing a single block of a data to the file? Or when writing multiple blocks of data to the file?
Target of the test	An Oracle 12c server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query the Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. LISTENER NAME – Specify the Oracle listener name. By default, this will be the same as the Oracle SID.
8. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every temp file on the Oracle server</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

	Multiblock read time: Indicates the time taken by this file to service multiblock I/O requests during the last measurement period.	Secs/Read	Multiblock I/O read means reading multiple database blocks with a single operating system READ call. Typically, a database block is 8 KB. A single block read call results in one of these 8 KB blocks read from the datafile. Where a lot of data is to be read, it would be less efficient and more resource-intensive to read single blocks of data of 8KB each when the underlying I/O system is capable of reading say, 1 MB in one read. Oracle therefore issues a multiblock I/O and requests 1MB worth of block (128 8kb blocks) in one system READ call rather than 128 individual requests and therefore speeds up performance of the I/O requests. A very high value of this measure could indicate a bottleneck when processing multiblock read requests to a particular file. Compare the value of this measure across files to accurately identify that file from which multiple blocks of data were read from most slowly.
	Singleblock read time: Indicates the time taken for singleblock reads from this file during the last measurement period.	Secs/read	Typically, a database block is 8 KB. A single block read call results in one of these 8 KB blocks read from the datafile. Where a lot of data is to be read, it would be less efficient and more resource-intensive to read single blocks of data of 8KB each when the underlying I/O system is capable of reading say, 1 MB in one read. A very high value of this measure could indicate a bottleneck when processing single block read requests to a particular file. Compare the value of this measure across files to accurately identify that file from which a single block of data was read from most slowly.
	Multiblock write time: Indicates the time taken for multiblock writes into this file during the last measurement period.	Secs/write	Multiblock I/O write means writing multiple database blocks to a file with a single operating system WRITE call. A very high value of this measure could indicate a bottleneck when processing multiblock write requests to a particular file. Compare the value of this measure across files to accurately identify that file to which multiple blocks of data were written most slowly.
	Singleblock write time: Indicates the time taken for singleblock writes to this file during the last measurement period.	Secs/write	In case of a singleblock write, a write call results in a single 8KB block being written into the file. If a lot of data is to be written to a file, single block writes can significantly increase I/O processing overheads and related resource costs. A very high value of this measure could indicate a bottleneck when processing single block write requests to a particular file. Compare the value of this measure across files to accurately identify that file to which a single block of data was written most slowly.

	Sync read latency: Indicates the average latency for singleblock synchronous reads for single request since last test cycle on this file.	Msecs/request	If there is a high latency for critical files, you may want to consider relocating these files to improve their service time.
	IO time since last measure: Indicates the time taken by IOPS on this file during the last measurement period.	Secs	A high value could indicate a processing bottleneck. Compare the value of this measure across files to identify that file, the reads and writes to which take the maximum time.

2.9 The Oracle Service Layer

This layer tracks the overall health of the service offered by the database server to clients. As indicated earlier, the availability and responsiveness of the database server are measured using the OracleSqlNet test. An additional OracleSessions test reports session-level information regarding the usage of the database server (see Figure 2.11).

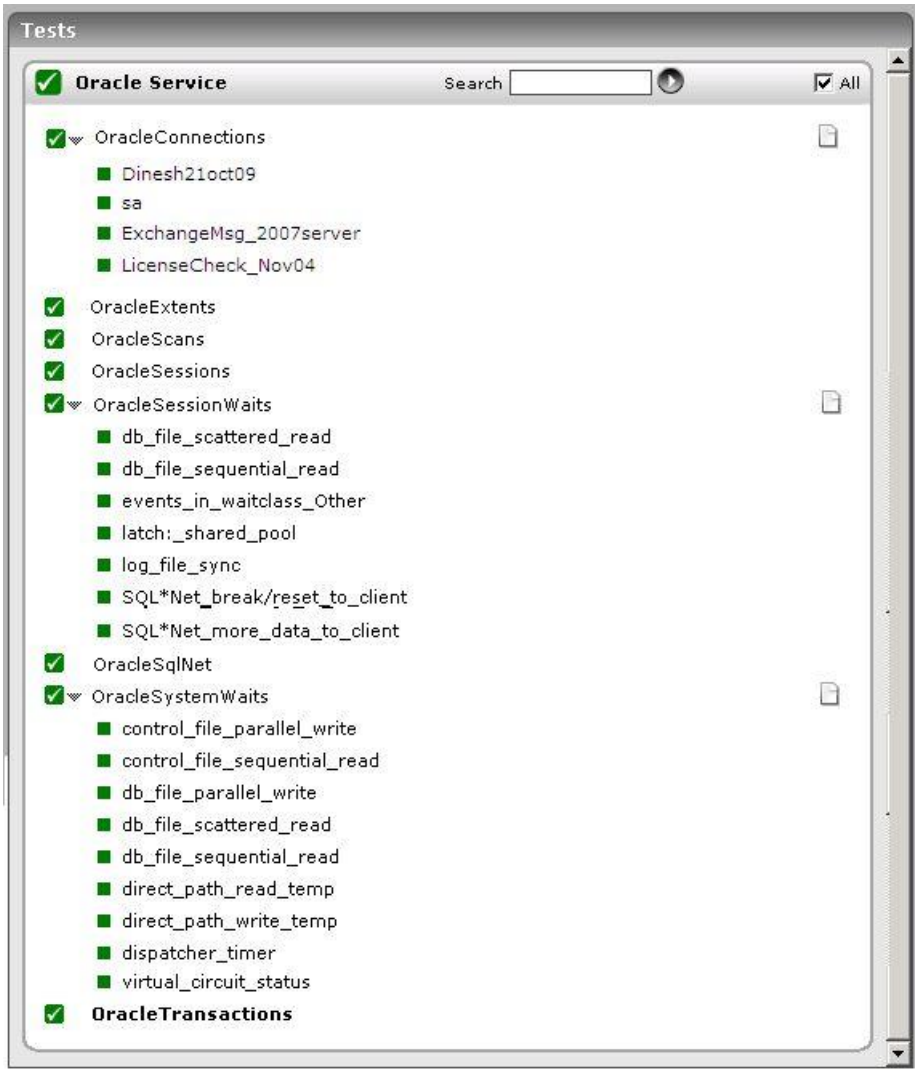


Figure 2.11: Tests mapping to the Oracle Service layer

2.9.1 Oracle User Connections Test

This test reports the number and state of sessions of each user who is currently connected to the Oracle database server. Using the metrics reported by this test, administrators can promptly isolate idle sessions, which are a drain on a server's resources.

Purpose	Reports the number and state of sessions of each user who is currently connected to the Oracle database server
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.

<table>
<tr><td></td><td colspan="3">8. EXCLUDEUSER - In the EXCLUDEUSER text box, specify a comma-separated list of user names that need to be excluded from monitoring. By default, none is displayed here indicating that this test monitors connections initiated by all current users to the MS SQL server, by default.
9. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every user who is currently connected to the Oracle server</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total connections:
Indicates the total number of connections currently established by this user on the server.</td><td>Number</td><td></td></tr>
<tr><td>Active connections:
Indicates the number of connections of this user that are currently active.</td><td>Number</td><td>The detailed diagnosis of this measure, if enabled, will provide the complete details of the active sessions of a particular user. Using this information, you can understand how each of the connections were made - i.e., using which program - and from where - i.e., from which host.</td></tr>
<tr><td>Inactive connections:
Indicates the number of sessions initiated by this user that are currently idle.</td><td>Number</td><td>Ideally, the value of this measure should be low. A high value is indicative of a large number of idle sessions, which in turn causes the unnecessary consumption of critical server resources. Idle sessions also unnecessarily lock connections from the connection pool, thereby denying other users access to the server for performing important tasks.
The detailed diagnosis of this measure, if enabled, will provide the complete details of the idle sessions of a particular user. Using this information, you can understand how each of the idle connections were made - i.e., using which program - and from where - i.e., from which host.</td></tr>
</table>

	Background connections: Indicates the number of background processes that were started when sessions are initiated by this user.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the background sessions of a particular user. Using this information, you can understand how each of the background connections were made - i.e., using which program - and from where - i.e., from which host.
	Blocked connections: Indicates the number of sessions initiated by this user were blocked.	Number	Blocking occurs when one session holds a lock on a resource that another session is requesting. As a result, the requesting session will be blocked - it will hang until the holding session gives up the locked resource. In almost every case, blocking is avoidable. In fact, if you find that your session is blocked in an interactive application, then you have probably been suffering from the lost update bug as well, perhaps without realizing it. That is, your application logic is flawed and that is the cause of blocking. The five common DML statements that will block in the database are **INSERT**, **UPDATE**, **DELETE**, **MERGE** and **SELECT FOR UPDATE**. Ideally, the value of this measure should be low. A high value may cause unnecessary consumption of critical server resources thereby blocking access to potential active sessions. The detailed diagnosis of this measure, if enabled, will provide the complete details of the blocked sessions of a particular user. Using this information, you can understand how each of the blocked connections were made - i.e., using which program - and from where - i.e., from which host.

	Cached connections: Indicates the number of sessions of this user that were cached for future use.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the cached sessions of a particular user. Using this information, you can understand how each of the cached connections were made - i.e., using which program - and from where - i.e., from which host.
	Killed connections: Indicates the number of sessions of this user that were terminated due to inactivity.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the killed sessions of a particular user. Using this information, you can understand how each of the killed connections were made - i.e., using which program - and from where - i.e., from which host.
	Sniped connections: Indicates the number of sessions of this user that were idle for a period more than the profile's maximum idle time while waiting for a client's response.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the sniped sessions of a particular user. Using this information, you can understand how each of the sniped connections were made - i.e., using which program - and from where - i.e., from which host.

2.9.2 Oracle Extents Test

An extent in Oracle is a set of contiguous blocks allocated in Oracle for storage of data. Since this is one of the basic units of allocation, proper management of extents is essential for efficient database performance. The OracleExtents test helps in identifying objects that are running out of extents and those that are using too many extents.

This test is disabled by default. go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

This test is applicable only for Oracle databases with 'Dictionary Managed Tablespace'. From Oracle 9i onwards, all tablespaces are 'Locally Managed Tablespaces'. Therefore, this test is applicable only upto Oracle 8i.

Purpose	To identify the objects that are running out of extents and those that are using too many extents
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **MAXEXTENT** - This test reports a **Large extent objects** measure, which reveals the number of objects that exceed a pre-configured number of extents; this limit is set using the **MAXEXTENT** parameter. If you enter a number in the **MAXEXTENT** text box, then, the **Large extent objects** measure of this test will return the count of objects with more extents than the number specified in the **MAXEXTENT** text box. By default, the **MAXEXTENT** parameter is set to 1000.

8. **ALTERNATE VIEW** – In large environments, where the volume of transactions to the Oracle database server is generally very high, this test may take time to execute and retrieve the desired results. To ensure that the test is faster and is resource-efficient, administrators of such environments can create an alternate 'view' on the target Oracle database server, and grant *select* privileges to the view to the special database **USER** mentioned above. Once the view is created, the test should be configured to use the alternate view for metrics collection; to achieve this, specify the name of the view in the **ALTERNATE VIEW** text box. By default, this text box is set to *none*, which implies that the alternate view is not used by default.

   This alternate 'view' should be created with the following structure:

```
CREATE OR REPLACE VIEW <VIEW_NAME> (
TABLESPACE_NAME,
FILE_ID,
BLOCK_ID,
BYTES,
BLOCKS,
RELATIVE_FNO
) AS
select /*+ use_hash (tsfi, fet2) */ tsfi.tablespace_name,
       tsfi.file_id,
       fet2.block_id,
       tsfi.blocksize * fet2.blocks,
       fet2.blocks,
       tsfi.relfile#

from   (select /*+ use_hash (ts, fi) */ ts.name tablespace_name,
               fi.file# file_id,
               ts.BLOCKSIZE,
               fi.relfile#,
               ts.ts#
        from   sys.ts$ ts,
               sys.file$ fi
        where  ts.ts# = fi.ts#
        and    ts.online$ in (1,4)) Tsfi,
       (select f.block# block_id,
               f.length blocks,
               f.file# file_id,
               f.ts#
        from   sys.fet$ f
        union all
        select f.ktfbfebno block_id,
               f.ktfbfeblks blocks,
               f.ktfbfefno,
               ktfbfetsn
        from   sys.x$ktfbfe f) Fet2
where  fet2.file_id = tsfi.relfile#
and    fet2.ts# = tsfi.ts# ;
```

9.

<table>
<tr><td></td><td colspan="3">10. 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 enabled/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.

11. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every Oracle server</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Cannot extend objects:
This measure indicates the number of objects that cannot extend any further by acquiring new extents.</td><td>Number</td><td>This measure indicates the objects have run out of extents. This could be because the objects are either too fragmented or the number of extents allocated to them are too less. Consider modifying the "maxextents" parameter to rectify this. Alternately if fragmentation is the cause then, consider exporting and then dropping and re-importing the object.

The detailed diagnosis of the Cannot extend objects measure, if available, provides a complete list of objects that cannot be extended. Once the objects are identified, administrators can then consider increasing the extents allocated to the listed objects.</td></tr>
<tr><td>Large extent objects:
This measure indicates the object that exceeds a pre-specified number of extents. The threshold value for the number of extents is configured via the admin user interface.</td><td>Number</td><td>This indicates that the objects are using up more extents than the threshold set. This over-utilization can be due to different types of fragmentation. Consider exporting and then dropping and re-importing the object. Note that the storage parameters set also affects how the extents are allocated.

The detailed diagnosis of the Large extent objects measure reveals the list of objects that exceed a pre-specified number of extents.</td></tr>
</table>

2.9.3 Oracle Session Waits Test

The test monitors the session level wait events on the Oracle database server and reports key performance statistics pertaining to every event. Effective wait analysis helps determine where the database spends most of its time, and which current connections are responsible for the reported waits.

Purpose	To monitor the session level wait events on the Oracle database server
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **EXCLUDE** - Provide a comma-separated list of wait events that need not be monitored. For example, your specification can be: *buffer_busy_waits,SQL*Net_message_from_client*. By default, 'none' is displayed here indicating that all wait events are monitored, by default. 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.

<table>
<tr><td></td><td colspan="3">The option to selectively enabled/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.
9. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every session wait event monitored on the Oracle server</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>New waits:
Indicates the total number of times waits happened on this event since the last measurement period.</td><td>Number</td><td>If the value of this measure is very high, then you can drill down further using the detailed diagnosis capability (if enabled) of the eG Enterprise suite to discover which current connections may be responsible for this. The detailed diagnosis of this measure reveals the session IDs of the sessions that caused the wait events to occur, the users who initiated the sessions, and the total number of waits, wait time, and the maximum wait time for every session.</td></tr>
<tr><td>Total waits timedout:
Indicates the total number of waits on this event that timed out since the last measurement period.</td><td>Number</td><td>A large number of timed out wait events is typically, undesirable. Use Oracle-specific documentation to probe the cause of the timeout.</td></tr>
<tr><td>Avg time waited:
Indicates the average duration for which the waits on this wait event persisted since the last measurement period.</td><td>Secs</td><td>Ideally, the value of this measure should be low. A very high value or a consistent increase in this value is indicative of a problem situation requiring further investigation. Use the detailed diagnosis capability to zoom into the session that has contributed to the abnormal increase in wait time.</td></tr>
<tr><td></td><td>Max time waited:
Indicates the high watermark of wait time for this wait event.</td><td>Secs</td><td></td></tr>
</table>

2.9.4 Oracle System Waits Test

The test monitors the system level wait events on the Oracle database server and reports key performance statistics pertaining to every event. Effective wait analysis helps determine where the database spends most of its time, and which current connections are responsible for the reported waits.

Purpose	To monitor the system level wait events on the Oracle database server
Target of the test	An Oracle server
Agent deploying the test	An internal agent
Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **EXCLUDE** - Provide a comma-separated list of wait events that need not be monitored. For example, your specification can be: *Data_file_init_write,db_file_single_write*. By default, 'none' is displayed here indicating that all system wait events are monitored, by default. 8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every system wait event monitored on the Oracle server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	New waits: Indicates the total number of times waits happened on this event system-wide, since the last measurement period.	Number	High waits indicate a problem, but not always. Sometimes waits are just a normal part of database operations. For example, high waits on 'db file sequential read' events may indicate a disk bottleneck, but you must check your average disk queue length for each disk spindle to be sure that these waits are abnormal. If a high number of waits are observed on a specific event, you can use the detailed diagnosis capability of the OraSessionWaitTest to figure out whether any current connections have contributed to the increase in waits.
	Total waits timedout: Indicates the total number of waits on this event that timed out since the last measurement period.	Number	A large number of timed out wait events is typically, undesirable. Use the Oracle-specific documentation to probe the cause of the timeout.
	Avg time waited: Indicates the average duration for which the waits on this wait event persisted since the last measurement period.	Secs	By comparing the value of this measure across all monitored wait events, you can determine where the database spends most of its time.
	Time waited: Indicates the total amount of time for which the waits on this wait event persisted.	Secs	

2.9.5 Oracle Sessions Test

In the database context, the connection between the user process and the server process is called a session. The server process communicates with the connected user process and performs tasks on behalf of the users. The OracleSessions test is used by an eG agent to track user activity related to a database server instance.

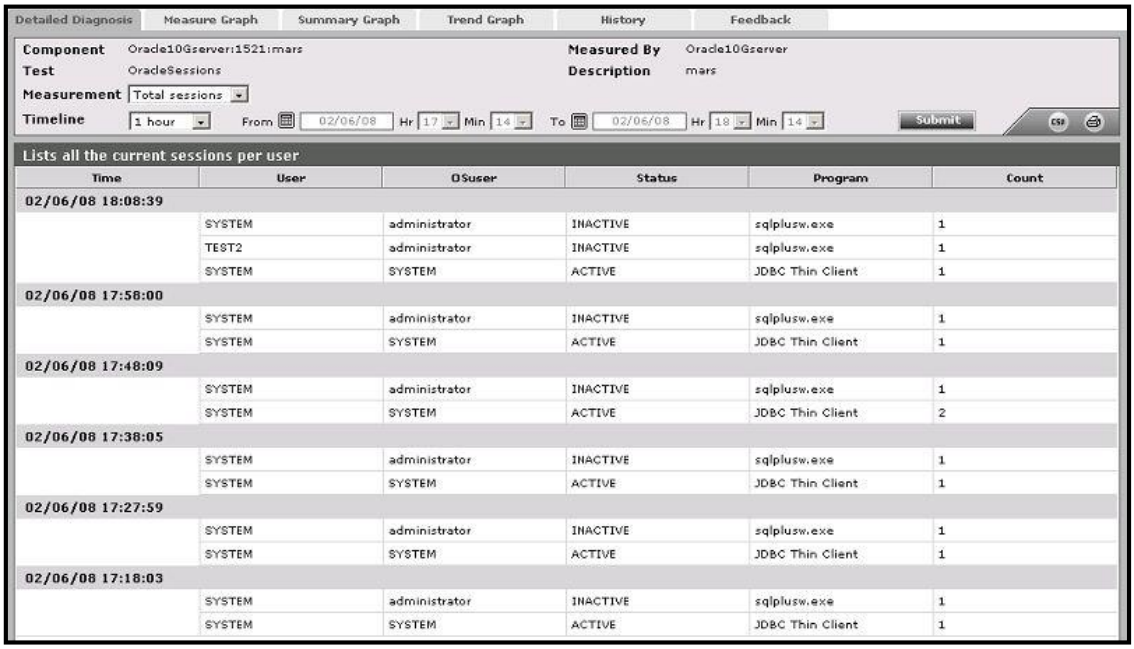
Purpose	This test indicates the level of activity on a database in terms of the number of active sessions and inactive sessions.
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.

	8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every SID monitored.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Total sessions: Indicates the total number of users connected to the database server.	Number	A high value may indicate that there is a high load on the server.
	Active sessions: Indicates the number of sessions that are currently accessing the database.	Number	A high value may indicate that there is a high load on the server.
	Background sessions: Indicates the number of sessions that are created when the database starts.	Number	A high value may indicate that there is a high load on the server. The detailed diagnosis capability, if enabled, lists all the background sessions.
	Blocked sessions: Indicates the number of sessions that were blocked in this database.	Number	Blocking occurs when one session holds a lock on a resource that another session is requesting. As a result, the requesting session will be blocked - it will hang until the holding session gives up the locked resource. In almost every case, blocking is avoidable. In fact, if you find that your session is blocked in an interactive application, then you have probably been suffering from the lost update bug as well, perhaps without realizing it. That is, your application logic is flawed and that is the cause of blocking. The five common DML statements that will block in the database are **INSERT**, **UPDATE**, **DELETE**, **MERGE** and **SELECT FOR UPDATE**. Ideally, the value of this measure should be zero. The detailed diagnosis capability, if enabled, lists all the blocked sessions of this database.
	Cached sessions: Indicates the number of sessions that were temporarily cached for use by this database.	Number	A high value may indicate that there is a high load on the server. The detailed diagnosis capability, if enabled, lists all the cached sessions of this database.

	Inactive sessions: Indicates the number of sessions that were inactive in this database.	Number	Ideally, the value of this measure should be zero. The detailed diagnosis capability, if enabled, lists all the inactive sessions in this database.
	Killed sessions: Indicates the number of inactive sessions that were terminated in this database.	Number	When a session is terminated, any active transactions of the session are rolled back, and resources held by the session (such as locks and memory areas) are immediately released and available to other sessions. A low value is desired for this measure. The detailed diagnosis capability, if enabled, lists all the sessions that were killed in this database.
	Sniped sessions: Indicates the number of sessions that were idle for a period more than the profile's maximum idle time and were waiting for a response from the client.	Number	The idle time is the time limit that is provided against the *IDLE_TIME* parameter in the user's profile or the default profile. A low value is desired for this measure. The detailed diagnosis capability, if enabled, lists all the sniped sessions of this database.

The detailed diagnosis of the *Total sessions* measure, if enabled, lists all the current user sessions to the Oracle server (see Figure 2.12). Using this information, administrators to the database can identify the number of user sessions that are inactive, and can terminate such sessions.



Time	User	OSuser	Status	Program	Count
02/06/08 18:08:39					
	SYSTEM	administrator	INACTIVE	sqlplusw.exe	1
	TEST2	administrator	INACTIVE	sqlplusw.exe	1
	SYSTEM	SYSTEM	ACTIVE	JDBC Thin Client	1
02/06/08 17:58:00					
	SYSTEM	administrator	INACTIVE	sqlplusw.exe	1
	SYSTEM	SYSTEM	ACTIVE	JDBC Thin Client	1
02/06/08 17:48:09					
	SYSTEM	administrator	INACTIVE	sqlplusw.exe	1
	SYSTEM	SYSTEM	ACTIVE	JDBC Thin Client	2
02/06/08 17:38:05					
	SYSTEM	administrator	INACTIVE	sqlplusw.exe	1
	SYSTEM	SYSTEM	ACTIVE	JDBC Thin Client	1
02/06/08 17:27:59					
	SYSTEM	administrator	INACTIVE	sqlplusw.exe	1
	SYSTEM	SYSTEM	ACTIVE	JDBC Thin Client	1
02/06/08 17:18:03					
	SYSTEM	administrator	INACTIVE	sqlplusw.exe	1
	SYSTEM	SYSTEM	ACTIVE	JDBC Thin Client	1

Figure 2.12: The detailed diagnosis of the Total sessions measure

The detailed diagnosis of the *Active sessions* measure, if enabled, lists all the active user sessions to the Oracle server (see Figure 2.13).

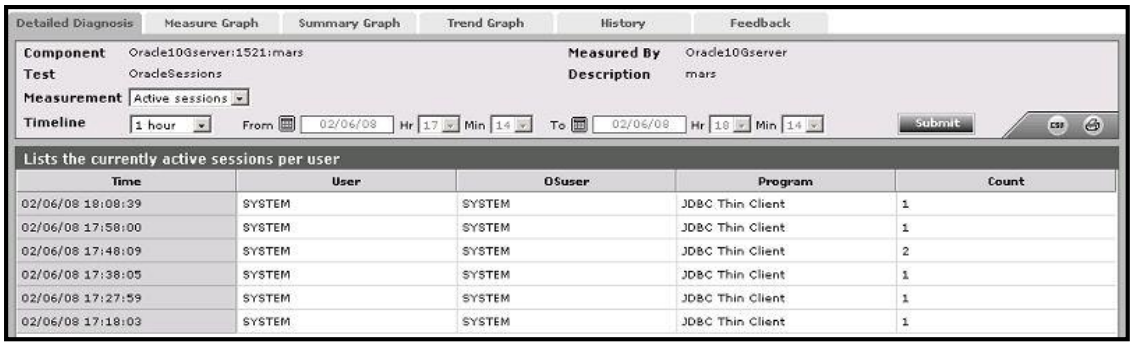


Figure 2.13: The detailed diagnosis of the Active sessions measure

2.9.6 Oracle Scans Test

Full table scans on a database instance can degrade the performance of the database. This test monitors the extent of full table scans happening on the database.

Purpose	This test monitors the extent of full table scans happening on the database.
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Percent long table scans:** The percentage of long table scans happening in the database	Percent	Ideally, this value should be lower than 10%. If more than 20% of scans are happening on long tables, the database/accesses to the database may need to be tuned. Full table scans may happen due to several reasons. For instance, the indexes of a table may not be used properly in queries. By tuning the queries, the full table scans can be reduced and the database performance significantly improved.
	Long table scans: The number of long table scans that happened on the database instance during the last measurement period	Number	
	Short table scans: The number of short table scans that happened on the database instance during the last measurement period	Number	

	Full table scans: The number of full table scans that happened on the database instance during the last measurement period.	Number	This type of scan reads all rows from a table and filters out those that do not meet the selection criteria. There are two types of full-table scans, those against small tables STR-FTS and large-tables LT-FTS. The rule for evaluative and tuning LT-FTS is simple. We evaluate the query and see if index access would result in less physical reads than the existing full-table scan. This usually involves timing the execution speed for the query (with the set timing on command in SQL*Plus) and timing the query with different index access plans: • **Creating a function-based index** - One common technique is to match the WHERE clause of the query with a function-based index. • **Using index hints** - If the CBO does not have enough statistical information about an index, you can force the CBO (temporarily) to use the index by adding an index hint to the query. Once the fastest execution plan is derived, the tuning professional will enforce the execution plan by creating schema statistics to ensure that the CBO will always use the best index access. The problem with ST-FTS occurs when a popular table is referenced. Because the FTS data blocks are not touched (pinged to the MRU end of the buffer), ST-FTS rows age quickly from the buffer, requiring Oracle to re-read them, over and over again.

			In Oracle9i and beyond hidden parameter called *_adaptive_direct_read* that ensures that small table scans are cached. However, it is still a good idea to identify these small tables yourself and cache them in your KEEP pool. The KEEP pool is a wonderful resource for ensuring that an object always resides in the data buffer RAM, and this is one of the few ways to guarantee 10% caching. Now that we see the benefit of caching frequently-referenced table and indexes, we see how the KEEP pool is most important to small objects that are read into the data buffers via full-table scans. Also, remember that frequently-referenced data blocks accessed via an index will tend to remain in the data buffer without using the KEEP pool because they are pinged to the MRU end of the buffer every time they are referenced.

2.9.7 Oracle RAC Session Waits Test

Like the wait activity on stand-alone Oracle servers, administrators also need to observe the wait activity on Oracle servers that are part of a Real Application Cluster (RAC). This test connects to the global view on the monitored Oracle server in an RAC, and pulls out critical statistics pertaining to the session-level cluster-related events that wait on the global cache or any other global resource on that Oracle server. Using this information, administrators can determine the cluster events on which the database spends most of its time, and which Oracle instances and current connections are responsible for the reported waits. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

This test needs to be enabled only while monitoring the Oracle servers in an RAC environment.

Purpose	Monitors session-level cluster-related events on Oracle servers in an RAC environment
Target of the test	An Oracle server in an RAC environment
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **EXCLUDE** - Provide a comma-separated list of wait events that need not be monitored. For example, your specification can be: *gc cr request,gc buffer busy*. By default, 'none' is displayed here indicating that all wait events are monitored, by default. 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 enabled/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. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every cluster-related session wait event monitored on the Oracle server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Total waits: Indicates the total number of times waits happened on this event since the last measurement period.	Number	If the value of this measure is very high, then you can drill down further using the detailed diagnosis capability (if enabled) of the eG Enterprise suite to discover which Oracle instances and current connections may be responsible for this. The detailed diagnosis of this measure reveals the Oracle instance on which the events are waiting, the session IDs of the sessions that caused the wait events to occur, the users who initiated the sessions, and the total number of waits, wait time, and the maximum wait time for every session.
	Total waits timedout: Indicates the total number of waits on this event that timed out since the last measurement period.	Number	A large number of timed out wait events is typically, undesirable. Use Oracle-specific documentation to probe the cause of the timeout.
	Avg time waited: Indicates the average duration for which the waits on this wait event persisted since the last measurement period.	Secs	Ideally, the value of this measure should be low. A very high value or a consistent increase in this value is indicative of a problem situation requiring further investigation. Use the detailed diagnosis capability to zoom into the session that has contributed to the abnormal increase in wait time.
	Max time waited: Indicates the high watermark of wait time for this wait event.	Secs	

2.9.8 Oracle RAC System Waits Test

Like the wait activity on stand-alone Oracle servers, administrators also need to observe the wait activity on Oracle servers that are part of a Real Application Cluster (RAC). This test connects to the global view on an Oracle server in an RAC, and pulls out information pertaining to the system-level cluster-related events that wait on the global cache

or any other global resource on that Oracle server. Using this information, administrators determine the cluster events on which the database spends most of its time, and which current connections are responsible for the reported waits.

This test needs to be enabled only while monitoring the Oracle servers in an RAC environment.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	To monitor the cluster-related system-level wait events on an Oracle server in an RAC environment
Target of the test	An Oracle server in an RAC environment
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. EXCLUDE - Provide a comma-separated list of wait events that need not be monitored. For example, your specification can be: gc cr request,gc buffer busy. By default, 'none' is displayed here indicating that all system wait events are monitored, by default.
8. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every system wait event monitored on the Oracle server</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Total waits:** Indicates the total number of times waits happened on this event system-wide, since the last measurement period.	Number	High waits indicate a problem, but not always. Sometimes waits are just a normal part of database operations. For example, high waits on 'db file sequential read' events may indicate a disk bottleneck, but you must check your average disk queue length for each disk spindle to be sure that these waits are abnormal. If a high number of waits are observed on a specific event, you can use the detailed diagnosis capability of the OraSessionWaitTest to figure out whether any current connections have contributed to the increase in waits.
	Total waits timedout: Indicates the total number of waits on this event that timed out since the last measurement period.	Number	A large number of timed out wait events is typically, undesirable. Use the Oracle-specific documentation to probe the cause of the timeout.
	Avg time waited: Indicates the average duration for which the waits on this wait event persisted since the last measurement period.	Secs	By comparing the value of this measure across all monitored wait events, you can determine where the database spends most of its time.

2.9.9 Oracle Objects Test

The OracleObjects test is used to monitor one or more database user accounts. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors one or more database user accounts
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **ACCOUNTS** - Specify the database user account to be monitored. Multiple database user accounts can be provided as a comma-separated list. 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 enabled/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.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every user account monitored.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Invalid objects: The number of invalid objects in the database for a specific user account. The detailed diagnosis capability, if turned on, provides the list of invalid objects and their type.	Number	The detailed diagnosis capability for this measure, if enabled, lists the names and the type of the invalid objects of a user.
	Modified objects: The number of objects for a specific user account that have been modified in the last measurement period	Number	This measure allows changes to the database structure to be tracked over time. The detailed diagnosis capability, if enabled, indicates the names of the objects that have been modified and their type.

The detailed diagnosis of the *Invalid objects* measure lists the invalid objects in the database for a user (see Figure 2.14). By dropping such invalid objects, tablespace can be conserved.

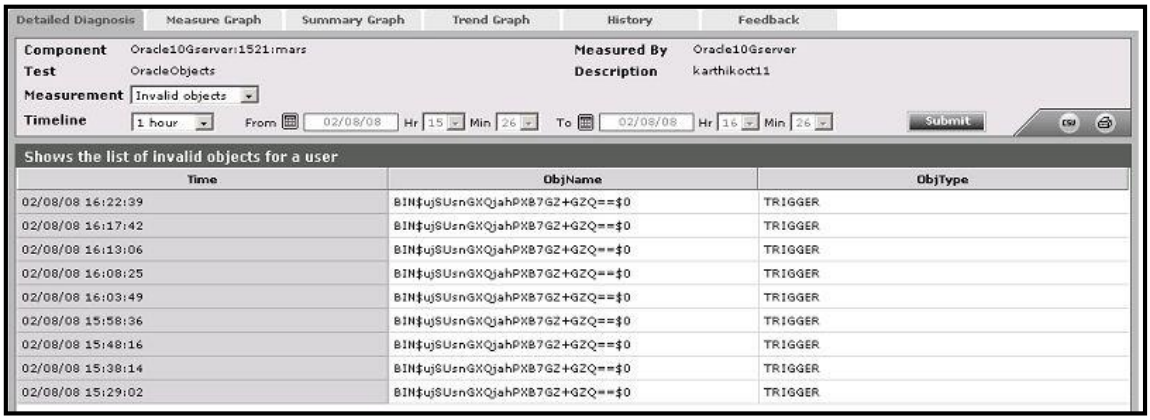


Figure 2.14: The detailed diagnosis of the Invalid objects measure

2.9.10 Oracle User Tablespaces Test

This test measures the number of users currently using the system tablespace. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	To measure the number of users currently using the system tablespace
Target of the test	An Oracle server

Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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. 8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every instance of an Oracle database		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	System tablespace users: Indicates the number of users (other than sys/system users) who are currently using the system tablespace as their default or temporary tablespace	Number	It is not desirable for application users to be using system tablespace as their default or temporary tablespace. If you want the eG Enterprise suite to ignore a few users while measuring the number of current users, then set the **Maximum Threshold** accordingly. By default, the **Maximum Threshold** is set to "none". For eg., OUTLN and DBSNMP are two default users of the system tablespace. Therefore, the value of this measure will also be 2, by default. Moreover, since the **Maximum Threshold** of this measure, is by default "none", any value greater than 0, will trigger an alarm. Since the value 2 is greater than 0, an alarm will be generated for this measure. To avoid such an alarm-generation, change the **Maximum Threshold** to 2. When this is done, the eG Enterprise system will ignore the 2 default users and will generate an alarm only when additional users begin using the system tablespace.

The detailed diagnosis of the *System tablespace users* measure, if enabled, lists the users who are currently using the system tablespace (see Figure 2.15). When there is a steep increase in the number of system tablespace users, then, this information will help in identifying the specific users using the tablespace. Such users, if found unwanted, can be dropped. Alternatively, the default or temporary tablespace of such users can be changed.

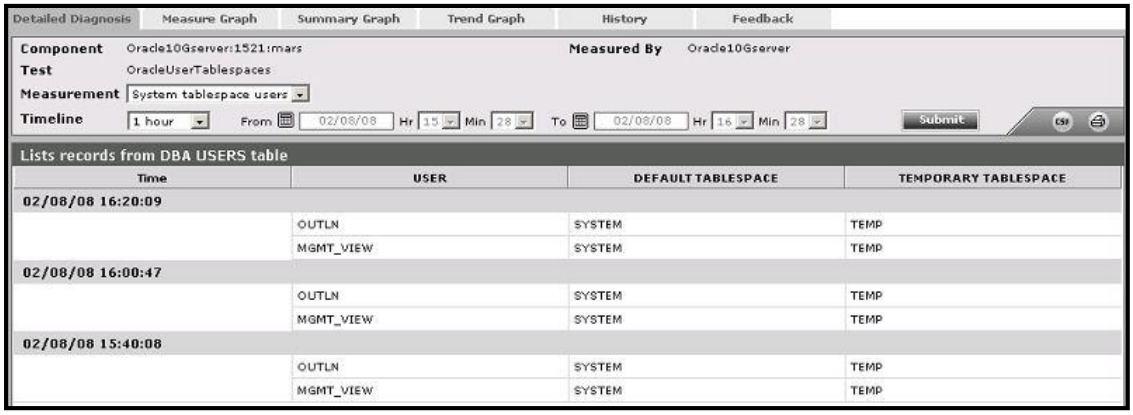


Figure 2.15: The detailed diagnosis of the System tablespace users measure

2.9.11 Oracle TS Parameters Test

This test measures the number of tablespaces that are not locally managed. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick

Oracle Database as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

This test reports the count and details of 'dictionary managed tablespaces' only. Since Oracle 9i (and above) supports 'Locally Managed Tablespaces alone, this test is applicable only up to Oracle 8i.

Purpose	To measure the number of tablespaces that are not locally managed
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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. 8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every instance of an Oracle database		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Nonlocal managed tablespaces: Indicates the number of tablespaces that are not locally managed	Number	

2.9.12 Oracle Transactions Test

Rollbacks are costly operations on the database. This test monitors the percentage of rollbacks happening for user transactions with a database instance.

Purpose	Monitors the percentage of rollbacks happening for user transactions with a database instance
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">
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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>User commits:
The number of user commits that have happened during the last measurement period</td><td>Number</td><td></td></tr>
</table>

	User rollbacks: The number of user rollbacks that have happened during the last measurement period	Number	Ideally, there should be few user rollbacks happening. Typically, whenever a delete, insert or update operation is performed on the database, Undo tablespace is consumed, I/O overheads increase, and considerable server time is spent in performing that operation. When such operations are rolledback, these resources are wasted! To conserve resources, its best to keep rollbacks at a minimum.
	Percent rollbacks: The number of user rollbacks as a percentage of the total user transactions (user commits + user rollbacks) with the database	Percent	The closer the percentage of rollbacks is to zero, the lower the overhead on the database due to rollbacks. The acceptable value of rollbacks will vary from one instance to another and will have to be configured based on the patterns of requests being handled by the database instance.

2.9.13 Oracle Parameters Test

This test tracks changes made to the default values of parameters. This test is disabled by default. To enable the test, open the **AGENTS – TESTS CONFIGURATION** page using the Agents -> Tests -> Configure menu sequence, select *Oracle Database* from the **Select a component type** list, go to the **DISABLED TESTS** section, click on the check box preceding this test, and finally, click on the **Update** button.

Purpose	To track the changes made to the default values of parameters
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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. 8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every instance of an Oracle database		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Non-default parameters: Indicates the number of parameters for which the default value has changed.	Number	Changes in this value enables users to track changes made to the database settings. In case you want to ignore a few parameters, change the thresholding policy for this measure, to the above effect.

The detailed diagnosis capability of the *Non-default parameters* measure, if enabled, lists the parameters that have not retained their default values (see Figure 2.16). This information helps an administrator in tracking the changes made to the parameters.

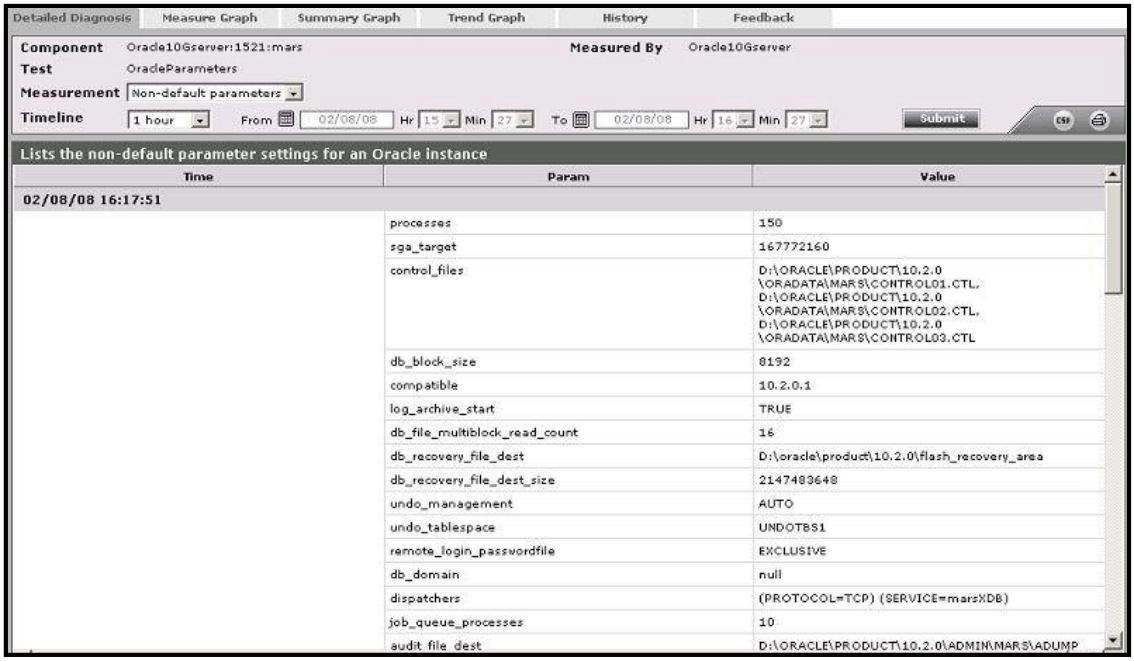


Figure 2.16: The detailed diagnosis of the Non-default parameters measure

2.9.14 Oracle Archive Test

This test tracks the mode on which the database is running. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	To track the mode on which the database is running
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every instance of an Oracle database</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Archive log mode:** Indicates whether the database is running in the archive log mode or not.	Number	If this value is 0, then it denotes that the database is not running in the archive log mode. The value 100, on the other hand, indicates that the database is running in the archive log mode. In case you do not want to be alerted when running in the "no-archive log" mode, change the thresholding policy accordingly.

2.9.15 Oracle Alerts Test

This oracle-specific test periodically tracks the errors newly added to the Oracle alert log. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	To view the errors newly added to the alert log
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 server listens 4. **ALERTFILE -** By default, this is set to *none*, indicating that the eG agent auto-discovers the path to the Oracle alert log file to be monitored. If required, you can manually specify the full path to the alert log file to be monitored. For eg, /user/john/alert_egurkha.log 5. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 6. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 7. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 8. **SEARCHPATTERN** - Enter the specific patterns of alerts to be monitored. The pattern should be in the following format: *<PatternName>:<Pattern>*, where *<PatternName>* is the pattern name that will be displayed in the monitor interface and *<Pattern>* is an expression of the form - **expr** or expr or **expr or expr**, etc. A leading '*' signifies any number of leading characters, while a trailing '*' signifies any number of trailing characters. For example, say you specify ORA:ORA-* in the SEARCHPATTERN text box. This indicates that "ORA" is the pattern name to be displayed in the monitor interface. "ORA-*" indicates that the test will monitor only those lines in the alert log which start with the term "ORA-". Similarly, if your pattern specification reads: offline:*offline, then it means that the pattern name is offline and that the test will monitor those lines in the alert log which end with the term offline.

<table>
<tr><td></td><td colspan="3">Multiple search patterns can be specified as a comma-separated list. For example: ORA:ORA-*,offline:*offline*,online:*online.

Specify all if all Oracle alerts are to be monitored.

9. LINES - Specify two numbers in the format x:y. This means that when a line in the alert file matches a particular pattern, then x lines before the matched line and y lines after the matched line will be reported in the detail diagnosis output (in addition to the matched line). The default value here is 0:0. Multiple entries can be provided as a comma-separated list.

If you give 1:1 as the value for LINES, then this value will be applied to all the patterns specified in the SEARCHPATTERN field. If you give 0:0,1:1,2:1 as the value for LINES and if the corresponding value in the SEARCHPATTERN filed is like ORA:ORA-*,offline:*offline*,online:*online then:

0:0 will be applied to ORA:ORA-* pattern

1:1 will be applied to offline:*offline* pattern

2:1 will be applied to online:*online pattern

10. EXCLUDE PATTERN - Provide a comma-separated list of message patterns to be excluded from monitoring. For instance, if you want to monitor all alert messages that begin with "ORA-", except the messages that begin with "ORA—" and "ORA-info", you can configure ORA-* as the SEARCHPATTERN and configure ORA--,ORA-info as the EXCLUDE PATTERN.

11. 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 enabled/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.

12. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every alert log file</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Recent errors:
Indicates the number of new errors that have been written by oracle to the alert log file.</td><td>Number</td><td>The value of this measure is a clear indicator of the number of "new" alerts that have come into the alert log of the monitored database.</td></tr>
</table>

	File size: Indicates the current size of the alert log file.	MB	**This measure will only be reported for the 'Summary' descriptor of this test.**
	Growth rate : Indicates the rate at which the alert log file is growing	MB/Sec	**This measure will only be reported for the 'Summary' descriptor of this test.** A high value for this measure or a consistent increase in its value indicates that the alert log is rapidly growing and may end up occupying too much space on the volume. Under such circumstances, it is recommended that you delete the alert log file and then issue a log file switch, so that Oracle automatically creates a new alert log file for you the next time a database activity needs to be logged.

2.9.16 Idle Oracle Sessions Test

Inactive sessions to an Oracle database server are serious resource-drainers! Such sessions do not execute any transactions, but consume resources significantly, thereby depriving critical database-related operations of the resources. Idle sessions should hence be identified promptly and terminated quickly, so as to prevent the unnecessary locking of resources.

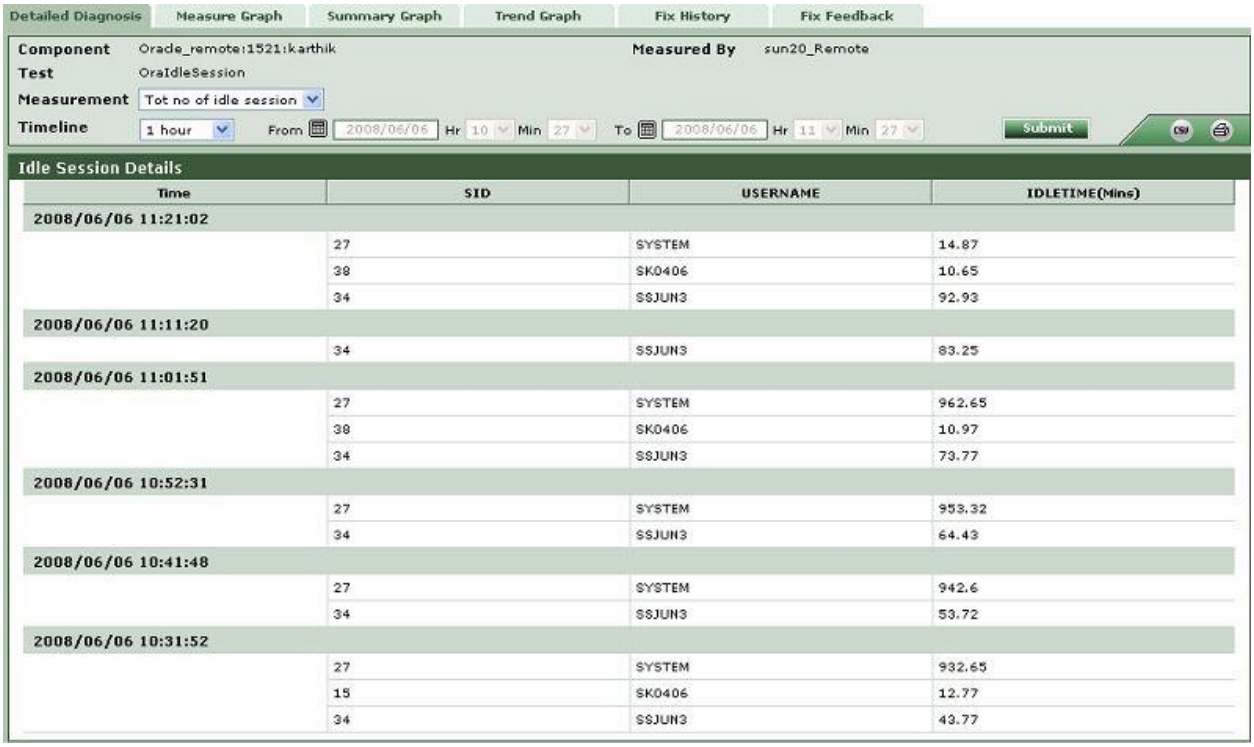
The IdleOracleSessionsTest keeps an eye out for idle sessions, and alerts administrators as soon as an idle session is detected. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Alerts administrators whenever an idle session is detected
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 8. **INACTIVE PERIOD** - Specify the duration (in minutes) of inactivity beyond which a session is considered to be "idle" by this test. By default, this parameter takes the value 10 (minutes); this implies that by default, the test counts all sessions that have been inactive for over 10 minutes as idle sessions. 9. **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 enabled/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.		
Outputs of the test	One set of results for every instance of an Oracle database		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Current idle sessions: Indicates the number of idle sessions; **note that all sessions that have been passive beyond the INACTIVE PERIOD configured for this test will be counted as idle sessions by this test..**	Number	Ideally, the value of this measure should be 0. A high value indicates that a number of sessions are idle and using up resources unnecessarily.

Using the detailed diagnosis of this measure, you can quickly identify the idle sessions and terminate them, so that resources are released for the use of critical processes. Alternatively, you can set a lower idle time in the user profile, so that the user session automatically aborts upon reaching the set idle time.



Time	SID	USERNAME	IDLETIME(Mins)
2008/06/06 11:21:02			
	27	SYSTEM	14.87
	38	SK0406	10.65
	34	SSJUN3	92.93
2008/06/06 11:11:20			
	34	SSJUN3	83.25
2008/06/06 11:01:51			
	27	SYSTEM	962.65
	38	SK0406	10.97
	34	SSJUN3	73.77
2008/06/06 10:52:31			
	27	SYSTEM	953.32
	34	SSJUN3	64.43
2008/06/06 10:41:48			
	27	SYSTEM	942.6
	34	SSJUN3	53.72
2008/06/06 10:31:52			
	27	SYSTEM	932.65
	15	SK0406	12.77
	34	SSJUN3	43.77

Figure 2.17: The detailed diagnosis of the IdleOracleSessions Test

2.9.17 Oracle Long Running Queries Test

This test tracks the currently executing queries on an Oracle database server and determines the number of queries that have been running for a long time. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Tracks the currently executing queries on an Oracle database server and determines the number of queries that have been running for a long time
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 8. **ELAPSED TIME** - In the **ELAPSED TIME** text box, specify the duration (in seconds) for which a query should have executed for it to be regarded as a long running query. The default value is 10. 9. **DISPLAYQUERY FULLTEXT** - The detailed diagnosis of this test lists the queries that have been running for a long time. In the **DETAILED DIAGNOSIS** page by default, query strings that are very long are truncated to display the first 1000 characters of the query alone. This is why, the **DISPLAYQUERY FULLTEXT** flag is set to **No** by default. To view the full query in the detailed diagnosis page, set this flag to **Yes**. **Note that setting this flag to 'Yes' may increase the size of your eG database**.

<table>
<tr><td></td><td colspan="3">10. 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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every instance of an Oracle database</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Long running queries:
Indicates the number of queries currently executing on the database server that have been running for more time than the configured ELAPSED TIME.</td><td>Number</td><td>The detailed diagnosis for this measure indicates the exact queries and which user is executing the queries. This information can be very useful in identifying queries that may be candidates for optimization.</td></tr>
</table>

2.9.18 Oracle Dump Area Test

Trace files, typically used for troubleshooting issues with key database operations, are stored in dump area destinations marked for every such operation. For instance, a *background dump destination* can be specified using the BACKGROUND_DUMP_DEST initialization parameter in Oracle; trace files for the background processes are written to this destination only. Similarly, trace files for user processes are generated and stored in the *user dump destination*, which is set using the USER_DUMP_DEST parameter in Oracle.

The dump destinations so created should be adequately sized, so that there is always enough space in the destination directory for storing trace files. If any of the destination directories become full, then trace files cannot be created for the corresponding database operation; while the absence of trace files can make debugging difficult, in some cases, it can even bring the database operations to a standstill.

In order to avoid such anomalies, the usage of each dump destination should be monitored, and administrators promptly alerted to space inadequacies, so that required space is made available in the dump directory. The OraDumpArea test serves this purpose effectively. This test runs periodic checks on the usage of every dump destination that has been configured for monitoring, and alerts administrators if any of the configured dump destinations or dump drives are likely to run out of space.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

For this test to work, the eG install user should be in the **Oracle User Group**.

Purpose	Monitors dump space usage
Target of the test	An Oracle database server (9i and 10g)
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. DUMPFILE - By default, this parameter is set to none. This implies that the eG agent auto-discovers the name and the full path to all the dump destinations on the Oracle database server. If required, you can manually specify the path to the dump destination. For eg, /user/john/udump. In this case, multiple paths can also be provided as a comma-separated list - eg.,/user/john/udump, /user/john/bdump
5. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
6. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
7. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
8. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every DUMPFILE that is auto-discovered</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Used Dump area**: Indicates the amount of space in this dump destination that is currently occupied by trace files.	MB	
	Used drive space: Indicates the space in the dump drive that is currently occupied by all files, including trace files.	MB	
	Relative dump area usage: Indicates the percentage of the total used space in the dump drive that is occupied by trace files.	Percent	This measurement value should ideally be below 50%. Any value higher than 50%, indicates that the trace files are consuming more space than the other files in the dump drive. To free some space in that drive, you can adopt any of the following approaches: • Add more disk space to the dump drive; • Take backups of the old trace files to tape or to another destination, and remove them from the dump drive; • Temporarily, you can even zip all trace files in the dump destination. If sufficient space is not made available to the dump destination soon, then trace files can no longer be created in the directory; sometimes, this can cause the Oracle instance to fail.
	Available drive space: Indicates the available free space in the dump drive.	MB	

	Free drive space: Indicates the percentage of space in the dump drive that is currently unused.	Percent	This measurement value should ideally be high. If the value is consistently low, you may want to check the value of the **Relative dump area usage** measure to determine what is causing the space drain - is it because of the trace files, or the other files in the dump destination drive? If the trace files appear to be consuming excessive space in the drive, you can free some space in the drive by adopting any of the following approaches: • Add more disk space to the dump drive; • Take backups of the old trace files to tape or to another destination, and remove them from the destination directory; • Temporarily, you can even zip all trace files in the dump drive.
	Dump area growth rate: Indicates the rate at which dump files are eroding the space in the dump drive.	MB/Sec	Ideally, the value of this measure should be low. A consistent increase in this value is a cause for concern as it indicates that free space in the dump drive is getting eroded at a rapid pace. This in turn hints at a potential space crunch in the directory, which if not averted, could cause the performance of the database server to deteriorate.

2.9.19 Oracle Flash Area Usage Test

The Flash Recovery Area is a specific area of disk storage that is set aside exclusively for retention of backup components such as datafile image copies, archived redo logs, and control file autobackup copies. These features include:

- **Unified Backup Files Storage.** All backup components can be stored in one consolidated spot. The Flash Recovery Area is managed via Oracle Managed Files (OMF), and it can utilize disk resources managed by Oracle Automated Storage Management (ASM). In addition, the Flash Recovery Area can be configured for use by multiple database instances if so desired.
- **Automated Disk-Based Backup and Recovery.** Once the Flash Recovery Area is configured, all backup components (datafile image copies, archived redo logs, and so on) are managed automatically by Oracle.
- **Automatic Deletion of Backup Components.** Once backup components have been successfully created, RMAN can be configured to automatically clean up files that are no longer needed (thus reducing risk of insufficient disk space for backups).
- **Disk Cache for Tape Copies.** Finally, if your disaster recovery plan involves backing up to alternate media, the Flash Recovery Area can act as a disk cache area for those backup components that are eventually copied to tape.
- **Flashback Logs.** The Flash Recovery Area is also used to store and manage flashback logs, which are used during Flashback Backup operations to quickly restore a database to a prior desired state.

Oracle recommends that the Flash Recovery Area should be sized large enough to include all files required for backup and recovery. Using the OraFlashAreaUsage test, administrators can figure out whether the Flash Recovery Area is adequately sized or not, and accordingly make sizing recommendations.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

This test is applicable only to Oracle database server 10g (and above).

Purpose	Monitors the usage of the Flash recovery area
Target of the test	An Oracle database server 10g
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the flash recovery area on the Oracle server</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td></td><td>Flash area usage:
Indicates the space currently occupied by the flash recovery files.</td><td>MB</td><td></td></tr>
</table>

	Maximum flash area size: Indicates the maximum space allocated for flash recovery files.	MB	
	Flash area usage: Indicates the percentage of space occupied by the flash recovery files.	Percent	Oracle recommends that the Flash Recovery Area should be sized large enough to include all files required for backup and recovery. Therefore, ideally, the value of this measure should be very low. A value close to 100% indicates excessive usage of the recovery area; this implies that the flash recovery area could soon run out of space. in such a case you can resize the flash recovery area by reconfiguring the parameter "**db_recovery_file_dest_size**" in database parameter file, provided enough disk space is available. If not, then Oracle recommends that the flash area be sized at least large enough to contain any archived redo logs that have not yet been backed up to alternate media. Alternatively, you can remove the old files from the flash recovery area to create space for the new recovery files.
	Free flash area: Indicates the free space currently available for recovery files.	MB	

	Percent free disk space: Indicates the percentage of disk space that is currently available for use.	Percent	Disk space is a key factor in sizing the flash recovery area. The value to be set for the **db_recovery_file_dest_size** parameter should be decided after carefully considering the total disk space that is available for use. If the disk space is low, then Oracle recommends that the flash area be sized at least large enough to contain any archived redo logs that have not yet been backed up to alternate media. If very little disk space is free, then it would make more sense to free up some space in the disk or in the flash recovery area by removing old, unused files, so that enough space is available for the future files.

2.9.20 Oracle Object Statistics Test

This test is used for finding waits that are associated with a specific Oracle table. The most important of these object-level wait events will give you clues to the source of the contention. The **Oracle Object Statistics** test monitors these waits and reports the number and type of waits.

To perform this test , you have to set the **STATISTICS_LEVEL** parameter in the Init parameter file (in the **<ORA_HOME>\ora<oracle_version>\database** directory by default) either to **Typical** or **ALL**. By default it is **Typical.** Otherwise the Oracle will not be able to collect the Object level statistics.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	To identify the status of jobs.
Target of the test	An Oracle server (9i and 10g)
Agent executing the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password>
container=current default tablespace <name_of_default_tablespace>
temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. OBJECT_NAME – Provide the names of the objects to be monitored in the following format: <username>.<objectname>. For eg., to monitor user john's alarm table, the OBJECT_NAME would be: john.alarm. Multiple username.objectname pairs can be provided as a comma-separated list.
6. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
7. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
8. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every username.objectname configured</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**ITL waits:** Indicates the number of times ITL waits have occurred during the last measurement period.	Number	In oracle, when a row is locked by a transaction, that information is placed in the block header where the row is located. When another transaction wishes to acquire the lock on the same row, it has to travel to the block containing the row anyway, and upon reaching the block, it can easily tell that the row is locked from the block header. There is no need to queue up for some single resource like a lock manager. ITL is the portion of the block header that contains information on locking. It is a simple data structure called **"Interested Transaction List" (ITL)**, a linked list data structure that maintains information on transaction address and rowid. ITL contains several slots or place holders for transactions. When a row in the block is locked for the first time, the transaction places a lock in one of the slots with the rowid of the row that is locked. In other words, the transaction makes it known that it is interested in the row. During the table creation, the **INITRANS** parameter defines how many slots are initially created in the ITL. When the transactions exhaust all the available slots and a new transaction comes in to lock a row, the ITL grows to create another slot. The ITL can grow up to the number defined by the **MAXTRANS** parameter of the table, provided there is space in the block.

			Sometimes, transaction may not be able to find a free slot to place its lock information. This can occur because either (i) the block is so packed that the ITL cannot grow to create a free slot, or (ii) the MAXTRANS has already been reached. To overcome this, do the following: • Reorganizing the table by setting a high value of INITRANS will make sure that there are enough free slots in the ITL, and there will be minimal or no dynamic extension of the ITL. • The other option is to make sure the data is less packed so that ITL can grow enough to accommodate the surges in ITL.

	Buffer busy waits: Indicates the number of times Buffer Busy waits that have occurred during the last measurement period.	Number	Buffer busy waits occur within Oracle when a task goes to fetch a data block, but it must wait because another task has control of the data block in the buffer. A buffer busy wait is often caused by contention on an Oracle table header block because multiple tasks are waiting their turn to grab a freelist to place their new data rows. This *buffer busy wait* condition happens for two reasons: ➢ Another session has the buffer block locked in a mode that is incompatible with the waiting session's request. ➢ The block is being read into the buffer by another session, so the waiting session must wait for the block read to complete. Ideally, value of this measure should be low. If this measure is high, you can reduce the buffer busy wait events by doing the following: ▪ By tuning the SQL to access this object's rows with fewer block reads by adding indexes; ▪ Adding freelists to tables and indexes, or by adding this object to keep cache.
	Row lock waits: Indicates the number of times Row Lock Waits have occurred on this object during the last measurement period.	Number	

	Physical reads: Indicates the number of physical reads that occurred on this object during the last measurement period.	Number	Physical reads/writes – whether direct or not – increase the processing overheads incurred by the database. Therefore, ideally, the value of these measures should be kept at a minimum at all times. In the event of an unusually high number of *Buffer busy waits*, you might want to take a look at the values of these measures, to identify the bottleneck.
	Direct physical reads: Indicates the number of times direct physical reads occurred on this object during the last measurement period.	Number	
	Physical writes: Indicates the number of physical writes that occurred on this object during the last measurement period.	Number	
	Direct physical writes: Indicates the number of times direct physical writes occurred on this object during the last measurement period.	Number	
	Logical reads: Indicates the number of logical reads that occurred on this object during the last measurement period.	Number	

2.9.21 Oracle Object Wait Events Test

This test monitors wait events on objects, and accurately reveals which objects have been waiting for too long a time, and which wait event was active on that object during those times.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors wait events on objects, and accurately reveals which objects have been waiting for too long a time, and which wait event was active on that object during those times
Target of the test	An Oracle 10g database server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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.

	8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every wait event on the Oracle server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Average wait time of objects Indicates the average time this object has waited.	Secs	Ideally, the value of this measure should be low. If this measure shows high value, then by making use of detailed diagnosis you can find the name of the wait event, the total number of waits, the total time waited, and the average wait time per event. With the help of such data you can identify those objects with a high wait time; such objects naturally are resource-intensive. To reduce the resource drain induced by the object, you need to make sure that accesses to the object are low on wait time. This can be ensured by employing the following techniques: ▪ adding indexes to right column; ▪ rebuilding the indexes; ▪ reorganizing the table; ▪ By fine-tuning thequery which accesses the object.
	Number of object wait events : Indicates the number of the object waits for each event.	Number	

2.9.22 Oracle Session Wait Events Test

This test monitors wait events to reveal the events per wait class, and the average wait time experienced by the events of a class. Using the metrics reported by this test, DBAs can isolate the type of wait events that are occurring frequently on the database server, and how long such events last. Since the wait class reveals the source of the frequent wait events, from here, all the administrator needs to do is perform simple sequence of diagnostics to figure out why events of this type recur on the database server.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors wait events to reveal the events per wait class, and the average wait time experienced by the events of a class
Target of the test	An Oracle 10g server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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.

	8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every wait class on the Oracle server monitored		
Measurements made by the	**Measurement**	**Measurement Unit**	**Interpretation**
	Average time waited: Indicates the sum of wait times of all wait events of this wait class during this measurement period.	Secs	If the value of this measure is unusually high, then you can use the detailed diagnosis of the measure to accurately identify that wait even under the class that has contributed to the increase in wait time.
	Number of session wait events: This measure indicates the percentage of waits for each wait class during this measurement period.	Number	

2.9.23 Oracle Sql Wait Events Test

This test monitors wait events generated by SQL queries, reveals the total number of such events and the their total wait time. The test additionally supports a detailed diagnosis capability, which when enabled, helps administrators identify the exact SQL query that has generated a time-consuming wait event. Using this information, administrators can easily fine-tune that query alone and make sure that such wait events do not recur.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors wait events generated by SQL queries, reveals the total number of such events and the their total wait time
Target of the test	An Oracle 10g database server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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.

	8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every wait class generating one/more SQL wait events on the Oracle server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Average time waited by SQL: Indicates the average time the SQL queries waited during this measurement period.	Secs	If this measure registers a high value, then you use the detailed diagnosis of this measure to nail the time-consuming SQL query.
	Number of SQL wait events: Indicates the number of the SQL wait events under this wait class during this measurement period.	Number	

2.9.24 Oracle System Wait Events Test

This test monitors system wait classes for the number and average time of system wait events. Using this test, administrators can nail the wait class on which the Oracle server spends more time and why, so that performance tuning decisions can be taken.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors system wait classes for the number and average time of system wait events
Target of the test	An Oracle 10g server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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.

	8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every system wait class on the Oracle server		
Measurements made by the	**Measurement**	**Measurement Unit**	**Interpretation**
	Average time waited: Indicates the average time in seconds for each wait event in this class, during this measurement period.	Secs	If the value of this measure is unusually high, then you can get the identify time-consuming wait event, using the detailed diagnosis of this measure.
	Number of system wait events: This measure indicates the number of waits for each wait class during this measurement period.	Number	

2.9.25 Oracle Archive Area Test

An Oracle database can run in one of two modes. By default, the database is created in NOARCHIVELOG mode. When in NOARCHIVELOG mode the database runs normally, but there is no capacity to perform any type of point in time recovery operations or online backups. In ARCHIVELOG mode on the other hand, the database will make copies of all online redo logs after they are filled. These copies are called archived redo logs. The archived redo logs are created via the ARCH process. The ARCH process copies the archived redo log files to one or more archive log destination directories.

Note that while the database is being run in the ARCHIVELOG mode, then once an online redo log has been filled, it cannot be reused until it has been archived. If, in the meantime, the destination directory for the archived redo logs runs out of space, then Oracle cannot archive the online redo log. Instead, it will switch to the next online redo log and keep working, while continuing its efforts to archive the log file.

If the database is unable to archive the redo log files for a long time, then at some point it might run out of available online redo logs. Since it cannot reuse the unarchived redo logs for writing the new redo log entries, the database freezes all its operations and stops processing user requests until such time that space is freed in the archive log destination directories.

To ensure that the database is always available to process requests, administrators need to ensure that the archive log destination directories are adequately sized. The OraArchiveArea test periodically monitors the usage of the archive log destination directories, and warns administrators about a sudden/consistent decrease in the free space available in the directories. This enables administrators to act fast and free sufficient space in the directories, so as to prevent the database from suspending its activities.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

For this test to work, the eG install user should be in the **Oracle User Group**.

Purpose	Monitors the archive space usage.
Target of the test	An Oracle server (9i and 10g)
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The host for which the test is to be configured
3. ARCHIVELOGFILE - By default, the eG agent auto-discovers the location of the Oracle archive log file. This is why, the ARCHIVELOGFILE parameter is set to none by default. If required, you can manually specify the path to the Oracle archive log file to be monitored. For eg, /user/john/archive
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the ARCHIVELOGFILE configured/auto-discovered</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Used archive area:** Indicates the space currently occupied by the archive files in the archive destination directory.	MB	
	Used drive space: Indicates the space in the archive destination drive that is currently occupied by all files, including archive files.	MB	
	Relative archive area usage: Indicates the percentage of total used space in the archive destination drive, which is occupied by the archive files.	Percent	This measurement value should ideally be below 50%. Any value higher than 50%, indicates that the archive files are consuming more space than the other files in the archive destination drive. To free some space in that drive, you can adopt any of the following approaches: • Add more disk space to the archive drive; • Take backups of the old archive files to tape or to another destination, and remove them from the destination directory; • Temporarily, you can even zip all archive files in the archive destination.
	Available drive space: Indicates the current free space in the archive destination.	MB	

	Percent drive space free: Indicates the percentage of unused space in the archive destination.	Percent	This measurement value should ideally be high. If the value is consistently low, you may want to check the value of the **Relative archive area usage** measure to determine what is causing the space drain - is it because of the archive files, or the other files in the archive destination drive? If the archive files appear to be consuming excessive space in the drive, you can free some space in the drive by adopting any of the following approaches: ➢ Add more disk space to the archive drive; ➢ Take backups of the old archive files to tape or to another destination, and remove them from the destination directory; ➢ Temporarily, you can even zip all archive files in the archive destination.
	Archive area growth rate: Indicates the rate at which archive files occupied space in the archive destination directory.	MB/Sec	Ideally, the value of this measure should be low. A consistent increase in this value is a cause for concern as it indicates that free space in the archive destination directory is getting eroded at a rapid pace. This in turn hints at a potential space crunch in the directory, which if not averted, could cause the performance of the database server to deteriorate.

2.9.26 Oracle RMAN Job Details Test

The Oracle Recovery Manager (RMAN) provides a comprehensive foundation for efficiently backing up and recovering the Oracle database. It is designed to work intimately with the server, providing block-level corruption detection during backup and restore. It provides a common interface, via command line and Enterprise Manager, for backup tasks across different host operating systems and offers features not available through user-managed methods, such as parallelization of backup/restore data streams, backup files retention policy, and detailed history of all backups. Since errors in backup/recovery jobs can result in loss of critical data, it is essential to keep a close watch on the activities of the RMAN. Using the OraRmanJobTest, you can monitor the status of backup/recovery jobs executed by the RMAN so that, you can be forewarned of issues in these critical processes.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Provides a comprehensive foundation for efficiently backing up and recovering the Oracle database
Target of the test	An Oracle server

Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.

	8. **ELAPSED TIME** - This test reports an **Exceeded time limit jobs** measure, which reveals the number of jobs that have been running beyond a time limit (in minutes) that is configured in the **ELAPSED TIME** text box. For instance, if the **ELAPSED TIME** is set to 5 minutes, then the **Exceeded time limit jobs** measure will report the count of jobs that have been running for over 5 minutes.		
Outputs of the test	One set of results for every Oracle server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Completed jobs: Indicates the number of jobs completed during the last measurement period.	Number	Use the detailed diagnosis of this measure to view the details of the completed jobs.
	Failed jobs: Indicates the count of failed jobs in the last measurement period.	Number	Ideally, the value of this measure should be 0. If a non-zero value is reported, use the detailed diagnosis of this measure to determine which jobs failed at what time.
	Running jobs: Indicates the number of jobs that were running during the last measurement period.	Number	Use the detailed diagnosis of this measure to view the details of the jobs that were running.
	Jobs running with errors: Indicates the number of jobs that were running during the last measurement period, but with errors.	Number	Ideally, this value should be low. If the value is high, you may want to check the detailed diagnosis of this measure to know which jobs are running with errors.
	Jobs running with warnings: Indicates the number of jobs that were running during the last measurement period, but with warnings.	Number	Ideally, this value should be low. If the value is high, you may want to check the detailed diagnosis of this measure to know which jobs are running with warnings.
	Jobs completed with errors: Indicates the number of jobs that were completed during the last measurement period, but with warnings.	Number	Ideally, this value should be low. If the value is high, you may want to check the detailed diagnosis of this measure to know which completed jobs have errors.
	Jobs completed with warnings: Indicates the number of jobs that were completed during the last measurement period, but with errors.	Number	Ideally, this value should be low. If the value is high, you may want to check the detailed diagnosis of this measure to know which completed jobs are with warnings.

	Jobs that exceeded time limits: Indicates the number of jobs that are taking an abnormal amount of time to complete.	Number	If this measure reports a non-zero value, then, it indicates that one/more jobs are taking too long to complete. Since such jobs could drain the server of resources, it is imperative that you determine why the jobs are taking so much time to execute, and fix the problem. A possible reason could be that these jobs are waiting for objects that have been locked by other sessions; if these sessions are less-critical, you may want to terminate them in order to enable the jobs to use the locked resources and resume execution. To know the jobs that are taking too long a time, use the detailed diagnosis of this measure.

2.9.27 Oracle Object Fragmentation Test

Fragmentation of tables and indexes may reduce performance, depending on the way data is accessed. Fragmentation also leads to greater overall storage space usage.

Table fragmentation will result in longer query times when a full table scan is performed. Since data is not as evenly packed in the data blocks, many blocks may have to be read during a scan to satisfy the query. These blocks may be distributed on various extents. In this case, Oracle must issue recursive calls to locate the address of the next extent in the table to scan.

Index fragmentation may bring a higher penalty to application performance. When accessing data through an index and an index range scan, Oracle must read each block in the specified range to retrieve the indexed values. If the index is highly fragmented, Oracle may have to search many more blocks, and possibly levels, to get this information, thus delaying query processing and degrading overall performance.

The first step to resolving the performance threat posed by fragmentation is to identify which objects (tables, indexes, or both) are fragmented. The **Oracle Object Fragmentation** test helps in this regard. This test scans a pre-configured object sample for high and very high levels of fragmentation, and reports the count of fragmented objects. Using the detailed diagnosis capability of the test, you can also quickly drill down to the specific objects that have been fragmented. You can thus proceed to rebuild the fragmented objects to reduce disk I/O.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Scans a pre-configured object sample for high and very high levels of fragmentation, and reports the count of fragmented objects
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed. **As this test, if executed frequently, may increase the processing overheads of the eG agent, It is recommended that you run this test less frequently - say, once a day (24 hrs).** 2. **HOST** – The host for which the test is to be configured 3. **PORT** - The port on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session*, and *analyze any* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant analyze any to oratest* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to **<user_name>**;* *Grant select_catalog_role to **<user_name>**;* *Grant analyze any to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **OBJECT NAME** - Specify a comma-separated list of objects – i.e., tables and/or indexes - that need to be checked for fragmentation. Every object name should be specified in the following format: *<DisplayName>:<schema_name>.<object_name>*, where *schema_name* refers to the name of the object owner, and *object_name* refers to the name of the table/index you want to monitor. The *DisplayName* in your specification will appear as the descriptor of this test. For instance, to monitor the fragmentation-levels of *alarm* and *history* tables owned by user *admin*, your specification would be: *AlarmMon1:admin.alarm,AlarmMon2:admin.history*. To monitor all objects in a schema, the specification would be of the following format: *<DisplayName>:<schema_name>.**. For example, to monitor all the objects in the *admin* schema, your specification would be:

	*AlarmMon:admin.**.

	You can also configure the **OBJECT NAME** to indicate what percentage of records in a table are to be considered by this test for running fragmentation checks. To achieve this, your **OBJECT NAME** specification should be of the following format: *<DisplayName>:<schema_name>.<table_name>@<Percentage_of_records_in_the table>*. For instance, say that you want to configure this test to monitor the fragmentation level of **20%** of the *alarm* table and **30%** of the *history* table. The **OBJECT NAME** specification in this case will be: *AlarmMon:admin.alarm@20,AlarmMon1:admin.history@30*. **It is recommended that you keep this 'percentage value' small, as higher values will make this test that much more resource-intensive**. **Note:** Make sure that you configure the **OBJECT NAME** parameter with only table names and/or index names, and not view names. This is because, tables and indexes alone get fragmented, and not views. 8. **QUERYTIMEOUT** - Specify the time period upto which a query has to wait to obtain the required result set from the database in the **QUERYTIMEOUT** text box. If the query is not successful or if the query waits for a time period exceeding the specified time limit, the test will automatically kill the query. 9. **INCLUDE INDEX** – By default, this test reports metrics on table-level fragmentation only. This is why, the **INCLUDE INDEX** flag is set to **No** by default. If you want the test to report metrics on index-level fragmentation as well, set this flag to **Yes**. 10. **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 enabled/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. 11. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every *DisplayName* configured for the **OBJECT NAME** parameter of this test		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**

	Highly fragmented Oracle objects: Indicates the number of highly fragmented objects of this type.	Number	If 30% - 49% of an object is found to be fragmented, then such an object is counted as a highly fragmented object. Table Fragmentation occurs when we update/delete data in table. The space which gets freed up during non-insert DML operations is not immediately re-used (or sometimes, may not get reused ever). This leaves behind holes in the table, which results in table fragmentation. When rows are not stored contiguously, or if rows are split onto more than one block, performance decreases because these rows require additional block accesses. Index fragmentation is characterized by *splitting* and *spawning*. Splitting happens when an index node becomes full with keys and a new index node is created at the same level as a full node. This widens the B*-tree horizontally. Spawning is the process of adding a new level to an index. As a new index is populated, it begins life as a single-level index. As keys are added, a spawning takes place and the first-level node reconfigures itself to have pointers to lower-level nodes. Both these phenomenon are key performance degraders. This is why, a high value of this measure, if left unchecked, can cause disk I/O to mount, queries to run for long periods, and the overall performance of the database server to deteriorate.

			Use the detailed diagnosis of this measure to identify the highly fragmented objects and the percentage fragmentation of each object, so that you can understand how badly that object is fragmented and can proceed to rebuild it.
	Very highly fragmented Oracle objects: Indicates the number of objects of this type that are very highly fragmented.	Number	If 50% or more of an object is found to be fragmented, then such an object is counted as a very highly fragmented object. Table Fragmentation occurs when we update/delete data in table. The space which gets freed up during non-insert DML operations is not immediately re-used (or sometimes, may not get reused ever). This leaves behind holes in the table, which results in table fragmentation. When rows are not stored contiguously, or if rows are split onto more than one block, performance decreases because these rows require additional block accesses. Fragmentation is characterized by *splitting* and *spawning*. Splitting happens when an index node becomes full with keys and a new index node is created at the same level as a full node. This widens the B*-tree horizontally. Spawning is the process of adding a new level to an index. As a new index is populated, it begins life as a single-level index. As keys are added, a spawning takes place and the first-level node reconfigures itself to have pointers to lower-level nodes. Both these phenomenon are key performance degraders. This is why, a high value of this measure, if left unchecked, can cause disk I/O to mount, queries to run for long periods, and the overall performance of the database server to deteriorate.

			Use the detailed diagnosis of this measure to identify the very highly fragmented objects and the percentage fragmentation of each object, so that you can understand how badly that object is fragmented and can proceed to rebuild it.

2.9.28 Oracle Jobs Test

This test monitors Oracle jobs and reports the number of jobs that have failed and those that are broken. The detailed diagnosis capability offered by this test enables administrators perform further diagnosis on failed/broken jobs, by additionally revealing the complete details of the failed and broken jobs.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors Oracle jobs and reports the number of jobs that have failed and those that are broken
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.

	8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every Oracle server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Failed Oracle jobs: Indicates the number of jobs that failed.	Number	Ideally, the value of this measure should be 0. Any value greater than zero, is a cause of concern, as it indicates the existence of a failed job. To know which job(s) has failed, use the detailed diagnosis capability of this measure. Typically, if a job fails, Oracle attempts to run the job again 16 times, at fixed time intervals. You are advised to investigate the reason for the failure and fix it, by the time Oracle completes its 16th attempt. This is because, if the 16th attempt too fails, Oracle flags the job as a 'broken job', which can then be executed only manually.
	Broken Oracle jobs: Indicates the number of jobs broken.	Number	Ideally, the value of this measure should be 0. Any value greater than 0 is a problem, as it indicates the existence of one/more broken jobs. A job is considered broken, only if the 16th attempt made by Oracle to run the job fails. To know which jobs have broken, use the detailed diagnosis capability of this measure. Once the jobs are identified, you can proceed to manually run the broken jobs through the DBMS_JOB.RUN procedure after logging in as the owner of that job.

2.9.29 Oracle Block Corruption Test

A data block is corrupted when it is not in a recognized Oracle Database format, or its contents are not internally consistent. Block corruptions may affect only a single block or a large portion of the database. It is hence important to rapidly isolate and eliminate corrupted blocks.

Using the **Oracle Block Corruption** test, you can not only identify how many blocks are corrupted in your database, but can also determine which blocks have been corrupted.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose

this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors Oracle jobs and reports the number of jobs that have failed and those that are broken
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.

	8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every database on the Oracle server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**

	Corrupt blocks: Indicates the number of corrupted blocks in this Oracle database.	Number	Ideally, the value of this measure should be zero. A high value indicates that too many blocks in the database are corrupted. If allowed to grow, it could affect the whole database, causing significant loss of data. The first step towards correcting corruption therefore is know where the corruption has occurred and understand the nature of the corruption. Towards this end, eG Enterprise offers the detailed diagnosis capability. By enabling this capability for this test, you can view the details of corrupted blocks such as the exact file number, file name, block number where the corruption starts, number of corrupted blocks, corruption change and the corruption type. Once the corrupted objects are identified, you can use one of the many methods that Oracle provides for rectifying the corruption. One method of correction is to drop and re-create an object after the corruption is detected. However, this is not always possible or desirable. If data block corruption is limited to a subset of rows, another option is to rebuild the table by selecting all data except for the corrupt rows. Yet another way to manage data block corruption is to use the **DBMS_REPAIR** package. You can use **DBMS_REPAIR** to detect and repair corrupt blocks in tables and indexes. Using this approach, you can address corruptions where possible, and also continue to use objects while you attempt to rebuild or repair them.

2.9.30 Oracle Logons Test

User logons serve as good indicators of how applications are using the databases on the Oracle server. Abuse/inefficient

use of the database can severely hamper the performance of the corresponding application.

This test reports the number of user logons that has occurred per second in each Oracle database on a monitored Oracle server.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the number of user logons that has occurred per second in each Oracle database on a monitored Oracle server
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every database on the Oracle server</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

	Logon rate: Indicates the rate at which the users are logged on in this Oracle database.	Number/Sec	A high logon rate may indicate that an application is inefficiently accessing the database. Database logons are costly operations. If an application is performing a logon for every SQL access, that application will experience poor performance as well as affect the performance of other applications on the database. If there is a high logon rate try to identify the application that is performing the logons to determine if it could be redesigned such that session connections could be pooled, reused or shared.

2.9.31 Oracle Data File Errors Test

The most common reasons for data file errors are corrupted blocks and invalid blocks. Both these can cause damage to portions of the database or the whole database, and can thus result in minimal to heavy loss of data. This is why, you should waste no time in identifying the error-prone data files and in doing all that is necessary to clear the errors and salvage the data. The **Oracle Data File Errors** test can play a key role in this exercise.

This test combs all the data files on the target Oracle server for errors and reports the number of errors (if any). The test also provides the complete details of every error, thus enabling a speedy and effective resolution.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Combs all the data files on the target Oracle server for errors and reports the number of errors (if any)
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.

	8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for the Oracle server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Error files count: Indicates the number of data file errors that have occurred.	Number	Ideally the value of this measure should be zero. The detailed diagnosis of this measure indicates the File number, Status, Error, Recover and the Tablespace name for each error that has occurred in the datafiles.

2.9.32 Oracle DB Wait Times Test

Oracle's response time for an operation is composed of **time executing** (=CPU time) and **time spent waiting** (=Waiting time). An increase in either or both the above-mentioned factors will adversely impact the responsiveness of the Oracle database server.

When Oracle executes an SQL statement, it is not constantly executing. Sometimes it has to wait for a specific *event* to happen before it can proceed. For example, if Oracle (or the SQL statement) wants to modify data, and the corresponding database block is not currently in the SGA, Oracle waits for this block to be available for modification. The *Waiting time* refers to the time spent by the Oracle server waiting for such *events* to complete. Oracle has a bunch of events that it can wait for - eg., buffer busy waits, db file scattered read, db file sequential read.

Whenever users complaint of a slowdown of the database server, it would be helpful to know where the database server is spending too much time - is the time executing more than the time spent waiting, or vice-versa? To determine this, you should monitor both the *CPU time* and the *Waiting time* of the database server. This test enables you to perform 'half' this analysis. In other words, this test reports the percentage of time that the Oracle server spent on waiting for one/more events to complete. This way, the test helps you understand whether/not the *waiting time* is contributing to the poor responsiveness of the server.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the percentage of time that the Oracle server spent on waiting for one/more events to complete
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Oracle server being monitored</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

	DB time spent waiting: Indicates the percentage of time the database spent on waiting for one/more events to complete.	Percent	A high value is indicative of the following cases: • An increase in load (either more users, more calls, or larger transactions) • I/O performance degradation (I/O time increases and wait time increases, so DB time increases) • Application performance degradation • CPU-bound host (foregrounds accumulate active run-queue time, wait event times are artificially inflated)

2.9.33 Oracle Defer Transaction Errors Test

Oracle uses deferred transactions to propagate data-level changes asynchronously among master sites in an advanced replication system as well as from an updatable snapshot to its master table. If a transaction is not successfully propagated to the remote site, Oracle rolls back the transaction, and logs the transaction in the SYS.DEFERROR view in the remote destination database.

This test reports the number of transactions that failed to materialize to the destination database from the source database due to errors.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the number of transactions that failed to materialize to the destination database from the source database due to errors
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Oracle server being monitored</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

	Defer transaction error count: Indicates the number of transactions that failed due to errors while being transferred from the source database to the destination database.	Number	An error in applying a deferred transaction may result from a database problem, such as a lack of available space in the table to be updated, or may be the result of an unresolved insert, update, or delete conflict.

2.9.34 Oracle Defer Transactions Test

Oracle uses deferred transactions to propagate data-level changes asynchronously among master sites in an advanced replication system as well as from an updatable snapshot to its master table.

This test reports the number of deferred transactions in this Oracle database.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the number of deferred transactions in this Oracle database
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Oracle server being monitored</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td></td><td>Defer transaction count:
Indicates the number of deferred transactions in the Oracle database.</td><td>Number</td><td></td></tr>
</table>

2.9.35 Oracle Materialized View Intervals Test

A materialized view is a database object that contains the results of a query. They are local copies of data located remotely, or are used to create summary tables based on aggregations of a table's data. Materialized views, which store data based on remote tables, are also known as snapshots. A snapshot can be redefined as a materialized view.

A materialized view in Oracle is a replica of a target master from a single point in time. The master can be either a master table at a master site or a master materialized view at a materialized view site. Whereas in multimaster, replication tables are continuously updated by other master sites, materialized views are updated from one or more masters through individual batch updates, known as refreshes, from a single master site or master materialized view site.

Monitoring the time window between two refreshes will provide you with a fair idea of the volume of changes that materialized views have been updated with. This test reports this refresh interval.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the elapsed time since the last refresh of the materialized view
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Oracle server being monitored</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

	Last refresh: Indicates the duration of time elapsed since the materialized view was last refreshed.	Minutes	When a materialized view is fast refreshed, Oracle must examine all of the changes to the master table or master materialized view since the last refresh to see if any apply to the materialized view. Therefore, if any changes were made to the master since the last refresh, then a materialized view refresh takes some time to apply the changes to the materialized view. If, however, no changes at all were made to the master since the last refresh of a materialized view, then the materialized view refresh should be very quick.

2.9.36 Oracle Listener Test

The listener is a separate process that runs on the database server computer. It receives incoming client connection requests and manages the traffic of these requests to the database server.

If listener logging is enabled, then the listener log file will log audit trail information that will enable you to gather and analyze network usage statistics, as well as information indicating the following:

- A client connection request

  A **RELOAD**, **START**, **STOP**, **STATUS**, or **SERVICES** command issued by the Listener Control utility
  In addition, the log file will log the following:

- Service-registration related events;
- Direct hand-off events;
- Messages indicating the failure of the listener's subscription to the Oracle Notification Service (ONS) **node down** event
- Messages indicating that the listener successfully notified CRS (Cluster Ready Service) that its libraries were installed

One problem that happens in very active databases is that the listener.log file can grow very large. If this growth is not controlled soon, then the directory storing the log file may run out of space, thereby rendering the log file incapable of capturing new messages.

Using this test, you can not only determine whether listener logging is enabled or not, but also continuously track the usage of space by the listener log file, so that you can detect any rapid growth in log file size early and take adequate measures to curb the growth. Also, when clients complaint of inaccessibility of the Oracle database server, you can use this test to determine whether/not the listener process is currently available on the target server. By reporting how long the process has been up and running, the test also intimates you of intermittent breaks in availability of the

listener process.

Purpose	Helps determine whether listener logging is enabled or not, but also continuously track the usage of space by the listener log file, so that you can detect any rapid growth in log file size early and take adequate measures to curb the growth. Also, when clients complaint of inaccessibility of the Oracle database server, you can use this test to determine whether/not the listener process is currently available on the target server. By reporting how long the process has been up and running, the test also intimates you of intermittent breaks in availability of the listener process
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ORACLE HOME – By default, this test auto-discovers the full path to the Oracle installation directory. This is why, the ORACLE HOME parameter is set to none by default.
8. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Oracle server being monitored</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

	Size of the listener log: Indicates the current size of the listener log file.	MB	A high value for this measure or a consistent increase in its value indicates that the listener log is rapidly growing and may end up occupying too much space on the volume. Often people may overlook the growth in the log file size until they stop getting new log messages or the volume/filing system holding the log file is running out of space. Remember that on most 32bit operating systems, there is usually a 2 GB file size limit. On most 64bit OS's the file can grow a lot larger, often 1TB for a single file. Sometimes this can be the cause of using all the available space on a volume. You may notice that even after you delete the file, the space does not come back as being usable. Specifically, on Unix/Linux - If you delete a file that is open for writing, the space doesn't get freed up until the process that is writing the file terminates. Normally if you rename or delete the listener.log file it will create a new file, but it won't free up the space taken by the old file. This is because the file is open for writing the whole time by the listener. If you **can** stop the listener, then rename the file and then restart it, so that the space is freed.

<table>
<tr>
<td></td>
<td>Is listener logging enabled?:
Indicates whether listener logging is enabled or not.</td>
<td></td>
<td>The values that this measure reports and their corresponding numeric values are as follows:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Yes</td><td>100</td></tr><tr><td>No</td><td>0</td></tr></table>
Note:
By default, this measure reports the Measure Values displayed in the table above to indicatus the listener log status. In the graph of the measure however, the status is represented using the corresponding numeric equivalents - i.e., 0 and 100.</td>
</tr>
<tr>
<td></td>
<td>Listener status:
Indicate whether the listener is currently available or not.</td>
<td></td>
<td>The values that this measure reports and their corresponding numeric values are as follows:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Up</td><td>100</td></tr><tr><td>Down</td><td>0</td></tr></table>
Note:
By default, this measure reports the Measure Values displayed in the table above to indicatus the listener status. In the graph of the measure however, the status is represented using the corresponding numeric equivalents - i.e., 0 and 100.</td>
</tr>
<tr>
<td></td>
<td>Uptime of the listener:
Indicates how long since the last measurement period, the listener has been up and running.</td>
<td>Minutes</td>
<td>A high value is typically desired for the measure. A low value indicates that the listener process rebooted recently.</td>
</tr>
</table>

2.9.37 Oracle ASM Disk I/O Test

ASM is a volume manager and a file system for Oracle database files that supports single-instance Oracle Database and Oracle Real Application Cluster (Oracle RAC) configuration. ASM is Oracle's recommended storage management solution that provides an alternative to conventional volume managers, file systems, and raw devices.

ASM uses disk groups to store datafiles; an ASM disk group is a collection of disks that ASM manages as a unit. Within a disk group, ASM exposes a file system interface for Oracle database files. The content of files that are stored in a disk group are evenly distributed, or striped, to eliminate hot spots and to provide uniform performance across the disks.

You need to periodically monitor the read-write activity on each disk in a disk group to make sure that I/O load is uniformly balanced across all disks in a group. The **ASM Disk I/O** test helps you do just that. At pre-configured intervals, this test monitors the I/O activity on each disk in a disk group, reveals I/O-intensive and error-prone disks, and brings irregularities in load balancing to the fore.

To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors the I/O activity on each disk in a disk group, reveals I/O-intensive and error-prone disks, and brings irregularities in load balancing to the fore
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each DiskGroup:Disk pair on the Oracle server being monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Reads:
Indicates the rate at which reads occur on this disk.</td><td>Reads/Sec</td><td>Compare the values of each of these measures across the disks in a disk group to identify the I/O-intensive</td></tr>
</table>

	Writes: Indicates the rate at which writes occur on this disk.	Writes/Sec	disks in that group. In the process, you can also determine whether/not I/O load is equally balanced across all the disks in the group. If any irregularities are noticed in load-balancing are noticed, you may want to consider adding more disks to the group.
	Read errors: Indicates the number of errors that occur per second while reading from this disk.	ReadErrors/Sec	The value 0 is desired for both these measures. A non-zero value is indicative of I/O errors. By comparing the values of each of these measures across disks and across disk groups, you can not only point to the error-prone disks and groups, but can also figure out when most of the errors occurred on the disk/group - when reading? or when writing?
	Write errors: Indicates the number of errors that occur per second when writing to this disk.	WriteErrors/Sec	

2.9.38 Oracle ASM Disk Space Test

ASM is a volume manager and a file system for Oracle database files that supports single-instance Oracle Database and Oracle Real Application Cluster (Oracle RAC) configuration. ASM is Oracle's recommended storage management solution that provides an alternative to conventional volume managers, file systems, and raw devices.

ASM uses disk groups to store datafiles; an ASM disk group is a collection of disks that ASM manages as a unit. Within a disk group, ASM exposes a file system interface for Oracle database files. The content of files that are stored in a disk group are evenly distributed, or striped, to eliminate hot spots and to provide uniform performance across the disks.

To ensure that a disk group always has sufficient space to store the critical organizational data, you will have to continuously track the space usage of the disk group. This will provide you with early pointers to potential space contentions and help you swiftly provide more space to the group by adding more disks. The **ASM Disk Space** test enables you to achieve this end. This test closely monitors how each disk in a disk group uses the space available to it, points you to the disks that are running out of space, and thus holds a mirror to space contentions on a disk group.

To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Closely monitors how each disk in a disk group uses the space available to it, points you to the disks that are running out of space, and thus holds a mirror to space contentions on a disk group
Target of the test	An Oracle server

<table>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. ORACLEHOME - By default, this test auto-discovers the full path to the Oracle installation directory. This is why, the ORACLE HOME parameter is set to none by default.
5. USER - Specify the name of the user with rights to access the ASM instance being monitored. This user should also have SELECT privileges to v$asm_disks.
6. PASSWORD - Provide the password of the USER.
7. CONFIRM PASSWORD - Confirm the password by retyping it in this text box.
8. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each DiskGroup:Disk pair on the Oracle server being monitored</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td></td><td>Used space:
Indicates the amount of space currently used in this disk.</td><td>MB</td><td>Ideally, the value of this measure should be low. A consistent increase in this value is a cause for concern.</td></tr>
<tr><td></td><td>Free space:
Indicates the amount of space in this disk that is currently free - i.e., available for use.</td><td>MB</td><td>Ideally, the value of this measure should be high. A consistent decrease in this value is a cause for concern.</td></tr>
<tr><td></td><td>Space availability:
Indicates the percentage of space in this disk that is currently unused.</td><td>Percent</td><td>A high value is typically desired for this measure. By comparing the value of this measure across disks and across disk groups, you can quickly isolate the disks/groups that are running short of space. If the free space is alarmingly low for all disks in a group, it indicates that the group requires more space. You can then consider making space by adding more disks to the group.</td></tr>
</table>

	Space usage: Indicates the percentage of space in this disk that is currently used.	Percent	A low value is typically desired for this measure. By comparing the value of this measure across disks and across disk groups, you can quickly isolate the disks/groups that are utilizing space excessively. If the used space is alarmingly high for all disks in a group, it indicates that the group is rapidly running out of space. You can then consider making space by adding more disks to the group.
	Used space growth: Indicates the growth in space usage of this disk since the last measurement period.	MB/Sec	If you observe the variations to this measure over time, you will be able to detect early whether the space in the disk is being steadily eroded or not. This way, you can initiate measures to conserve space much before the disk exhausts all the space available to it.

2.9.39 Oracle User Expiry Details Test

A user account in Oracle expires when the **PASSWORD_LIFE_TIME** configured for that user is reached. Upon expiry, Oracle automatically locks the user account, preventing that user from logging in. To avoid this, users need to be proactively alerted to the impending expiry of their accounts, thereby giving them time to reset the **PASSWORD_LIFE_TIME** configuration, if required. The **Oracle Users Expiry Details** test does just that. This test not only reports the number of expired users on an Oracle database server, but also reports the number of user accounts that are nearing expiry.

To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Oracle Database* as the **Component type**, *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the number of expired users on an Oracle database server and also reports the number of user accounts that are nearing expiry
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user <user_name> identified by <user_password> container=current default tablespace <name_of_default_tablespace> temporary tablespace <name_of_temporary_tablespace>;
Grant create session to <user_name>;
Grant select_catalog_role to <user_name>;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. NO OF DAYS - The Users nearing expiry measure of this test reports the number of user accounts that will be expiring within the duration (in days) specified here.
8. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Oracle database server being monitored</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

	Expired users: Indicates the current number of expired users.	Number	Use the detailed diagnosis of this measure to know which user accounts have expired.
	Users nearing expiry: Indicates the number of users who will be expiring within the configured **NO OF DAYS**.	Number	Use the detailed diagnosis of this measure to know which users are about to expire

2.9.40 Oracle Session Resource Usage Test

In the database context, the connection between the user process and the server process is called a session. The server process communicates with the connected user process and performs tasks on behalf of the users.

This test tracks the resource usage of each session to the Oracle server and thus sheds light on the resource-intensive Oracle sessions. You can also use the detailed diagnostics provided by the test to understand why a user session is consuming CPU/memory resources excessively and then try to attack the problem source.

This test does not apply to Windows platforms.

Purpose	Reports the number of expired users on an Oracle database server and also reports the number of user accounts that are nearing expiry
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.

	8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for the Oracle database server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Maximum CPU used by a session: Indicates the percentage of CPU utilization that is the highest among all the sessions of this Oracle server.	Percent	A high value indicates that one/more Oracle sessions are consuming CPU excessively. Use the detailed diagnosis of the *CPU used by Oracle sessions* measure to know which session is consuming maximum CPU. Using this session information, you can optimize the query executed by that session to minimize CPU usage.
	CPU used by Oracle sessions: Indicates the percentage of CPU that is currently utilized by all sessions of this Oracle server.	Percent	
	Maximum memory used by a session: Indicates the percentage of memory utilization that is the highest among all the sessions of this Oracle server.	Percent	A high value indicates that one/more Oracle sessions are consuming memory excessively. Use the detailed diagnosis of the *Memory used by Oracle sessions* measure to view which session is consuming maximum memory. Using this session information, you can optimize the query executed by that session to reduce memory usage.
	Memory used by Oracle sessions: Indicates the percentage of memory that is currently utilized by all sessions of this Oracle server.	Percent	

2.9.41 Oracle Wait Class Test

When Oracle executes an SQL statement, it is not constantly executing. Sometimes it has to wait for a specific event to happen before it can proceed. For example, if Oracle (or the SQL statement) wants to modify data, and the

corresponding database block is not currently in the SGA, Oracle waits for this block to be available for modification. Every such wait event belongs to a class of wait events. The following list describes each of the wait classes.

Wait Class	Description
Administrative	Waits resulting from DBA commands that cause users to wait (for example, an index rebuild)
Application	Waits resulting from user application code (for example, lock waits caused by row level locking or explicit lock commands)
Cluster	Waits related to Real Application Cluster resources (for example, global cache resources such as 'gc cr block busy'
Commit	This wait class only comprises one wait event - wait for redo log write confirmation after a commit (that is, 'log file sync')
Concurrency	Waits for internal database resources (for example, latches)
Configuration	Waits caused by inadequate configuration of database or instance resources (for example, undersized log file sizes, shared pool size)
Idle	Waits that signify the session is inactive, waiting for work (for example, 'SQL*Net message from client')
Network	Waits related to network messaging (for example, 'SQL*Net more data to dblink')
Other	Waits which should not typically occur on a system (for example, 'wait for EMON to spawn')
Scheduler	Resource Manager related waits (for example, 'resmgr: become active')
System I/O	Waits for background process IO (for example, DBWR wait for 'db file parallel write')
User I/O	Waits for user IO (for example 'db file sequential read')

Since wait events are resource-drains and serious performance degraders, administrators need to keep a close eye on these wait classes, figure out how much time the Oracle database server actually spends waiting for each class, and rapidly decipher why, so that measures can be initiated to minimize these events. To achieve this, you can use the **Oracle Wait Class** test. This test reports the time spent by the Oracle server waiting for events of each wait class, helps identify those wait classes with wait events that have remained active for a long time, and also reveals the number of sessions that have been impacted by the waiting. With the help of the detailed diagnostics of this test, you can also zoom into these sessions and identify the queries that they executed that may have caused wait events to occur; this way, inefficient queries can be isolated.

Purpose	Reports the time spent by the Oracle server waiting for events of each wait class, helps identify those wait classes with wait events that have remained active for a long time, and also reveals the number of sessions that have been impacted by the waiting
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: o The eG manager license should allow the detailed diagnosis capability o Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

	8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for each wait class active on the Oracle database server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Active sessions: Indicates the current number of sessions in which events of this wait class are currently active.	Number	A high value indicates that too many sessions are waiting owing to the events of a particular wait class. To know more about these sessions, the wait events that each session triggered, and which query triggered the events, use the detailed diagnosis of this measure. With the help of the detailed metrics, you can quickly isolate the queries that require optimization.

	Max wait time: Indicates the maximum time for which the Oracle server has waited for events of this wait class.	Secs	A high value is indicative of the following: • An increase in load (either more users, more calls, or larger transactions) • I/O performance degradation (I/O time increases and wait time increases, so DB time increases) • Application performance degradation • CPU-bound host (foregrounds accumulate active run-queue time, wait event times are artificially inflated) Compare the value of this measure across wait classes to identify which wait class has caused the Oracle database server to wait for the maximum time. You can then use the detailed diagnostics reported by the *Active sessions* measure to identify which sessions were impacted, and what queries were executed by those sessions to increase wait time. Inefficient queries can thus be identified and optimized to ensure that waiting is eliminated or at least minimized.

2.9.42 Oracle Login Sessions Test

Database administrators should eye sessions that have been open for a long time suspiciously, as such sessions are often indicato of performance bottlenecks. By zooming into such sessions, administrators can identify inefficient queries, hung/unresponsive transactions, or session logout failures that may be causing the sessions to remain open for abnormal time periods. This investigation may also bring inactive sessions to light. Inactive sessions unnecessarily hold on to critical server resources, causing business-critical transactions to fail for want of resources! To quickly isolate such problem sessions and the users who initiated them, and to rapidly determine the reason for the problems, administrators can use the **Oracle Login Sessions** test. This test tracks user logins to the database server, identifies users who have sessions open for over a configured duration, and reports the count of such sessions per user. Using the detailed diagnosis of this test, you can also figure out the status of each session. This way, administrators will not only be able to determine the number of sessions that are 'suspect', but can also drill down to the reason why the sessions have been open for an unreasonable period of time. In addition, by reporting session status, the test also leads administrators to inactive sessions that are needlessly draining critical server resources.

Purpose	Tracks user logins to the database server, identifies users who have sessions open for over a configured duration, and reports the count of such sessions per user
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **INCLUDE USER** - By default, this is set to *none*. This indicates that by default, the test monitors all users who are currently logged into the database server. If required, you can provide a comma-separated list of users who are to be monitored. In this case, the test will report the open session count for each user in this comma-separated list only. 8. **EXCLUDE USER**- By default, this is set to *none*. This indicates that by default, the test does not exclude any user from the purview of monitoring. If required, you can provide a comma-separated list of users who are to be excluded from monitoring. In this case, the test will not report the open session count for the excluded users, though they may be currently logged in. 9. **USER LOGIN TIME** - By default, the *Number of sessions* measure reported by this test includes only those sessions that have been open for over 5 minutes. Accordingly, the **USER LOGIN TIME** is set to *5* (minutes) by default. You can override this default setting by changing the duration (in minutes) specification against **USER LOGIN TIME**.

<table>
<tr><td></td><td colspan="3">10. 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 enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:
o The eG manager license should allow the detailed diagnosis capability
o Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.
11. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each user with one/more sessions that have been open for over the configured USER LOGIN TIME</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

	Number of sessions: Indicates the number of sessions for this user that have been open for a duration beyond the configured **USER LOGIN TIME**.	Number	A high value indicates that the user has too many sessions open for an abnormal period of time. By comparing the value of this measure across users, you can quickly identify the user who has the maximum number of such sessions. To know what is causing the sessions to be open for such broad time windows, use the detailed diagnosis of this test. The detailed diagnosis reveals the session start time, the machine from which the session was initiated, the program/query executed in the session, and the session status. From this information, administrators can figure out whether a long-running query / inefficient query is causing the session to remain open for a long time. Such queries can be terminated to close the. Also, by looking at the session status in the detailed diagnosis, administrators can ascertain whether/not the session is active. Once a session is identified as inactive, administrators can proceed to terminate the session, to release critical server resources.

2.9.43 Oracle Timed Workload Test

Workload analysis for an Oracle database server involves:

- Determining the number of transactions that applications execute on the database server at any given point in time;
- Understanding the type of database operations these transactions trigger – executes? Updates? reads? Writes? Rollbacks? Parses?
- Knowing how many users are active on the database server at a given point in time;
- Determining how quickly the server processes this load and how much processing power was spent on the same.

This not only reveals the current workload of the database server, but also highlights the processing ability of the server, pinpoints bottlenecks in processing, and leads administrators to where these bottlenecks lie. To perform such detailed workload analysis, administrators can use the **Oracle Timed Workload** test. This test reports the current CPU usage of the server to indicate its current load. In addition, the test reveals the number and type of transactions the server processes every second, so that administrators can understand how well the server handles the load and can

accurately identify where bottlenecks lie. By comparing the CPU usage of the server with its processing ability, administrators can intelligently figure out if the server requires additional CPU resources for improved performance.

Purpose	Reports the current CPU usage of the server to indicate its current load. In addition, the test reveals the number and type of transactions the server processes every second, so that administrators can understand how well the server handles the load and can accurately identify where bottlenecks lie. By comparing the CPU usage of the server with its processing ability, administrators can intelligently figure out if the server requires additional CPU resources for improved performance
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user oraeg identified by oraeg container=current default tablespace default temporary tablespace temp;
Grant connect, resource to oraeg;
Grant select_catalog_role to oraeg;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Database CPU usage:** Indicates the percentage CPU used by the server.	Percent	A value close to 100% is indicative of excessive CPU usage. This in turn indicates that the server is using up all its processing power to service its current workload. It could be because the load is very high. It could also be owing to a few resource-intensive transactions executing on the server. In case of the former, you may want to allocate more CPU resources to the server, so as to enhance its processing ability.
	CPU time: Indicates the time for which the server has been hogging the CPU resources since the last measurement period.	Secs	A consistent increase in the value of this measure could indicate a steady increase in the workload of the server.
	Redo size: Indicates the rate at which modifications were written to the redo logs since the last measurement period.	MB/Sec	If the value of this measure keeps growing, it could indicate that data is changing rapidly in the databases. A steady drop in this value could indicate that changes are not written to the redo logs as quickly as they occur.
	Logical reads: Indicates the rate at which logical reads were performed by the server.	Reads/Sec	These measures are good indicators of the level of activity on the database server and how well the server handles these activity levels. In the event of a slowdown, you can compare the value of these measures to know where the slowdown may have originated – when making changes to data? When reading? When writing?
	Block changes: Indicates the rate at which database blocks were changed.	Blocks/Sec	
	Physical reads: Indicates the rate at which the server performed physical reads.	Reads/Sec	
	Physical writes: Indicates the rate at which the server performed physical writes.	Writes/Sec	
	User calls: Indicates the rate at which the server made user calls.	Calls/Sec	

	Parses: Indicates the rate at which the server parsed SQL statements.	Parses/Sec	Parsing is one stage in the processing of a SQL statement. When an application issues a SQL statement, the application makes a parse call to Oracle Database. During the parse call, Oracle Database: • Checks the statement for syntactic and semantic validity. • Determines whether the process issuing the statement has privileges to run it. • Allocates a private SQL area for the statement. If the value of this measure keeps increasing consistently, it could indicate that many SQL statements are being executed on the server, thus generating more parses every second. If the value of this measure drops consistently, it could indicate a bottleneck in parsing.
	Hard parses: Indicates the rate at which the server hard parsed SQL statements.	Parses/Sec	As opposed to a soft parse, a hard parse loads the SQL source code into RAM for parsing. If the value of this measure is decreasing steadily, it could mean that hard parsing is taking too long. It could also mean that very few hard parses are actually performed.

	WA memory processed: Indicates the rate at which work area memory is used Lby the server.	MB/Sec	Oracle Database reads and writes information in the PGA on behalf of the server process. An example of such information is the run-time area of a cursor. Each time a cursor is executed, a new run-time area is created for that cursor in the PGA memory region of the server process executing that cursor. For complex queries (such as decision support queries), a big portion of the run-time area is dedicated to **work areas** allocated by memory intensive operators, including: • Sort-based operators, such as `ORDER BY`, `GROUP BY`, `ROLLUP`, and window functions • Hash-join • Bitmap merge • Bitmap create • Write buffers used by bulk load operations For example, a sort operator uses a work area (sometimes called the sort area) to perform the in-memory sort of a set of rows. Similarly, a hash-join operator uses a work area (also called the hash area) to build a hash table from its left input. If the amount of data to be processed by these two operators does not fit into a work area, then the input data is divided into smaller pieces. This allows some data pieces to be processed in memory while the rest are spilled to temporary disk storage to be processed later. A consistent increase in the value of this measure is indicative of excessive usage of the work area. This could indicate that the workload is characterized by complex queries that use memory intensive operators such as sort, hash-join, etc. You may want to fine-tune the work area size in order to enable it to handle the memory-intensive load better.
	Logons: Indicates the rate at which users login to the database server.	Logons/Sec	A steady rise in this value is indicative of a steady increase in user activity on the server.
	Executes: Indicates the rate at which executions are performed by the server.	Executions/Sec	

	Rollbacks: Indicates the rate at which the server performs rollbacks.	Rollbacks/Sec	Ideally, the value of this measure should be low. This is because, rollbacks are expensive operations and should be avoided at all costs. A consistent increase in the value of this measure is hence a cause for concern.
	Transactions: Indicates the rate at which transactions were executed by the server.	Trans/Sec	A steady increase in the value of this measure could indicate an increase in the transaction load on the server. A consistent and notable drop in the value of this measure could indicate a bottleneck in transaction processing.

2.9.44 Oracle Transaction Workload Test

Knowing the count of transactions executing on the Oracle database server per second can indicate the transaction load on the server. However, the true impact of this load can be assessed and understood only if administrators are enabled to determine the number and type of database operations each transaction triggers. This is where the **Oracle Transaction Workload** test helps! This test reports how many key database operations – eg., data modifications, block changes, reads/writes, parses, rollbacks, etc. – are performed on the server per transaction. This way, the test reveals the real workload of the server. In addition, the test also enables administrators to compare current CPU usage with the real workload, so that they can figure out whether/not the server needs to be resized to handle its load.

Purpose	Reports how many key database operations – eg., data modifications, block changes, reads/writes, parses, rollbacks, etc. – are performed on the server per transaction. This way, the test reveals the real workload of the server
Target of the test	An Oracle server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">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 on which the server is listening
4. USER – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A Click here hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the select_catalog_role and create session privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:
create user oraeg identified by oraeg
create role oratest;
grant create session to oratest;
grant select_catalog_role to oratest;
grant oratest to oraeg;
The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::
alter session set container=<Oracle_service_name>;
create user oraeg identified by oraeg container=current default tablespace default temporary tablespace temp;
Grant connect, resource to oraeg;
Grant select_catalog_role to oraeg;
The name of this user has to be specified here.
5. PASSWORD – Password of the specified database user
This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID monitored.</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>CPU time:
Indicates the time for which the server has been hogging the CPU resources since the last measurement period.</td><td>Secs</td><td>A consistent increase in the value of this measure could indicate a steady increase in the workload of the server.</td></tr>
</table>

	Redo size: Indicates the amount of data written to the redo logs per transaction since the last measurement period.	MB/Trans	If the value of this measure keeps growing, it could indicate that transactions are making numerous and frequent changes to the data in the databases.
	Logical reads: Indicates the number of logical reads performed by the server per transaction.	Reads/Trans	These measures are good indicators of the level of activity that every transaction generated on the database server.
	Block changes: Indicates the number of database blocks that were changed per transaction.	Blocks/Trans	
	Physical reads: Indicates the number of physical reads performed per transaction.	Reads/Trans	
	Physical writes: Indicates the number of physical writes performed per transaction.	Writes/Trans	
	User calls: Indicates the number of user calls made per transaction.	Calls/Trans	
	Parses: Indicates the number of parses executed by the server per transaction.	Parses/Trans	Parsing is one stage in the processing of a SQL statement. When an application issues a SQL statement, the application makes a parse call to Oracle Database. During the parse call, Oracle Database: • Checks the statement for syntactic and semantic validity. • Determines whether the process issuing the statement has privileges to run it. • Allocates a private SQL area for the statement. If the value of this measure keeps increasing consistently, it could indicate on an average, transactions are executing many SQL statements on the server, thus generating more parses.

	Hard parses: Indicates the number of hard parses executed per transaction.	Parses/Trans	As opposed to a soft parse, a hard parse loads the SQL source code into RAM for parsing. A high value for this measure therefore indicates that the server is performing many hard parses.
	WA memory processed: Indicates the amount of work area memory used up by the server per transaction.	MB/Trans	Oracle Database reads and writes information in the PGA on behalf of the server process. An example of such information is the run-time area of a cursor. Each time a cursor is executed, a new run-time area is created for that cursor in the PGA memory region of the server process executing that cursor. For complex queries (such as decision support queries), a big portion of the run-time area is dedicated to **work areas** allocated by memory intensive operators, including: • Sort-based operators, such as `ORDER BY`, `GROUP BY`, `ROLLUP`, and window functions • Hash-join • Bitmap merge • Bitmap create • Write buffers used by bulk load operations For example, a sort operator uses a work area (sometimes called the sort area) to perform the in-memory sort of a set of rows. Similarly, a hash-join operator uses a work area (also called the hash area) to build a hash table from its left input. If the amount of data to be processed by these two operators does not fit into a work area, then the input data is divided into smaller pieces. This allows some data pieces to be processed in memory while the rest are spilled to temporary disk storage to be processed later. A consistent increase in the value of this measure is indicative of excessive usage of the work area by transactions. This could indicate that the transaction workload is characterized by complex queries that use memory intensive operators such as sort, hash-join, etc. You may want to fine-tune the work area size in order to enable it to handle the memory-intensive load better.
	Logons: Indicates the number of users logging in per transaction.	Logons/Trans	A steady rise in this value is indicative of a steady increase in user activity on the server.

	Executes: Indicates the number of executes performed per transaction.	Executions/Trans	
	Rollbacks: Indicates the number of rollbacks performed per transaction.	Rollbacks/Trans	Ideally, the value of this measure should be low. This is because, rollbacks are expensive operations and should be avoided at all costs. A consistent increase in the value of this measure is hence a cause for concern.
	Transactions: Indicates the rate at which transactions were executed by the server.	Trans/Sec	A steady increase in the value of this measure could indicate an increase in the transaction load on the server. A consistent and notable drop in the value of this measure could indicate a bottleneck in transaction processing.

2.9.45 Oracle SQL Workload Test

Nothing can degrade the performance of an Oracle database server like a resource-hungry or a long-running query! When such queries execute on the server, they either hog almost all the available CPU, memory, and disk resources or keep the resources locked for long time periods, thus leaving little to no resources for carrying out other critical database operations. This can significantly slowdown the database server and adversely impact user experience with the server. To ensure peak performance of the Oracle database server at all times, such queries should be rapidly identified and quickly optimized to minimize resource usage. This is where the **Oracle SQL Workload** test helps. At configured intervals, this test compares the usage levels and execution times of all queries that started running on the server in the last measurement period and identifies a 'top query' in each of the following categories - CPU usage, memory usage, disk activity, and execution time. The test then reports the resource usage and execution time of the top queries and promptly alerts administrators if any query consumes more resources or takes more time to execute than it should. In such a scenario, administrators can use the detailed diagnosis of this test to view the inefficient queries and proceed to optimize them to enhance server performance.

Purpose	At configured intervals, this test compares the usage levels and execution times of all queries that started running on the server in the last measurement period and identifies a 'top query' in each of the following categories - CPU usage, memory usage, disk activity, and execution time. The test then reports the resource usage and execution time of the top queries and promptly alerts administrators if any query consumes more resources or takes more time to execute than it should. In such a scenario, administrators can use the detailed diagnosis of this test to view the inefficient queries and proceed to optimize them to enhance server performance.
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 5. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DDCOUNT** – By default, the detailed diagnosis of this test reports the top-5 queries in resource usage and execution time. This is why, the **DDCOUNT** parameter is set to *5* by default. If you want detailed diagnosis to display less or more number of top queries, then change the **DDCOUNT**. 8. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 9. **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.		
Outputs of the test	One set of results for every SID monitored.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Top most query physical reads: Indicates the number of physical disk reads performed by the top query per execution.	Reads/execution	If the value of this measure is abnormally high, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries generating maximum physical disk activity. From this, you can identify the top query in terms of number of physical disk reads. You may then want to optimize the query to reduce the disk reads.
	Top most buffer gets: Indicates the number of memory buffers used by the top query per execution.	Memorybuffer gets/execution	If the value of this measure is abnormally high, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries consuming memory excessively. From this, you can easily pick that query which is consuming the maximum memory. You may then want to optimize the query to minimize memory usage.
	Top most query CPU time: Indicates the duration for which each execution of the top query was hogging the CPU resources.	Secs/execution	If the value of this measure is over 30 seconds, you can use the detailed diagnosis of this measure to the top-5 (by default) queries hogging the CPU resources. From this, you can easily pick that query which is consuming the maximum CPU. You may then want to optimize the query to minimize CPU usage.
	Top most query elapsed time: Indicates the running time of each execution of the top query.	Secs/execution	If the value of this measure crosses 10 seconds, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries that are taking too long to execute. . From this, you can easily pick that query with the maximum execution time. You may then want to optimize the query to minimize execution time.

3

Monitoring MS SQL Servers

Microsoft's SQL server has emerged as the database engine of choice for most applications hosted on the Microsoft Windows platform. Services in various domains – healthcare, manufacturing, banking, etc. – rely on the backend database servers for data storage and access. Any performance degradation or unavailability of the database servers can severely impact the performance of the entire service, often causing customer dissatisfaction and lost business revenue. Continuous monitoring of the MS SQL servers are hence imperative.

The pre-built MS SQL monitoring model that eG Enterprise offers (see Figure 3.1), provides in-depth monitoring for SQL database servers.

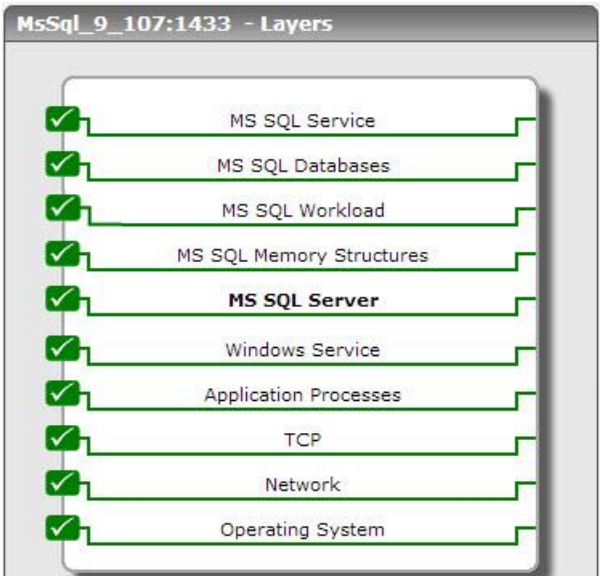


Figure 3.1: Layer model for MS SQL servers

Each of the layers of this hierarchical model reports a wide variety of metrics ranging from the basic operating system-level statistics to individual database related measurements to those indicating the database engine health. The table below sheds light on what the eG SQL Monitor reveals:

Database Service Monitoring	▪ Is the database server available for servicing requests? ▪ What is the response time for a typical query? ▪ How many logins/logouts are happening on the SQL server? ▪ Which applications/users are accessing the SQL server and what is their respective resource usage? ▪ What queries are each of the applications currently executing?

Database Server Engine Monitoring	▪ What is the CPU utilization of the database server engine? ▪ How much time is the SQL server spending on processing vs. I/O? ▪ What is the typical workload on the database server? ▪ Which databases are imposing most load on the database server engine? ▪ How many processes are running, and what queries are they executing? ▪ Which user(s) are executing these queries?
Lock Activity Monitoring	▪ What is the typical locking activity on the database? ▪ Which processes are being blocked and by whom? ▪ Which are the root-blocker processes, and what queries are they executing? ▪ Are any deadlocks happening?
Database Activity and Space Monitoring	▪ What databases are hosted on the SQL server? ▪ Is any of the databases reaching capacity? ▪ Which of the databases is seeing more transaction activity? ▪ How many active transactions are currently happening to each of the database server?
SQL Memory Monitoring	▪ Is there sufficient memory available for the SQL server? ▪ How much memory is the server consuming and how much is it willing to consume? ▪ How much memory is used for connections, how much for locks, and how much for query optimizations? ▪ What is the server's cache hit ratio? ▪ How many pages are available in the server's buffer pool? ▪ How many of these are free pages?
Operating System Monitoring	▪ Is there sufficient disk capacity? ▪ Is there excessive contention for CPU or memory resources? ▪ Are the disks unusually busy? ▪ Which processes are taking up most resources (CPU, memory, disk, etc.)?

The **Operating System**, **Network**, **Tcp**, **Application Processes**, and **Windows Service** layers of the layer model in Figure 3.1 have been discussed in the the *Monitoring Unix and Windows Servers* document. Above the **Windows Service** layer is the **MS SQL Server** layer.

3.1 The MS SQL Server Layer

The tests associated with this layer, indicate the following:

- Error frequency
- SQL engine performance
- SQL Uptime

- Buffer pool usage (specific to MS SQL 2005 and above)
- Active transactions and their effect on **tempdb** (specific to MS SQL 2005 and above)
- Blocked processes and how to deal with them
- The number and type of system processes currently running

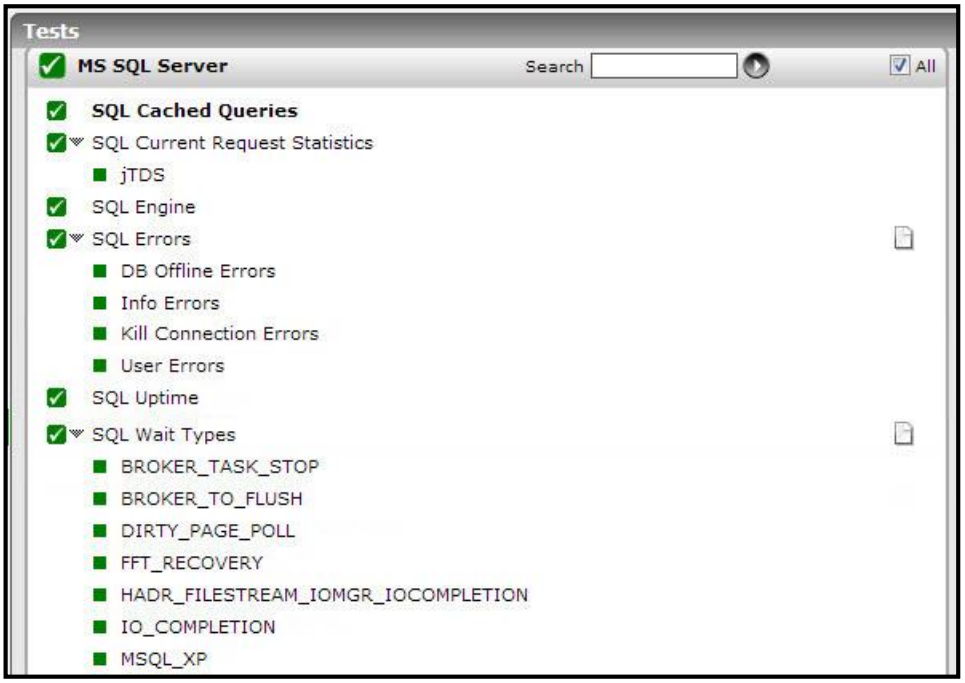


Figure 3.2: The tests associated with the MS SQL Server layer

3.1.1 SQL Wait Types Test

In SQL Server, wait types represent the discrete steps in query processing, where a query waits for resources as the instance completes the request. By analysing wait types and their wait times, administrators can receive quick and objective evidence of performance bottlenecks and their probable causes. The **SQL Wait Types** test enables this analysis. For every type of wait that is currently experienced by the server, this test reports the number, nature, and duration of waits, thereby leading you to the specific wait types that may have contributed to a general slowdown/deterioration in server performance.

Purpose	Reports statistics related to the MS SQL server engine
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. SSL – If the MS SQL server being monitored is an SSL-enabled server, then set the SSL flag to Yes. If not, then set the SSL flag to No.
5. INSTANCE - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the INSTANCE parameter.
6. USER – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the Sysadmin role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE, and PUBLIC roles in this text box.
7. PASSWORD - The password of the specified USER
8. CONFIRM PASSWORD - Confirm the password by retyping it.
9. DOMAIN - By default, none is displayed in the DOMAIN text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the DOMAIN can continue to be none. On the other hand, if 'Windows only' authentication has been enabled, then, in the DOMAIN text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the USER name and PASSWORD that you provide should be that of a user authorized to access the monitored SQL server.
10. ISNTLMV2 - In some Windows networks, NTLM (NT LAN Manager) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the ISNTLMV2 flag is set to No, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to Yes if NTLMv2 is enabled on the target host.
11. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every type of wait in the MS SQL server monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Waiting tasks count:
Indicates the number of waits of this type during the last measurement period.</td><td>Number</td><td>This counter is incremented at the start of each wait.</td></tr>
<tr><td>Tasks avg wait time:
Indicates the total wait time for this wait type during the last measurement period.</td><td>MilliSecs</td><td>A low value is desired for this measure.
When a user complaints that query execution takes too long, you can compare the value of this measure across wait types to know which type of wait is the key contributor to delays in query processing.</td></tr>
</table>

	Tasks avg signal wait time: Indicates the total signal wait time for this wait type during the last measurement period.	MilliSecs	The *signal wait* is the time between when a worker has been granted access to the resource and the time it gets scheduled on the CPU. A high value for this measure may imply a high CPU contention. To know which wait type registered the highest signal wait time and probably caused the CPU contention, compare the value of this measure across wait types.
	Tasks avg resource wait time: Indicates the total resource wait time for this wait type during the last measurement period.	MilliSecs	*Resource wait time* is the actual time a worker waited for the resource to be available. A high value for this measure indicates a delay in acquiring a resource. To know which wait type waited the longest for a resource and therefore contributed to a server slowdown, compare the value of this measure across wait types.
	Tasks wait time: Indicates the percentage of total wait time (across wait types) during which wait events of this type occurred.	Percent	When a user complaints that query execution takes too long, you can compare the value of this measure across wait types to know which type of wait is the key contributor to delays in query processing.
	Tasks signal wait time: Indicates the percentage of total signal wait time (across wait types) during which wait events of this type waited for a signal.	Percent	The *signal wait* is the time between when a worker has been granted access to the resource and the time it gets scheduled on the CPU. A high value for this measure may imply a high CPU contention. To know which wait type registered the highest signal wait time and probably caused the CPU contention, compare the value of this measure across wait types.
	Tasks resource wait time: Indicates the percentage of total resource wait time (across wait types) during which wait events of this type waited for a resource.	Percent	*Resource wait time* is the actual time a worker waited for the resource to be available. A high value for this measure indicates a delay in acquiring a resource. To know which wait type waited the longest for a resource and therefore contributed to a server slowdown, compare the value of this measure across wait types.

3.1.2 SQL Engine Test

The SQL Engine test reports statistics related to the MS SQL server engine.

Purpose	Reports statistics related to the MS SQL server engine
Target of the test	An MS SQL server

Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – By default, this test requires **Sysadmin** privileges to execute. This is because, by default, the test pulls out metrics uing a stored procedure named *sp_monitor*, which requires **Sysadmin** privileges for execution. The **USER** parameter should hence be configured with the name of a **USER** who has been assigned the **Sysadmin** role. However, if you prefer not to expose the credentials of a **Sysadmin** owing to security concerns, then first ensure that the test does not use the *sp_monitor* stored procedure. For this, set the **USE SP MONITOR** parameter of this test to **No**. Then, against the **USER** parameter, specify the name of a SQL user who has been assigned the **Public** role. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **USE SP MONITOR** - By default, this flag is set to **Yes**, indicating that this test uses the *sp_monitor* stored procedure (by default) to pull out the required metrics from the target server. This stored procedure mandates the **Sysadmin** role - i.e., you should configure the test with the credentials of a **USER** with the **Sysadmin** role, if you want the test to use the *sp_monitor*. Moreover, even if the required privileges are granted to the test, in some environments, the *sp_monitor* procedure may result in errors. Administrators of high-security Windows environments may not want to expose the credentials of their **Sysadmin** users. Neither would they want error-prone stored procedures to execute in their environment. In such environments therefore, you can use queries to extract the desired metrics from the Microsoft SQL server, instead of the *sp_monitor* procedure. To enable the use of queries, set this flag to **No**.

	12. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every MS SQL server monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Cpu usage: The percentage of time for which the server's CPU was engaged in processing requests to the server	Percent	A high value of this measure indicates a heavy load on the server. If this value comes close to 100%, it could indicate a probable delay in the processing of subsequent requests to the server. The detailed diagnosis measures associated with the *Background processes* measure of the MsSqlSysProcesses test will help you identify the processes that are consuming excessive CPU resources.
	I/O usage: The percentage of time for which the server was engaged in performing input/output operations	Percent	
	CPU idle time: The percentage of time for which the server was idle	Percent	A low value of this measure is indicative of high CPU utilization.
	Packets received: The rate at which input packets were read by the SQL server	Pkts/Sec	This measure is an indicator of the traffic to the server.
	Packets sent: The rate at which output packets were read by the SQL server	Pkts/Sec	This measure is an indicator of traffic from the server.
	Packet errors: The rate at which packet errors occurred	Errors/Sec	Ideally, this value should be 0.
	Disk reads: The rate of disk reads performed by the MS SQL server	Reads/Sec	The value of this measure should be kept at a minimum, as disk reads are expensive operations. Ideally, data reads should be performed from the server cache and not directly from the disk. To ensure effective cache usage, allocate adequate memory to the MS SQL server.

	Disk writes: The rate of disk writes performed by the MS SQL server	Writes/Sec	The value of this measure should be kept at a minimum, as disk writes are expensive operations. Ideally, data should be written to the data cache and not directly to the disk. To ensure effective cache usage, allocate adequate memory to the MS SQL server.
	Disk I/O errors: The rate of errors encountered by the MS SQL server while reading and writing	Errors/Sec	Disk read/write errors are normally caused by the following reasons: • Semaphore contention • Excessive disk space consumption

3.1.3 SQL Errors Test

This test reports the rate at which errors occur on the MS SQL Server 2005 (or above). This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the rate at which errors occur on the MS SQL Server 2005 (or above)
Target of the test	An MS SQL server 2005 (or above)
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. INSTANCE - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the INSTANCE parameter.
5. USEPERFMON – By default, this flag is set to Yes, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to No. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to No.
6. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
7. SSL - By default, the SSL flag is set to No, indicating that the target MS SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled MS SQL server, set the SSL flag to Yes.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every error type on the MS SQL Server 2005 (or above) that is being monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Error rate:
Indicates the rate at which errors of this type occur on the MS SQL Server 2005 (or above).</td><td>Errors/Sec</td><td>A very high value of this measure indicates a problem condition requiring further investigation.</td></tr>
</table>

3.1.4 SQL Uptime Test

In most production environments, it is essential to monitor the uptime of critical servers in the infrastructure. By tracking the uptime of each of the servers, administrators can determine what percentage of time a server has been up. Comparing this value with service level targets, administrators can determine the most trouble-prone areas of the infrastructure.

In some environments, administrators may schedule periodic reboots of their servers. By knowing that a specific server has been up for an unusually long time, an administrator may come to know that the scheduled reboot task is not working on a server.

This MsSqlUptimeTest monitors the uptime of the SQL servers.

Note:

This test is applicable only to Microsoft SQL Server version 2008 and above.

Purpose	To monitor the uptime of a Windows or Unix server
Target of the test	A Windows or Unix server
Agent deploying the test	An internal agent
Configurable parameters for 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. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 4. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 5. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 6. **PASSWORD** - The password of the specified **USER** 7. **CONFIRM PASSWORD** - Confirm the password by retyping it 8. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 9. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 10. **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.

	11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 12. **REPORTMANAGERTIME** – By default, this flag is set to **Yes**, indicating that, by default, the detailed diagnosis of this test, if enabled, will report the shutdown and reboot times of the device in the manager's time zone. If this flag is set to **No**, then the shutdown and reboot times are shown in the time zone of the system where the agent is running (i.e., the system being managed for agent-based monitoring, and the system on which the remote agent is running - for agentless monitoring).		
Outputs of the test	1. One set of results for every server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Has SQL server been restarted?: Indicates whether the server has been rebooted during the last measurement period or not.	Boolean	If this measure shows 1, it means that the server was rebooted during the last measurement period. By checking the time periods when this metric changes from 0 to 1, an administrator can determine the times when this server was rebooted. The detailed diagnosis of this measure, if enabled, indicates the date/time at which the server was shutdown, the date on which it was restarted, the duration of the shutdown, and whether the server was shutdown as part of a maintenance outine.
	Uptime since the last measurement: Indicates the time period that the server has been up since the last time this test ran.	Secs	If the server has not been rebooted during the last measurement period and the agent has been running continuously, this value will be equal to the measurement period. If the server was rebooted during the last measurement period, this value will be less than the measurement period of the test. For example, if the measurement period is 300 secs, and if the server was rebooted 120 secs back, this metric will report a value of 120 seconds. The accuracy of this metric is dependent on the measurement period – the smaller the measurement period, greater the accuracy.
	Uptime: Indicates the total time that the server has been up since its last reboot.	Mins	Administrators may wish to be alerted if a server has been running without a reboot for a very long period. Setting a threshold for this metric allows administrators to determine such conditions.

3.1.5 SQL System Processes Test

This test reports details about the system processes running on an MS SQL server.

Purpose	Reports details about the system processes running on an MS SQL server
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

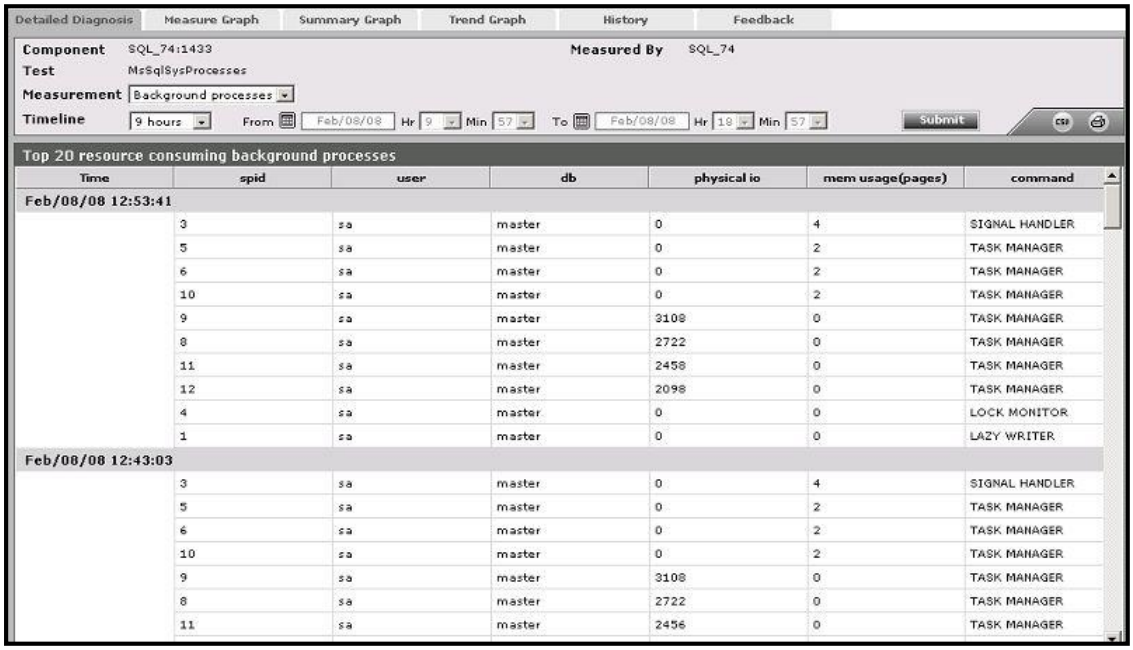
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **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. 12. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every MS SQL server monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Total processes:** The total number of MS SQL server processes	Number	The value of this measure is the sum of the number of background, running, sleeping, rollback, and suspended processes.
	Background processes: The total number of background processes run by the MS SQL server rather than by a user process	Number	The detailed diagnosis of this measure, if enabled, provides the details pertaining to the background processes currently executing.
	Running processes: The total number of running processes	Number	The detailed diagnosis of this measure, if enabled, provides details such as the ID of the running processes, the user executing each of the processes, the database on which every process is executing etc. **Note that while the query used by the eG agent for collecting the metrics of this test will be counted as a Running process, the detailed diagnosis of this measure will not include this eG query.**
	Sleeping processes: The total number of sleeping processes	Number	The detailed diagnosis of this measure, if enabled, provides details such as the ID of the sleeping processes, the user executing each of the processes, the database on which every process is executing, the sleep status, sleep time etc.
	Rollback processes: The total number of processes that were rolled back	Number	The detailed diagnosis of this measure, if enabled, reveals information such as the ID of the rolled back processes, the user executing each of the processes, the database on which every process is executing, etc.
	Blocked processes: If a process attempts to access a resource that is already in use by another process, then such a process will be blocked until such time that the other process releases the resource. This measures indicates the total number of blocked processes.	Number	The detailed diagnosis of the Blocked processes measure, if enabled, reveals information such as the ID of the blocked processes, the user executing each of the processes, the database on which every process is executing, the waiting time of the blocked process, etc. These details aid the user in identifying the blocked processes, the processes that are blocking them (i.e. the process that currently holds a lock on the resource), and also the duration for which the processes have been blocked. If a process is found to hold a lock for too long a time, then such processes can be killed so as to free the resource for the corresponding blocked process.

	Suspended processes: **Indicates the number of processes that are currently suspended.**	Number	A Microsoft SQL server marks a process as "suspended" when the process has made a request to a non-SQL process or resource and is awaiting a response. This happens a lot when you have slow disk drives; processes will be suspended while the SQL server waits for the drive to return data or report back after committing. Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the suspended processes.
	Rollback processes: Indicates the number of sessions initiated by this user in which transaction rollbacks are in progress.	Number	Ideally, the value of this measure should be low. If this value is very close to the Total processes value for a user, it indicates that many transactions executed by that user are being rolled back. This is a cause for concern, as rollbacks are expensive operations that need to be kept at a minimum; if not, processing overheads increase and the overall performance of the server deteriorates. The detailed diagnosis of this measure, if enabled, will provide the complete details of the user sessions with transaction rollbacks.
	Dormant processes: Indicates the number of processes being reset by the Microsoft SQL server.	Number	Ideally, the value of this measure should be low. If this value is high, it indicates that many processes are being reset. The detailed diagnosis of this measure, if enabled, will provide the complete details of the reset processes.
	Pending processes: Indicates the number of processes that are waiting for a worker thread to become available.	Number	A low value is desired for this measure. If the value of this measure is high, it indicates that many are unable to execute owing to the lack of worker threads. Use the detailed diagnosis of this measure to know which processes are waiting for worker threads.

	Spinloop processes: Indicates the number of processes that are waiting for a spinlock to free.	Number	Spinlocks are lightweight synchronization primitives which are used to protect access to data structures. They are generally used when it is expected that access to a given data structure will need to be held for a very short period of time. When a thread attempting to acquire a spinlock is unable to obtain access it executes in a loop periodically checking to determine if the resource is available instead of immediately yielding. After some period of time a thread waiting on a spinlock will yield before it is able to acquire the resource in order to allow other threads running on the same CPU to execute. This is known as a backoff. The detailed diagnosis of this measure will reveal the processes that are waiting for spinlock to free.

The detailed diagnosis of the *Background processes* measure, if enabled, provides the details pertaining to the background processes currently executing (see Figure 3.3). This information helps the user identify the processes consuming excessive CPU and memory resources. If found necessary, such processes can be killed so as to free adequate CPU resources.



Time	spid	user	db	physical io	mem usage(pages)	command
Feb/08/08 12:53:41						
	3	sa	master	0	4	SIGNAL HANDLER
	5	sa	master	0	2	TASK MANAGER
	6	sa	master	0	2	TASK MANAGER
	10	sa	master	0	2	TASK MANAGER
	9	sa	master	3108	0	TASK MANAGER
	8	sa	master	2722	0	TASK MANAGER
	11	sa	master	2458	0	TASK MANAGER
	12	sa	master	2098	0	TASK MANAGER
	4	sa	master	0	0	LOCK MONITOR
	1	sa	master	0	0	LAZY WRITER
Feb/08/08 12:43:03						
	3	sa	master	0	4	SIGNAL HANDLER
	5	sa	master	0	2	TASK MANAGER
	6	sa	master	0	2	TASK MANAGER
	10	sa	master	0	2	TASK MANAGER
	9	sa	master	3108	0	TASK MANAGER
	8	sa	master	2722	0	TASK MANAGER
	11	sa	master	2456	0	TASK MANAGER

Figure 3.3: The detailed diagnosis of the Background processes measure

The detailed diagnosis of the *Running processes* measure, if enabled, provides details such as the ID of the running processes, the user executing each of the processes, the database on which every process is executing etc. This information enables the user to understand the general user behavior on the server (see Figure 3.4).

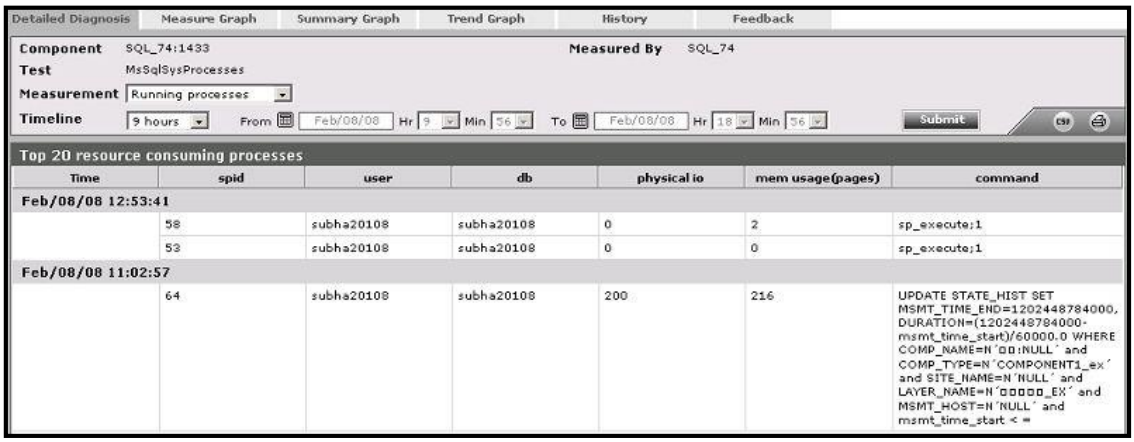


Figure 3.4: The detailed diagnosis of the Running processes measure

The detailed diagnosis of the *Sleeping processes* measure, if enabled, provides details such as the ID of the sleeping processes, the user executing each of the processes, the database on which every process is executing, the sleep status, sleep time etc. Using this information, users can identify those processes that have been idle for a long period of time (see Figure 3.5).

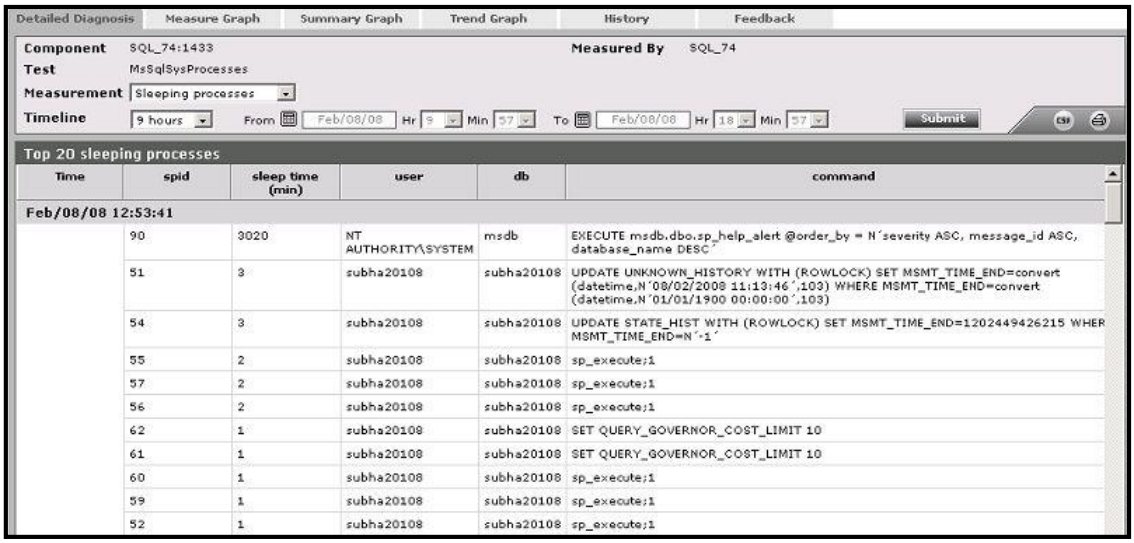


Figure 3.5: The detailed diagnosis of the Sleeping processes measure

The detailed diagnosis of the *Rollback processes* measure, if enabled, reveals information such as the ID of the rolledback processes, the user executing each of the processes, the database on which every process is executing, etc. Rollbacks are expensive operations on a server. The detailed measures provided by eG in this regard, enable the user to isolate the specific queries that have rolledback. Further analysis of these queries can be performed, in order to figure out the reason for the rollback and take adequate measures to prevent it from recurring.

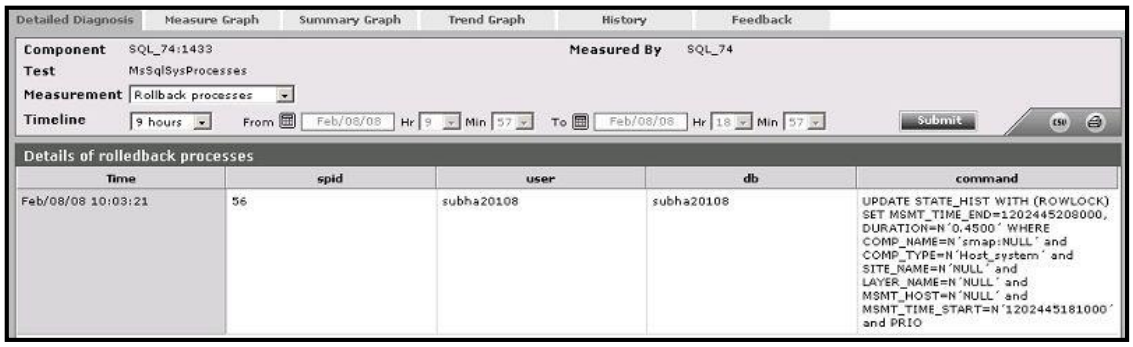


Figure 3.6: The detailed diagnosis of the Rollback processes measure

3.1.6 SQL Current Request Statistics Test

In the database context, the connection between the user process and the server process is called a session. The server process communicates with the connected user process and performs tasks on behalf of the users.

This test tracks the resource usage of the sessions to the target MS SQL server. In the process, the test turns the spotlight on resource-intensive SQL server sessions and the queries executed by such sessions that may require fine-tuning. Additionally, the test also reports the average wait time of sessions, leads you to that session that has been waiting for the maximum time, and points you to the exact query that the session has been taking too long to execute. Inefficient queries are thus revealed, enabling you to quickly initiate query optimization measures.

Purpose	Tracks the resource usage of the sessions to the target MS SQL server
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">
1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. SSL – If the MS SQL server being monitored is an SSL-enabled server, then set the SSL flag to Yes. If not, then set the SSL flag to No.
5. INSTANCE - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the INSTANCE parameter.
6. USER – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the Sysadmin role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE, and PUBLIC roles in this text box.
7. PASSWORD - The password of the specified USER
8. CONFIRM PASSWORD - Confirm the password by retyping it
9. DOMAIN - By default, none is displayed in the DOMAIN text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the DOMAIN can continue to be none. On the other hand, if 'Windows only' authentication has been enabled, then, in the DOMAIN text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the USER name and PASSWORD that you provide should be that of a user authorized to access the monitored SQL server.
10. ISNTLMV2 - In some Windows networks, NTLM (NT LAN Manager) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the ISNTLMV2 flag is set to No, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to Yes if NTLMv2 is enabled on the target host.
11. 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.
2. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every MS SQL server monitored</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Avg memory usage:** Indicates the average amount of memory that is currently used by all sessions of this MS SQL server.	KB	A high value indicates that one/more MS SQL sessions are consuming high memory. Use the detailed diagnosis of the *Max memory usage* measure to identify which session is consuming maximum memory. To reduce the memory consumption of the session, you may have to optimize the query displayed in the **SQL TEXT** column of the detailed diagnosis. To optimize the query, you would be required to do any one of the following: c. Check the fragmentation activity of the disks of the MS SQL server; d. Add additional indexes to the MS SQL server or; e. Include a hint to the query which would considerably reduce the memory usage of the server.
	Max memory usage: Indicates the maximum memory used by the SQL server sessions.	KB	The detailed diagnosis of this measure, if enabled, displays the session ID, the name of the database accessed by the session, the login name of the user who initiated the session, the login time of the user, the request start time, when the session was established, the session wait time and type, the session duration, the time for which the session hogged the CPU, the memory usage of the session, the session status, the total number of reads, writes, and logical reads performed by the session on the database, and the query executed by the session. From this information, you can easily identify the session that is consuming the maximum CPU/memory, the session that has been waiting for the maximum time for the query to execute, the session that has performed the maximum I/O activities on the SQL server, and the query that is responsible for all such resource-intensive tasks.

	Avg CPU time: Indicates the average time for which the SQL sessions used the CPU resources of the SQL server.	Secs	A high value indicates that one/more MS SQL sessions are hogging the CPU. Use the detailed diagnosis of the *Max CPU time* measure to identify which session is consuming the CPU resources excessively. To reduce the CPU consumption of a session, you may have to optimize the query displayed in the **SQL TEXT** column of the detailed diagnosis. To optimize the query, you would be required to do any one of the following: f. Check the fragmentation activity of the disks of the MS SQL server; g. Add additional indexes to the MS SQL server or; h. Include a hint to the query which would considerably reduce the memory usage of the server.
	Max CPU time: Indicates the maximum time for which the SQL sessions used the CPU.	Secs	The detailed diagnosis of this measure, if enabled, displays the session ID, the name of the database accessed by the session, the login name of the user who initiated the session, the login time of the user, the request start time, when the session was established, the session wait time and type, the session duration, the time for which the session hogged the CPU, the memory usage of the session, the session status, the total number of reads, writes, and logical reads performed by the session on the database, and the query executed by the session. From this information, you can easily identify the session that is consuming the maximum CPU/memory, the session that has been waiting for the maximum time for the query to execute, the session that has performed the maximum I/O activities on the SQL server, and the query that is responsible for all such resource-intensive tasks.

	Avg wait time: Indicates the average time for which the SQL sessions were waiting.	Secs	A high value indicates that one/more MS SQL sessions are waiting too long to perform a task – typically, to execute a query. Use the detailed diagnosis of the *Max wait time* measure to identify which session is taking too long for query execution. To reduce the wait time of a session, you may have to optimize the query displayed in the **SQL TEXT** column of the detailed diagnosis. To optimize the query, you would be required to do any one of the following: • Check the fragmentation activity of the disks of the MS SQL server; • Add additional indexes to the MS SQL server or; • Include a hint to the query which would considerably reduce the memory usage of the server.
	Max wait time: Indicates the maximum time for which the SQL sessions waited.	Secs	The detailed diagnosis of this measure, if enabled, displays the session ID, the name of the database accessed by the session, the login name of the user who initiated the session, the login time of the user, the request start time, when the session was established, the session wait time and type, the session duration, the time for which the session hogged the CPU, the memory usage of the session, the session status, the total number of reads, writes, and logical reads performed by the session on the database, and the query executed by the session. From this information, you can easily identify the session that is consuming the maximum CPU/memory, the session that has been waiting for the maximum time for the query to execute, the session that has performed the maximum I/O activities on the SQL server, and the query that is responsible for all such resource-intensive tasks.

	Avg I/O time for current queries: Indicates the average time taken by current queries for I/O processing.	Secs	A low value is desired for this measure. A high value indicates that one/more queries are I/O-intensive.
	Max I/O time for current queries: Indicates the maximum time that the current queries took for I/O processing.	Secs	If the value of this measure exceeds 10 seconds, you will have to check the disk I/O subsystem for the proper placement of files – LDF and MDF on separate drives, tempDB on a separate drive, hot spot tables on separate filegroups. I/O can also be reduced if the SQL server uses cover index instead of cluster index.

3.1.7 SQL Cached Queries Test

SQL Server maintains a cache, but not with canned results for queries. In an OLTP system, many tables are frequently updated; it is therefore highly unlikely that the same query yields the same result twice. Similarly, the likelihood of the same query reappering with exactly the same parameters is also very less. What SQL Server stores in its cache therefore, are recently accessed data pages, as well as query plans for recently submitted queries and invoked stored procedures. This makes it possible to retrieve the result of a query without accessing the disk for frequently accessed tables. Too few queries in the cache means more direct disk accesses! To minimize reads/writes to physical disks, more number of queries should execute in the cache. Using this test, you can determine the number of queries that are currently executing in the cache and also figure out the impact of cache misses on the physical disks. These metrics reveal whether/not cache usage is at a desired level. In the process, the test also measures the resource usage of and the I/O activity generated by the cached queries and sheds light on time-consuming, resource-intensive and I/O-intensive queries that are executing in the cache.

Purpose	Monitors cached queries and reports the resource usage of and the I/O activity generated by the cached queries
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

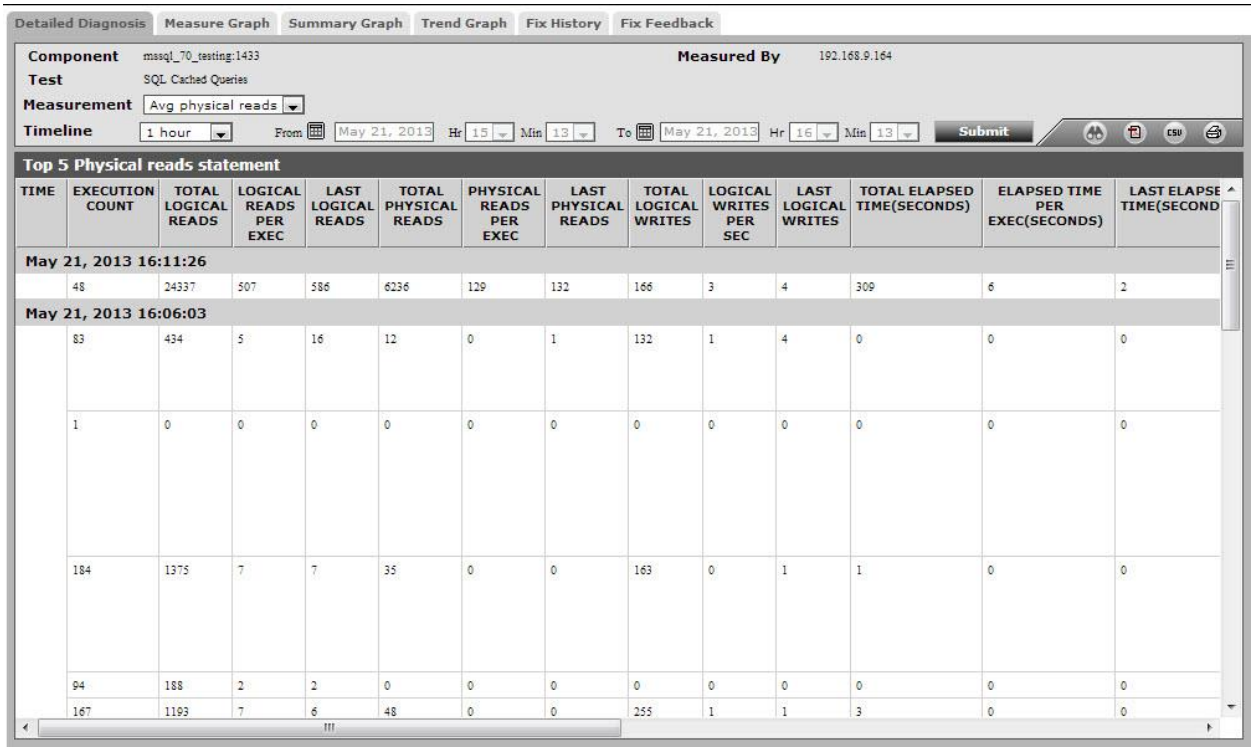
<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. SSL – If the MS SQL server being monitored is an SSL-enabled server, then set the SSL flag to Yes. If not, then set the SSL flag to No.
5. INSTANCE - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the INSTANCE parameter.
6. USER – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the Sysadmin role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE, and PUBLIC roles in this text box.
7. PASSWORD - The password of the specified USER
8. CONFIRM PASSWORD - Confirm the password by retyping it
9. DOMAIN - By default, none is displayed in the DOMAIN text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the DOMAIN can continue to be none. On the other hand, if 'Windows only' authentication has been enabled, then, in the DOMAIN text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the USER name and PASSWORD that you provide should be that of a user authorized to access the monitored SQL server.
10. ISNTLMV2 - In some Windows networks, NTLM (NT LAN Manager) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the ISNTLMV2 flag is set to No, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to Yes if NTLMv2 is enabled on the target host.
11. 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.
3. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every MS SQL server monitored</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Physical reads:** Indicates the rate at which the queries directly executed on and read from the physical disk.	Reads/Sec	A high value could indicate that direct disk accesses are occurring too frequently. This in turn implies poor cache usage. You may consider resizing your cache to accommodate more number of queries, so that direct disk reads are reduced.
	Avg physical reads: Indicates the average number of reads performed by the queries that executed directly on the physical disk.	Number	A high value indicates that one/more queries are reading too frequently from the physical disk. This is an unhealthy practice and can be attributed poor cache usage. Use the detailed diagnosis of this measure to know which queries are not executing in the cache and the number of times each of these queries read directly from the physical disk. This way, you can quickly identify that query which exerts the maximum pressure on the physical disk.
	Logical reads: Indicates the rate of data reads performed by queries to the cache.	Reads/Sec	A high value is desired for this measure. A low value is indicative of ineffective cache usage, typically caused by improper cache size.
	Avg logical reads: Indicates the average number of data reads performed by queries to the cache.	Number	A high value is desired for this measure. A low value is indicative of ineffective cache usage, typically caused by improper cache size. You can also use the detailed diagnosis of this measure to view the top-5 queries in terms of number of logical reads. This way, you can precisely identify the most I/O-intensive query to the cache.
	Logical writes: Indicates the rate at which the cached queries performed writes.	Writes/Sec	A high value is desired for this measure. A low value is indicative of ineffective cache usage, typically caused by improper cache size.
	Avg logical writes: Indicates the average number of times data writes were performed by a query to the cache.	Number	A high value is desired for this measure. A low value is indicative of ineffective cache usage, typically caused by improper cache size. You can also use the detailed diagnosis of this measure to view the top-5 queries in terms of number of logical writes. This way, you can precisely identify the most I/O-intensive query to the cache.
	CPU time: Indicates the percentage of time for which the cached queries hogged the CPU.	Percent	A high value is indicative of excessive CPU usage by the cached queries. Use the detailed diagnosis of this measure to know which query is CPU-intensive.

	Max elapsed time: Indicates the maximum time taken by the cached queries for execution.	Secs	If the value of this measure is very high, it could either indicate that the database is unable to process the queries quickly or that one/more queries to the database are taking too long to execute. Improper indexing and fragmented tables in the database are common causes for slowdowns at the database-level. Besides the above, queries that are improperly structured can also take time to execute. The longer a query executes on the database, higher would be the resource consumption of that query. It is therefore imperative that such resource-intensive queries are quickly isolated and fine-tuned, so as to prevent degradations in the performance of the database server. Using the detailed diagnosis of this measure, you can rapidly identify the resource-intensive queries to the database.
	Recently executed queries: Indicates the number of queries that executed in the cache since the last measurement period.	Number	A consistent rise in the value of this measure is a sign of optimal cache usage and minimal direct disk accesses.

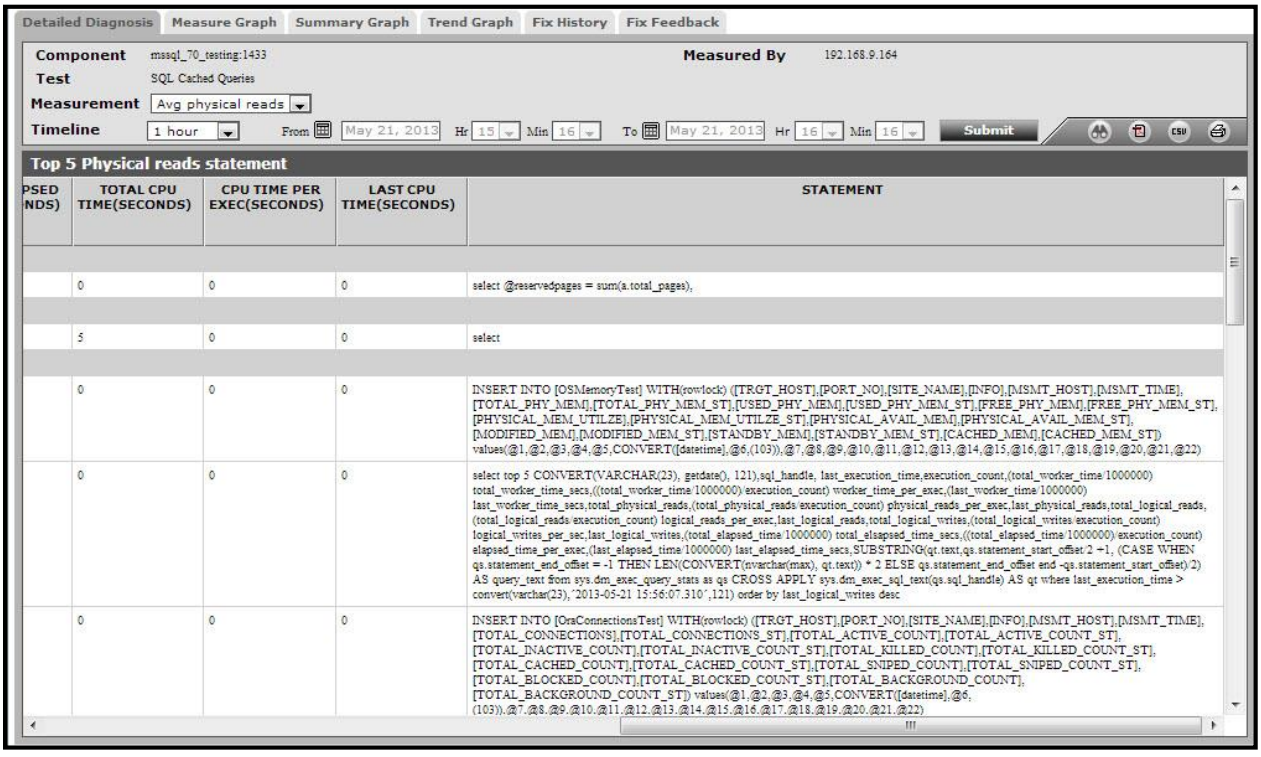
The detailed diagnosis of the *Avg physical read* measure lists the top-5 queries in terms of the number of times they

read directly from the physical disk. The query that exerts the maximum pressure on the disk can thus be isolated.



TIME	EXECUTION COUNT	TOTAL LOGICAL READS	LOGICAL READS PER EXEC	LAST LOGICAL READS	TOTAL PHYSICAL READS	PHYSICAL READS PER EXEC	LAST PHYSICAL READS	TOTAL LOGICAL WRITES	LOGICAL WRITES PER SEC	LAST LOGICAL WRITES	TOTAL ELAPSED TIME(SECONDS)	ELAPSED TIME PER EXEC(SECONDS)	LAST ELAPSE TIME(SECOND
May 21, 2013 16:11:26													
	48	24337	507	586	6236	129	132	166	3	4	309	6	2
May 21, 2013 16:06:03													
	83	434	5	16	12	0	1	132	1	4	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0
	184	1375	7	7	35	0	0	163	0	1	1	0	0
	94	188	2	2	0	0	0	0	0	0	0	0	0
	167	1193	7	6	48	0	0	255	1	1	3	0	0

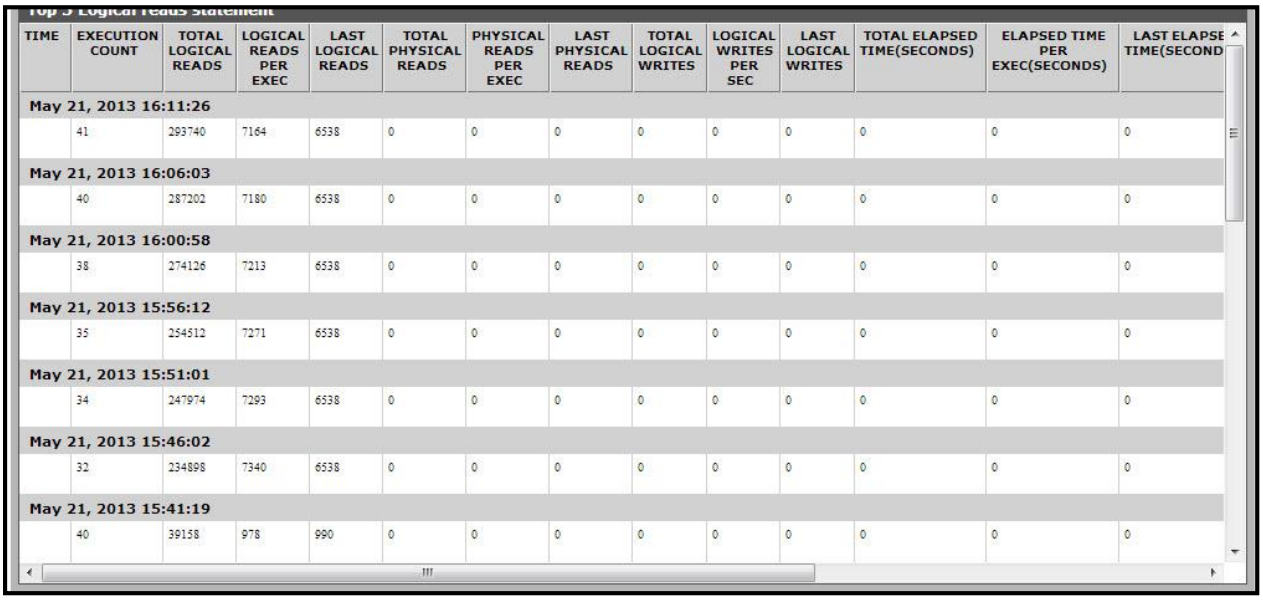
Figure 3.7a: The detailed diagnosis of the Avg physical reads measure



PSED NDS)	TOTAL CPU TIME(SECONDS)	CPU TIME PER EXEC(SECONDS)	LAST CPU TIME(SECONDS)	STATEMENT
	0	0	0	select @reservedpages = sum(a.total_pages),
	5	0	0	select
	0	0	0	INSERT INTO [OSMemoryTest] WITH(rowlock) ([TRGT_HOST],[PORT_NO],[SITE_NAME],[INFO],[MSMT_HOST],[MSMT_TIME], [TOTAL_PHY_MEM],[TOTAL_PHY_MEM_ST],[USED_PHY_MEM],[USED_PHY_MEM_ST],[FREE_PHY_MEM],[FREE_PHY_MEM_ST], [PHYSICAL_MEM_UTILZE],[PHYSICAL_MEM_UTILZE_ST],[PHYSICAL_AVAIL_MEM],[PHYSICAL_AVAIL_MEM_ST], [MODIFIED_MEM],[MODIFIED_MEM_ST],[STANDBY_MEM],[STANDBY_MEM_ST],[CACHED_MEM],[CACHED_MEM_ST]) values(@1,@2,@3,@4,@5,CONVERT([datetime],@6,(103)),@7,@8,@9,@10,@11,@12,@13,@14,@15,@16,@17,@18,@19,@20,@21,@22)
	0	0	0	select top 5 CONVERT(VARCHAR(23), getdate(), 121),sql_handle, last_execution_time,execution_count,(total_worker_time/1000000) total_worker_time_secs,((total_worker_time/1000000)/execution_count) worker_time_per_exec,(last_worker_time/1000000) last_worker_time_secs,total_physical_reads,(total_physical_reads/execution_count) physical_reads_per_exec,last_physical_reads,total_logical_reads, (total_logical_reads/execution_count) logical_reads_per_exec,last_logical_reads,total_logical_writes,(total_logical_writes/execution_count) logical_writes_per_sec,last_logical_writes,(total_elapsed_time/1000000) total_elsapsed_time_secs,((total_elapsed_time/1000000)/execution_count) elapsed_time_per_exec,(last_elapsed_time/1000000) last_elapsed_time_secs,SUBSTRING(qt.text,qs.statement_start_offset/2 +1, (CASE WHEN qs.statement_end_offset = -1 THEN LEN(CONVERT(nvarchar(max), qt.text)) * 2 ELSE qs.statement_end_offset end -qs.statement_start_offset)/2) AS query_text from sys.dm_exec_query_stats as qs CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) AS qt where last_execution_time > convert(varchar(23),'2013-05-21 15:56:07.310',121) order by last_logical_writes desc
	0	0	0	INSERT INTO [OraConnectionsTest] WITH(rowlock) ([TRGT_HOST],[PORT_NO],[SITE_NAME],[INFO],[MSMT_HOST],[MSMT_TIME], [TOTAL_CONNECTIONS],[TOTAL_CONNECTIONS_ST],[TOTAL_ACTIVE_COUNT],[TOTAL_ACTIVE_COUNT_ST], [TOTAL_INACTIVE_COUNT],[TOTAL_INACTIVE_COUNT_ST],[TOTAL_KILLED_COUNT],[TOTAL_KILLED_COUNT_ST], [TOTAL_CACHED_COUNT],[TOTAL_CACHED_COUNT_ST],[TOTAL_SNIPED_COUNT],[TOTAL_SNIPED_COUNT_ST], [TOTAL_BLOCKED_COUNT],[TOTAL_BLOCKED_COUNT_ST],[TOTAL_BACKGROUND_COUNT], [TOTAL_BACKGROUND_COUNT_ST]) values(@1,@2,@3,@4,@5,CONVERT([datetime],@6, (103)),@7,@8,@9,@10,@11,@12,@13,@14,@15,@16,@17,@18,@19,@20,@21,@22)

Figure 3.7b: The detailed diagnosis of the Avg physical reads measure

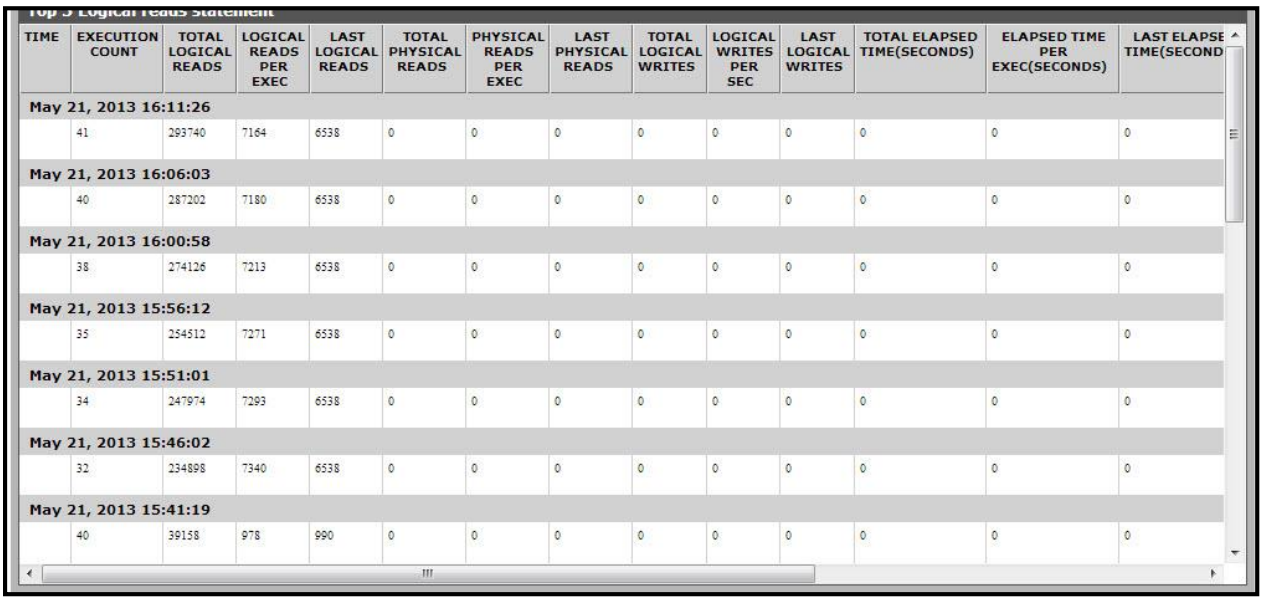
The detailed diagnosis of the *Avg logical reads* measure lists the top-5 cached queries in terms of the number of logical reads. The cached query that performed the maximum number of data reads can be identified. You can even analyse the query to figure out whether the number of reads it generates is justified or not; if not, you may have to optimize the query.



TIME	EXECUTION COUNT	TOTAL LOGICAL READS	LOGICAL READS PER EXEC	LAST LOGICAL READS	TOTAL PHYSICAL READS	PHYSICAL READS PER EXEC	LAST PHYSICAL READS	TOTAL LOGICAL WRITES	LOGICAL WRITES PER SEC	LAST LOGICAL WRITES	TOTAL ELAPSED TIME(SECONDS)	ELAPSED TIME PER EXEC(SECONDS)	LAST ELAPSE TIME(SECOND
May 21, 2013 16:11:26													
	41	293740	7164	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 16:06:03													
	40	287202	7180	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 16:00:58													
	38	274126	7213	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 15:56:12													
	35	254512	7271	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 15:51:01													
	34	247974	7293	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 15:46:02													
	32	234898	7340	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 15:41:19													
	40	39158	978	990	0	0	0	0	0	0	0	0	0

Figure 3.8: The detailed diagnosis of the Avg logical reads measure

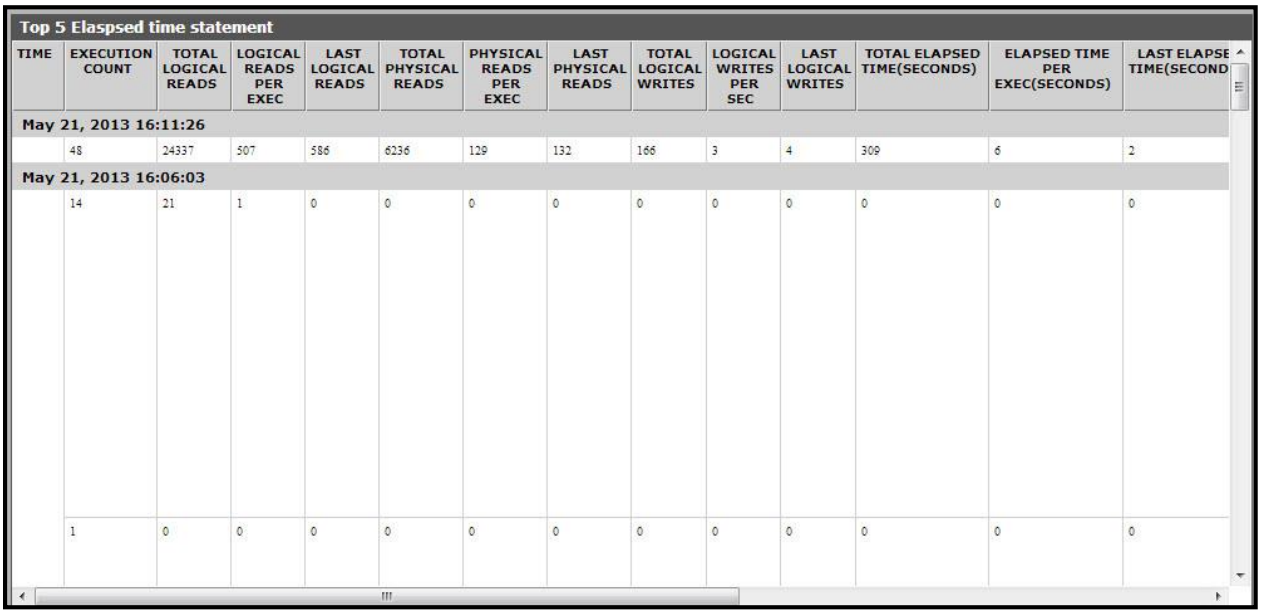
The detailed diagnosis of the *Avg logical writes* measure lists the top-5 cached queries in terms of the number of logical writes. The cached query that performed the maximum number of data writes can be inferred from this. You can even analyse the query to figure out whether the number of writes it generates is justified or not; if not, you may have to optimize the query.



TIME	EXECUTION COUNT	TOTAL LOGICAL READS	LOGICAL READS PER EXEC	LAST LOGICAL READS	TOTAL PHYSICAL READS	PHYSICAL READS PER EXEC	LAST PHYSICAL READS	TOTAL LOGICAL WRITES	LOGICAL WRITES PER SEC	LAST LOGICAL WRITES	TOTAL ELAPSED TIME(SECONDS)	ELAPSED TIME PER EXEC(SECONDS)	LAST ELAPSE TIME(SECOND
May 21, 2013 16:11:26													
	41	293740	7164	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 16:06:03													
	40	287202	7180	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 16:00:58													
	38	274126	7213	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 15:56:12													
	35	254512	7271	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 15:51:01													
	34	247974	7293	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 15:46:02													
	32	234898	7340	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 15:41:19													
	40	39158	978	990	0	0	0	0	0	0	0	0	0

Figure 3.9: The detailed diagnosis of the Avg logical reads measure

The detailed diagnosis of the *Max elapsed time* measure lists the top-5 cached queries in terms of the time they took to complete execution. The query that took the longest time to execute can thus be easily identified. You can analyse whey that query took long to execute, assess the resource foot print of that query, and if required, attempt to fine-tune the query to reduce execution time / resource usage.



Top 5 Elaspsed time statement

TIME	EXECUTION COUNT	TOTAL LOGICAL READS	LOGICAL READS PER EXEC	LAST LOGICAL READS	TOTAL PHYSICAL READS	PHYSICAL READS PER EXEC	LAST PHYSICAL READS	TOTAL LOGICAL WRITES	LOGICAL WRITES PER SEC	LAST LOGICAL WRITES	TOTAL ELAPSED TIME(SECONDS)	ELAPSED TIME PER EXEC(SECONDS)	LAST ELAPSE TIME(SECOND
May 21, 2013 16:11:26													
	48	24337	507	586	6236	129	132	166	3	4	309	6	2
May 21, 2013 16:06:03													
	14	21	1	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3.10: The detailed diagnosis of the Max elapsed time measure

The detailed diagnosis of the *Cpu time* measure lists the top-5 cached queries in terms of CPU usage. The most CPU-intensive query can thus be identified and the reasons for the same can be determined.

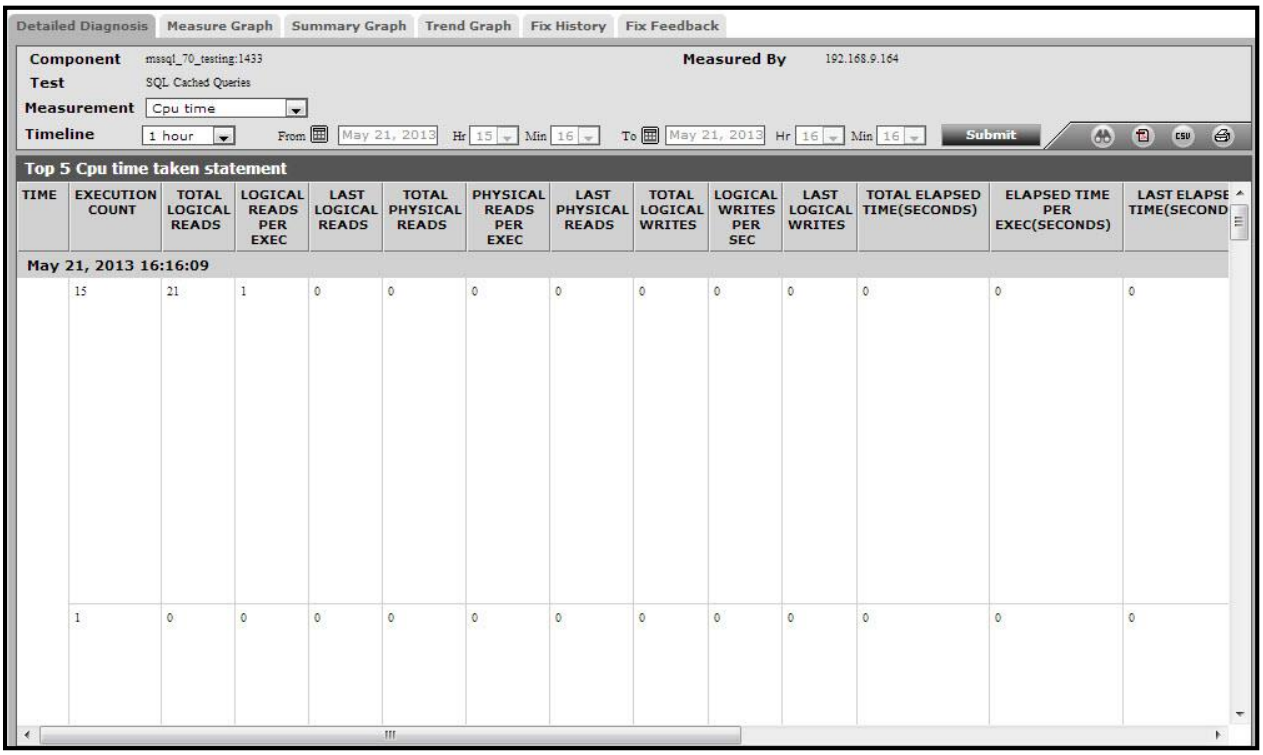


Figure 3.11: The detailed diagnosis of the Cpu time measure

3.1.1 SQL AlwaysOn Availability Test

The AlwaysOn Availability Groups feature is a high-availability and disaster-recovery solution that provides an enterprise-level alternative to database mirroring. Introduced in SQL Server 2012, AlwaysOn Availability Groups maximizes the availability of a set of user databases for an enterprise. An *availability group* supports a failover environment for a discrete set of user databases, known as *availability databases*, that failover together. An availability group supports a set of read-write primary databases and one to eight sets of corresponding secondary databases. Optionally, secondary databases can be made available for read-only access and/or some backup operations. An availability group fails over at the level of an availability replica. Failovers are not caused by database issues such as a database becoming suspect due to a loss of a data file, deletion of a database, or corruption of a transaction log. In environments where critical data is stored, it is important for the database to be highly available. The AlwaysOn Availability Groups if enabled, will help administrators in maintaining such high availability. Therefore, it is important to monitor the status of the AlwaysOn Availability Groups. The **SQL AlwaysOn Availability** test exactly helps in this regard.

This test reports whether the AlwaysOn Availability Group feature is enabled and the current state of the AlwaysOn Availability Groups manager.

Purpose	Reports whether the AlwaysOn Availability Group feature is enabled and the current state of the AlwaysOn Availability Groups manager
Target of the test	A Microsoft SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – While monitoring a Microsoft SQL Server 2012 and above, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

<table>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each database on the MS SQL server instance being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Is always-on enabled?:
Indicates whether/not the Always on Availability Groups feature is enabled.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>No</td><td>0</td></tr><tr><td>Yes</td><td>1</td></tr></table>Note:
This measure reports the Measure Values listed in the table above to indicate whether/not the Always on Availability Groups concept was enabled. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.</td></tr>
<tr><td>Alwayson manager status:
Indicates the current state of the Always on Availability Groups manager.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Failed</td><td>0</td></tr><tr><td>Pending Communication</td><td>1</td></tr><tr><td>Running</td><td>2</td></tr></table>Note:
This measure reports the Measure Values listed in the table above to indicate the current status of the Always on Availability Groups manager. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.</td></tr>
</table>

3.1.2 SQL AlwaysOn Member Status Test

The AlwaysOn feature not only combines the power of clustering and mirroring into one High Availability option, but also allows you to interact with the secondary databases. In addition, AlwaysOn Availability Groups allows you to configure failover for one database, a set of databases or the entire instance. Another important aspect of the AlwaysOn is that you can create multiple failover targets. If the Availability Group is enabled on multiple Microsoft SQL server instances in a cluster, then the administrators are required to monitor each member of the cluster node that is enabled with the AlwaysOn feature. In addition, if the failover concept has to be fool-proof then there arises a need for a file-

share witness or a disk witness. The file-share witness or disk witness is most commonly used when shared storage is available to a cluster. The file share witness or disk witness pings the members of a cluster and syncs the data from all the members to keep the database updated. In case of failover, the disk witness or file share witness will render the cluster node with an updated database. Since the file share witness or disk witness stores the updated database by constantly pinging the members of the cluster node on which AlwaysOn feature is enabled, it becomes mandatory to check the status of each member of the cluster node and the disk witness or file share witness. The **SQL AlwaysOn Member Status** test exactly helps you in this regard.

For each category available in the Microsoft SQL server instance enabled with AlwaysOn feature, this test reports the current status of each member. If the member is online, then this test will report whether/not the member is a primary member and whether the member has failed over.

Purpose	For each category available in the Microsoft SQL server instance enabled with AlwaysOn feature, this test reports the current status of each member. If the member is online, then this test will report whether/not the member is a primary member and whether the member has failed over.
Target of the test	A Microsoft SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – While monitoring a Microsoft SQL Server 2012 and above, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

<table>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each Category:Member available in the Microsoft SQL server that is being monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Member status:
Indicates the current state of this member.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Offline</td><td>0</td></tr><tr><td>Online</td><td>1</td></tr></table>Note:
This measure reports the Measure Values listed in the table above to indicate current state of this member. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.</td></tr>
<tr><td></td><td>Is primary?:
Indicates whether/not this member is the primary member.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>No</td><td>0</td></tr><tr><td>Yes</td><td>1</td></tr></table>Note:
This measure reports the Measure Values listed in the table above to indicate whether/not this member is the primary member. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.
This measure is not applicable for the DISK_WITNESS descriptor.</td></tr>
</table>

	Is switch over happened?: Indicates whether/not this member has failed over i.e., this member has switched over from primary to secondary and vice versa.		The values reported by this measure and their numeric equivalents are available in the table below: **Measure Value** / **Numeric Value** No / 0 Yes / 1 **Note:** This measure reports the **Measure Value**s listed in the table above to indicate whether/not this member has failed over. However, in the graph, this measure is indicated using the **Numeric Value**s listed in the above table. **This measure is not applicable for the DISK_WITNESS descriptor.**

3.1.3 SQL AlwaysOn Network Latency Test

If transaction log records are not sent quickly by the primary database or are not applied quickly by the secondary database, then the data in the primary and secondary databases will be out of sync; this will cause significant data loss during a failover. To avoid this, administrators must keep track of the log record traffic between the primary and secondary databases, proactively detect potential slowness in synchronization, figure out the probable source of the bottleneck, and clear it to ensure proper synchronization between the primary and secondary databases. This is where the **SQL AlwaysOn Network Latency** test helps.

This test measures the rate at which transaction log data is sent to the secondary database for synchronization on each SQL server instance, and the time taken by the secondary database to apply the data. In the process, the test pinpoints bottlenecks in database synchronization and where exactly the bottlenecks lie.

Purpose	Reports the number of transactions active in an instance of the Database Engine, and the effects of those transactions on resources such as the snapshot isolation level row version store in **tempdb**.
Target of the test	A Microsoft SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – While monitoring a Microsoft SQL Server 2012 and above, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for each database on the MS SQL server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Log sent: Indicates the amount of data (in bytes) sent from the primary availability replica to the secondary availability replica per second during the last measurement period.	KB/sec	
	Log transport: Indicates the amount of data (in bytes) sent over the network from the primary availability replica to the secondary availability replica per second during the last measurement period.	KB/sec	
	Log send wait time: Indicates the time duration for which the log stream messages were waiting in the *Flow Control* mode per second.	Msecs/sec	Ideally, the value of this measure should be low. A gradual/sudden increase in this measure indicates that the network over which the log messages are sent is experiencing slowdowns/network delays and noise. A high value for this measure is also indicative of potential data loss which is much more than the estimated Recovery Point Objective (RPO).
	Log send waits: Indicates the number of times *Flow Control* mode was initiated per second.	Waits/sec	A high value for this measure indicates that the network is congested and is experiencing slowdowns.
	Avg log send wait time: Indicates the average time the log messages should wait in the *Flow Control* mode.	Secs/Wait	This measure is a ratio of the *Log send wait time* and the *Log send waits* measures. A low value is desired for this measure.
	Alwayson messages resent: Indicates the number of Always on messages i.e., log stream messages that were resent over the network during the last measurement period.	Number	Ideally, the value of this measure should be low. A high value for this measure is a cause of concern as this indicates a high network latency or network congestion or network noise.

3.1.4 SQL AlwaysOn Page Repair Test

The AlwaysOn Availability Groups support automatic page repair. After certain types of errors corrupt a page on a local database, making it unreadable, an availability replica (primary or secondary) attempts to automatically recover the

page by resolving those errors that prevent reading the data in the page. If a secondary replica cannot read the page, the replica requests a fresh copy of the page from the primary replica. If the primary replica cannot read the page, the replica broadcasts a request for a fresh copy to all the secondary replicas and gets the page from the first replica that responds. If this request succeeds, the unreadable page is replaced by the copy, which usually resolves the error. If the database encounters too many errors simultaneously corrupting a large volume of pages, then the automatic page repair tries to resolve the errors. Certain types of pages such as File header page, Page 9 (the database boot page), the allocation pages etc cannot be repaired by the automatic page repair. If there are too many pages that cannot be repaired, then it indicates that the database is highly corrupted. If the administrators are warned proactively about the number of pages that cannot be repaired, then remedial action can be taken before serious issues occur! The **SQL AlwaysOn Page Repair** test helps administrators to proactively take remedial measures! For each message returned, this test reports the number of pages on which page repair was attempted. Using the detailed diagnosis of this test, administrators may be able to identify the pages on which page repair was attempted and the pages that cannot be repaired. Using this test, administrators can proactively identify the pages that are corrupted and take remedial measures before the availability databases become unavailable.

Purpose	For each message returned, this test reports the number of pages on which page repair was attempted
Target of the test	A Microsoft SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – While monitoring a Microsoft SQL Server 2012 and above, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for each message returned after page repair on the Microsoft SQL Server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Page count: Indicates the number of pages that returned this message after page repair was attempted.	Number	The detailed diagnosis of this measure if enabled, lists the DatabaseID, FileID, PageID, Error type and Page modification time.

3.1.5 SQL AlwaysOn Recovery Point Test

Recovery Point Objective (RPO) is defined as the amount of acceptable data loss or the point in time up to which the data can be recovered. Whenever a failover is detected, the administrators may want the secondary database to take over quickly from the primary database. If large quantity of data is not transferred to the secondary database from the primary database, then the users have to wait for a longer period to access the databases during failover. Often there would be a minimal data loss when a failover is in progress. This data loss may be due to the time lag that occurs during synchronization that happens between the primary and secondary databases. If the time taken is too long, it indicates that the synchronization process between the primary and secondary databases is taking too long to complete. This in turn will affect the users who will be compelled to wait for a prolonged time period to access the databases. To avoid such scenarios, it is essential to monitor the recovery point objective of the SQL server. The **SQL AlwaysOn Recovery Point** Test helps administrators in this regard. This test reports the amount of logs that had not been synchronized with the secondary database and the amount of hardened logs that are yet to be applied to the secondary database. In addition, this test helps administrators to analyze the time duration for which the log records were waiting in the redo queue before being rolled to the secondary database. This way, administrators may be proactively alerted to fine tune the time taken to roll the log to the secondary database so that the synchronization process completes in a quick and hassle free manner.

Purpose	Reports the amount of logs that had not been synchronized with the secondary database and the amount of hardened logs that are yet to be applied to the secondary database. In addition, this test helps administrators to analyze the time duration for which the log records were waiting in the redo queue before being rolled to the secondary database.
Target of the test	A Microsoft SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** –While monitoring a Microsoft SQL Server 2012 and above, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for each database on the Microsoft SQL server monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Log bytes flushed: Indicates the rate at which log bytes were flushed to the secondary database to complete synchronization since the last recovery point.	Flushes/sec	If the value of this measure is consistently increasing, then it indicates that the potential data loss can increase indefinitely.
	Log send queue size: Indicates the amount of log that had not been sent to the secondary database from this database to complete synchronization.	KB	Ideally, the value of this measure should be zero. A high value for this measure indicates that this much of data is unavailable in the secondary database during failover which directly implies that the customers would experience this data loss equal to this measure.
	Redo queue size: Indicates the total number of kilobytes of hardened log that currently remain to be applied to the secondary database to roll it forward.	KB	A low value is desired for this measure.
	Redo rate: Indicates the rate at which log records were rolled forward on the secondary database from this database.	KB/sec	
	Pending logs recovery time: Indicates the time duration for which the log records were waiting in the redo queue until being rolled forward to the secondary database.	Secs	Ideally, the value of this measure should be low.
	Pending logs flushed time: Indicates the time duration for which the logs were in the send queue until being flushed completely to the secondary database.	Secs	

3.1.6 SQL AlwaysOn Replica Status Test

An availability replica is an instantiation of an availability group that is hosted by a specific instance of SQL Server and maintains a local copy of each availability database that belongs to the availability group. There are two types of

availability replicas that exist in the availability group: single *primary replica* and one to eight *secondary replicas*.

An availability group fails over at the level of an availability replica. An availability replica provides redundancy only at the database level—for the set of databases in one availability group. Failovers are not caused by database issues such as a database becoming suspect due to a loss of a data file or corruption of a transaction log. The primary replica makes the primary databases available for read-write connections from clients. Also, in a process known as data synchronization, which occurs at the database level. The primary replica sends transaction log records of each primary database to every secondary database. Every secondary replica caches the transaction log records (hardens the log) and then applies them to its corresponding secondary database.

Whenever a failover is detected, the administrators may want the secondary replica to take over quickly from the primary replica. If the primary replica and secondary replicas are not in a position to apply the transaction logs to the primary and secondary databases, then there may be too much of non-sync between the primary replica and the secondary replicas during failover. In order to minimize such synchronization problems and maintain the secondary replicas on par with the primary replica, administrators are required to continuously monitor the operational state, synchronization status and synchronization health of the availability replicas. The **SQL AlwaysOn Replica Status** test helps administrators in this regard! This test continuously monitors the operational state, synchronization state and synchronization health of each availability replica of the SQL AlwaysOn Availability groups. In addition, administrators would be alerted to the current recovery state of the availability replica after a failover is initiated.

Purpose	Continuously monitors the operational state, synchronization state and synchronization health of each availability replica of the SQL AlwaysOn Availability groups. In addition, administrators would be alerted to the current recovery state of the availability replica after a failover is initiated
Target of the test	A Microsoft SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – While monitoring a Microsoft SQL Server 2012 and above, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

<table>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each availability replica on the Microsoft SQL server that is being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Operational state:
Indicates the current operational state of this availability replica.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>FAILED_NO_QUORUM</td><td>0</td></tr><tr><td>FAILED</td><td>1</td></tr><tr><td>OFFLINE</td><td>2</td></tr><tr><td>PENDING_FAILOVER</td><td>3</td></tr><tr><td>PENDING</td><td>4</td></tr><tr><td>ONLINE</td><td>5</td></tr></table>Note:
This measure reports the Measure Values listed in the table above to indicate the states of this availability replica. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.</td></tr>
<tr><td>Recovery state:
Indicates the current recovery state of this availability replica.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>ONLINE_IN_PROGRESS</td><td>0</td></tr><tr><td>ONLINE</td><td>1</td></tr></table>Note:
This measure reports the Measure Values listed in the table above to indicate the recovery states of this availability replica. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.</td></tr>
</table>

	Synchronization health state: Indicates the health of this availability replica during synchronization.		The values reported by this measure and their numeric equivalents are available in the table below: **Measure Value** / **Numeric Value** NOT_HEALTHY / 0 PARTIALLY_HEALTHY / 1 HEALTHY / 2 **Note:** This measure reports the **Measure Value**s listed in the table above to indicate the health status of the availability replica during synchronization. However, in the graph, this measure is indicated using the **Numeric Value**s listed in the above table.
	Connected state: Indicates the connection state of this availability replica with the primary/secondary availability replica.		The values reported by this measure and their numeric equivalents are available in the table below: **Measure Value** / **Numeric Value** DISCONNECTED / 0 CONNECTED / 1 **Note:** This measure reports the **Measure Value**s listed in the table above to indicate the connection state between the primary and secondary replicas. However, in the graph, this measure is indicated using the **Numeric Value**s listed in the above table. The Detailed diagnosis of this measure if enabled, lists the ReplicaID, IsLocal, Role, Last connect error number, Last connect error description, and the Last connect error time.

3.1.7 SQL AlwaysOn Replica Database Status Test

An availability database is a database that belongs to an availability group. For each availability database, the availability

group maintains a single read-write copy (the *primary database*) and one to eight read-only copies (*secondary databases*). Whenever a failover is detected, the administrators may want the secondary database to take over quickly from the primary database. If too much of data is not transferred to the secondary database from the primary database, then the users have to wait for a longer period to access the databases during failover. In order to avoid such delays, administrators are required to continuously monitor the synchronization status and synchronization health of the availability databases. For each availability database, this test reports the current state, synchronization state and synchronization health. In addition, administrators can be alerted to whether the availability database is suspended from the availability replica.

Purpose	Reports the current status, synchronization status and synchronization health of each availability database
Target of the test	A Microsoft SQL server
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – While monitoring a Microsoft SQL Server 2012 and above, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

<table>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each availability database on the target Microsoft SQL server that is being monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Is suspended?:
Indicates whether/not this availability database is suspended from the availability replica.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>No</td><td>0</td></tr><tr><td>Yes</td><td>1</td></tr></table>Note:
This measure reports the Measure Values listed in the table above to indicate whether this availability database is suspended or not. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.
The detailed diagnosis of this measure if enabled, lists the IsLocal and the reason for suspension of the availability database.</td></tr>
<tr><td></td><td>Synchronization state:
Indicates the current synchronization state of this availability database.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>NOT SYNCHRONIZING</td><td>0</td></tr><tr><td>INITIALIZING</td><td>1</td></tr><tr><td>REVERTING</td><td>2</td></tr><tr><td>SYNCHRONIZING</td><td>3</td></tr><tr><td>SYNCHRONIZed</td><td>4</td></tr></table>Note:
This measure reports the Measure Values listed in the table above to indicate the synchronization state of the database. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.</td></tr>
</table>

<table>
<tr>
<td></td>
<td>Synchronization health state:
Indicates the health of this availability database during synchronization.</td>
<td></td>
<td>The values reported by this measure and their numeric equivalents are available in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>NOT_HEALTHY</td><td>0</td></tr><tr><td>PARTIALLY_HEALTHY</td><td>1</td></tr><tr><td>HEALTHY</td><td>2</td></tr></table>Note:
This measure reports the Measure Values listed in the table above to indicate the health status of the availability database during synchronization. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.</td>
</tr>
<tr>
<td></td>
<td>Database state:
Indicates the current status of this availability database.</td>
<td></td>
<td>The values reported by this measure and their numeric equivalents are available in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>EMERGENCY</td><td>0</td></tr><tr><td>SUSPECT</td><td>1</td></tr><tr><td>RECOVERY_PENDING</td><td>2</td></tr><tr><td>RECOVERING</td><td>3</td></tr><tr><td>RESTORING</td><td>4</td></tr><tr><td>OFFLINE</td><td>5</td></tr><tr><td>ONLINE</td><td>6</td></tr></table>Note:
This measure reports the Measure Values listed in the table above to indicate the current status of this availability database. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.</td>
</tr>
</table>

3.1.8 Tests Disabled by Default

Tests related to database replication are disabled by default for the MS SQL server.

Replication is a set of technologies for copying and distributing data and database objects from one database to another and then synchronizing between databases to maintain consistency. Replication is typically performed to improve scalability and high availability of the database and for data warehousing and reporting purposes.

Transactional replication is the mechanism that Microsoft® SQL Server® provides to publish incremental data and schema changes to subscribers. The changes are published (the replication stream) in the order in which they occur, and typically there is low latency between the time the change is made on the Publisher and the time the change takes effect on the Subscriber. This enables a number of scenarios, such as scaling out a query workload or propagating data from a central office to remote offices and vice-versa. This form of replication always uses a hierarchical hub and spoke topology.

The following illustration is an overview of the components involved in transactional replication.

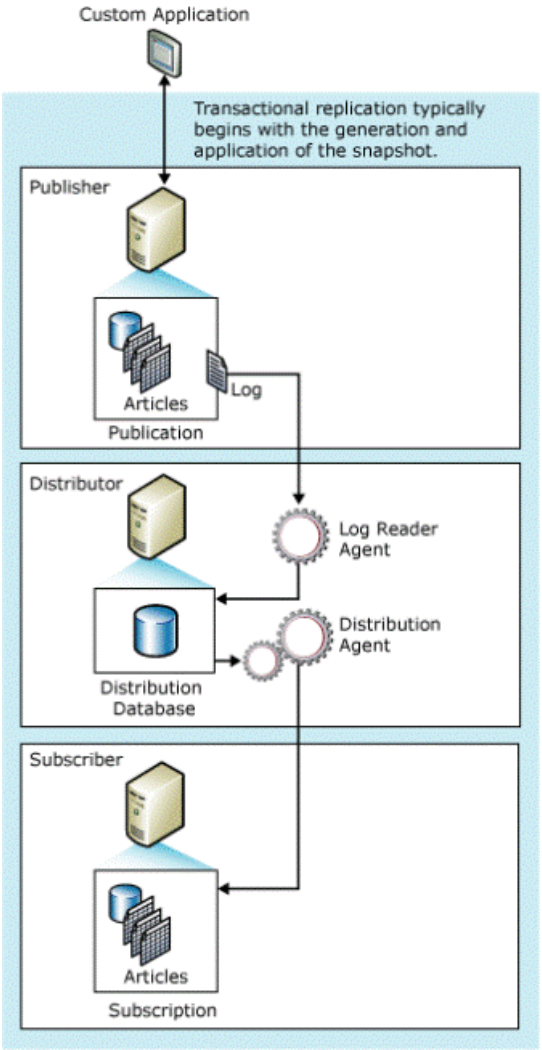


Figure 3.12: Architecture of Transactional replication

As is evident from Figure 3.12, a minimum of three server roles are required for transactional replication:

i. **Publisher**: A database server that makes data available for replication (source server) is referred to as the *publisher*; a collection of one or more database objects that are enabled for replication is called a *publication*.

j. **Distributor**: Replication is managed by the system database, which by default is called distribution. A distribution database—which can reside on the publisher, subscriber, or on a separate server—is created when you configure replication. The server that hosts the distribution database is referred to as the *distribution server* or *distributor*.

k. **Subscriber**: One or more servers that get data and/or transactions from the publisher are called *subscribers*.

Depending on the complexity of the replication topology, there may be multiple Subscriber servers. Furthermore, the roles of the various replication servers can be played by one server or by individual servers (the more common case), and it is possible for a server to play any combination of roles. Regardless, the various servers and databases must be protected to ensure that the replication stream is highly available.

Transactional replication relies on various agents to perform the tasks associated with tracking changes and distributing data. Soon after the Snapshot agent records information about the synchronization in the Distribution database, the Log Reader agent moves transactions marked for replication from the Publisher to the Distributor. These transactions are then moved to the Subscriber by the Distribution agent. If any changes are made on the Subscriber, then the Queue Reader agent moves these changes back to the Publisher.

Since replication saves the day by simplifying data recovery in the event of a database failure, care should be taken to ensure that the data on the Publisher and the Subscriber are always in sync. If one/more of the agents involved experience delays while discharging their duties, then the source and destination databases may remain out-of-sync for prolonged periods. At this juncture, if the source database becomes unavailable for any reason, the destination database cannot be used owing to the data non-sync, thereby beating the core purpose of replication! The eG Enterprise system introduces administrators to such slowdowns, much before users start complaining. Using the replication tests provided by the eG SQL Monitor, administrators can closely observe the operations of every type of agent discussed above, and can proactively capture potential latencies in their operations.

As stated earlier, these replication tests are disabled by default. To enable them, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose a 'replication test' from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

The sections that follow will discuss each test in detail.

3.1.8.1 Log Reader Replication Agent Test

The Replication Log Reader Agent is an executable that monitors the transaction log of each database configured for transactional replication and copies the transactions marked for replication from the transaction log into the distribution database. Each database published using transactional replication has its own Log Reader Agent that runs on the Distributor and connects to the Publisher.

If the replication process is found to take too long to complete, then, you can monitor the operations of the Log Reader agent to figure out whether/not this agent is contributing to the slowdown. This can be achieved with the help of the **Log Reader Application Agent** test. This test periodically monitors the activities of the Log Reader agent, and reveals how quickly the agent copies transactions to the Distributor; in the process, the test turns your attention to latencies (if any) in the agent's operations.

Purpose	Periodically monitors the activities of the Log Reader agent, and reveals how quickly the agent copies transactions to the Distributor; in the process, the test turns your attention to latencies (if any) in the agent's operations
Target of the test	An MS SQL server

<table>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. INSTANCE – The name of a specific MS SQL instance to be monitored. The default value of this parameter is “default”. To monitor an MS SQL instance named “CFS”, enter this as the value of the “instance” parameter.
5. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as “Not applicable’ by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every MS SQL server monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Commands Delivered:
Indicates the rate at which the Log Reader agent delivered commands to the Distributor.</td><td>Commands/Sec</td><td></td></tr>
<tr><td>Transactions Delivered:
Indicates the rate at which the Log Reader agent delivered transactions to the Distributor.</td><td>Trans/Sec</td><td>A high value is desired for this measure. If the value of this measure dips consistently over time, it is indicative of a slowdown.</td></tr>
</table>

	Latency: Indicates the current amount of time, in milliseconds, elapsed from when transactions are applied at the Publisher to when they are delivered to the Distributor.	MilliSec	Ideally, the value of this measure should be 0 or very low. A high value indicates that the Log Reader agent is taking too much time to copy transactions to the Distributor. The possible causes for this are as follows: • A large number of replicated transactions in the transaction log; • A large number of non-replication transactions; • A large number of virtual log files (VLFs) • Slow disk subsystem • Unexpected network I/O problems • Blocking by another SQL Replication Agent such as a the Distribution Cleanup Agent

3.1.8.2 Snapshot Replication Agent Test

The Snapshot Agent run at the Distributor and is typically used with all types of replication. It prepares schema and initial data files of published tables and other objects, stores the snapshot files, and records information about synchronization in the distribution database.

The first step towards troubleshooting a delay in database replication is to figure out at which step of the replication process the delay occurred, and which agent performed that step. Using the **Snapshot Replication Agent** test, you can determine whether/not the Snapshot agent is contributing to a slowdown (if any) in the replication process. This test, at pre-configured intervals, monitors how quickly the agent delivers transactions to the distribution database. Latencies in creation of snapshot files or in recording synchronization details in the Distributor are thus revealed.

Purpose	Monitors how quickly the agent delivers transactions to the distribution database; latencies in creation of snapshot files or in recording synchronization details in the Distributor are thus revealed
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **INSTANCE** – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter. 5. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every MS SQL server monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Commands Delivered: Indicates the rate at which the Snapshot agent delivered commands to the Distributor.	Commands/Sec	
	Transactions Delivered: Indicates the rate at which the Snapshot agent delivered transactions to the Distributor.	Trans/Sec	A high value is desired for this measure. If the value of this measure dips consistently over time, it is indicative of a slowdown. A contention for memory / CPU resources on the SQL server, or high disk I/O on the server could cause snapshotting to slow down.

3.1.8.3 Distribution Replication Agent Test

The Distribution Agent is used with snapshot replication and transactional replication. It applies the initial snapshot to the Subscriber and moves transactions held in the distribution database to Subscribers. The Distribution Agent runs at either the Distributor for push subscriptions or at the Subscriber for pull subscriptions.

If replication is taking longer than usual, you may want to check how each of the agents involved in the replication process is performing, so that you can isolate the agent that could be causing the delay. The **Distribution Replication Agent** test helps you run frequent health checks on the Distribution agent and reveals whether latencies (if any) in the operations of this agent are the root-cause for the replication slowdown.

Purpose	Helps you run frequent health checks on the Distribution agent and reveals whether latencies (if any) in the operations of this agent are the root-cause for the replication slowdown
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. INSTANCE – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter.
5. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every MS SQL server monitored</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Commands Delivered:
Indicates the rate at which the Distribution agent delivered commands to the Subscriber.</td><td>Commands/Sec</td><td></td></tr>
<tr><td>Transactions Delivered:
Indicates the rate at which the Distribution agent delivered transactions to the Subscriber.</td><td>Trans/Sec</td><td>A high value is desired for this measure. If the value of this measure dips consistently over time, it is indicative of a slowdown.</td></tr>
<tr><td>Latency:
Indicates the current amount of time, in milliseconds, elapsed from when transactions are delivered to the Distributor to when they are applied at the Subscriber.</td><td>MilliSec</td><td>Ideally, the value of this measure should be 0 or very low. A high value indicates that the Distribution agent is taking too much time to push transactions to the Subscriber.
The possible causes for this are as follows:
• A series of transactions could be trying to move a large batch of commands to the Subscribers
• Blocking of the Distribution Agent by another Replication Agent
• Long execution times in the INS/UPD/DEL stored procedures used to apply transactions to the subscriber
• SQL statements not being replicated as 'parameters'</td></tr>
</table>

3.1.8.4 Merge Replication Test

The Merge Agent is used with merge replication. It applies the initial snapshot to the Subscriber and moves and reconciles incremental data changes that occur. Each merge subscription has its own Merge Agent that connects to both the Publisher and the Subscriber and updates both. The Merge Agent runs at either the Distributor for push subscriptions or the Subscriber for pull subscriptions. By default, the Merge Agent uploads changes from the Subscriber to the Publisher and then downloads changes from the Publisher to the Subscriber.

The speed with which the Merge agent uploads/downloads changes between the Publisher and Subscriber and the count of conflicts it detects in the process influence how efficient the Merge agent is and how quickly data replication occurs. If data replication stalls, you may want to check the performance of the Merge agent to figure out possible reasons for the slowdown. The **Merge Replication** test enables this analysis. This test monitors the operations of the Merge agent and reports how quickly changes were uploaded and downloaded by the agent and how many conflicts were detected (per second) while merging. These statistics are useful when trying to determine whether/not issues with the Merge agent are causing problems in replication.

<table>
<tr><td>Purpose</td><td colspan="3">Monitors the operations of the Merge agent and reports how quickly changes were uploaded and downloaded by the agent and how many conflicts were detected while merging</td></tr>
<tr><td>Target of the test</td><td colspan="3">An MS SQL server</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. INSTANCE – The name of a specific MS SQL instance to be monitored. The default value of this parameter is “default”. To monitor an MS SQL instance named “CFS”, enter this as the value of the “instance” parameter.
5. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as “Not applicable’ by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every MS SQL server monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Downloaded changes:
Indicates the number of rows per second replicated from the Publisher to the Subscriber.</td><td>Rows/Sec</td><td rowspan="2">A high value is desired for these measures. If the value of these measures dips consistently over time, it could indicate a slowdown.</td></tr>
<tr><td>Uploaded changes:
Indicates the number of rows per second replicated from the Subscriber to the Publisher.</td><td>Rows/Sec</td></tr>
</table>

	Conflicts: Indicates the number of conflicts per second occurring during the merge process. .	Conflicts/Sec	Merge replication allows multiple nodes to make autonomous data changes, so situations exist in which a change made at one node may conflict with a change made to the same data at another node. In other situations, the Merge Agent encounters an error such as a constraint violation and cannot propagate a change made at a particular node to another node. Ideally, the rate of such conflicts should be low. A high value or a steady increase in this value indicates that too many conflicts have been detected, but only a few have been resolved. A large number of merge conflicts and delays in their resolution can adversely impact replication. The Merge Agent detects conflicts by using the **lineage** column of the **MSmerge_contents** system table; if column-level tracking is enabled for an article, the **COLV1** column is also used. These columns contain metadata about when a row or column is inserted or updated, and about which nodes in a merge replication topology made changes to the row or column. As the Merge Agent enumerates changes to be applied during synchronization, it compares the metadata for each row at the Publisher and Subscriber. The Merge Agent uses this metadata to determine if a row or column has changed at more than one node in the topology, which indicates a potential conflict. After a conflict is detected, the Merge Agent launches the conflict resolver specified for the article with a conflict and uses the resolver to determine the conflict winner. The winning row is applied at the Publisher and Subscriber, and the data from the losing row is written to a conflict table.

			Conflicts are resolved automatically and immediately by the Merge Agent unless you have chosen interactive conflict resolution for the article.

3.1.8.5 Replication Agents Test

This test reports the number of replication agents that are currently running on a target MS SQL server.

Purpose	Reports the number of replication agents that are currently running on a target MS SQL server		
Target of the test	An MS SQL server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **INSTANCE** – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter. 5. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every MS SQL server monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Running Agents: Indicates the number of replication agents currently running..	Number	

3.2 The MS SQL Memory Structures Layer

This layer tracks the health of the memory and buffer structures of an MS SQL server tests shown in Figure 3.13. The details of the tests are available in the following sections.

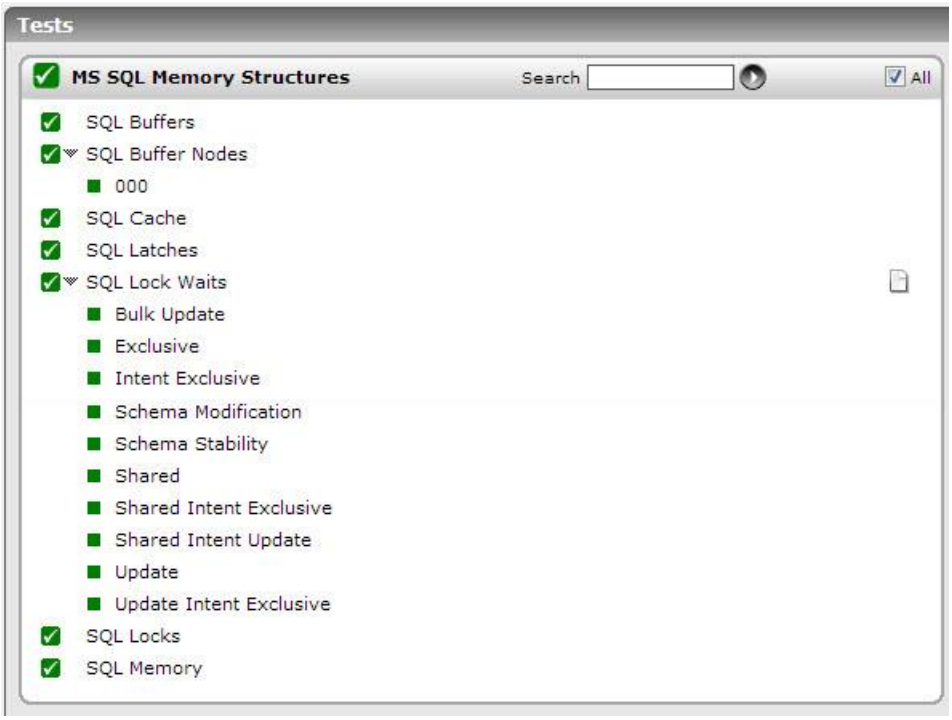


Figure 3.13: Tests pertaining to the MS SQL Memory Structures layer

3.2.1 SQL Memory Test

This test monitors the memory usage of an MS SQL server.

Purpose	This test measures the memory usage of a MS SQL server
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
4. HOST – The IP address of the MS SQL server.
5. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
6. INSTANCE – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter.
7. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
8. USEPERFMON – By default, this flag is set to Yes, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to No. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to No.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every MS SQL server monitored</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total server memory:
This value indicates the total amount of memory that is being currently used by the server.</td><td>MB</td><td>An unusually large usage of memory by the server is a cause of concern. Further analysis is required to determine if specific users or queries are consuming excess memory.</td></tr>
<tr><td>Target server memory:
This value indicates the total amount of dynamic memory, which the server can consume.</td><td>MB</td><td>If, over time, the Total server memory measure is less than the Target server memory counter, then this means that SQL server has enough memory to run efficiently. On the other hand, if the Total server memory measure is greater than the Target server memory counter, this indicates that SQL Server may be under memory pressure and could use access to more physical memory.</td></tr>
<tr><td>Sql cache memory:
This value indicates the total amount of dynamic memory that the server is using for the dynamic SQL cache.</td><td>MB</td><td>The amount of data cache available to SQL Server can significantly affect SQL Server's performance. If the dynamic SQL cache memory usage is low, consider tuning the cache management parameters of SQL server.</td></tr>
</table>

	Optimizer memory: This value indicates the total amount of dynamic memory, which the server is using for query optimization.	MB	If the optimizer memory usage is low, consider tuning the optimizer memory management parameters of the MS SQL server.
	Max workspace memory: This value indicates the maximum amount of memory allocated for the execution of processes. This memory is used primarily for operations like hash, sort and create index.	MB	This parameter is useful in conjunction with the grant workspace memory. When the grant workspace memory reaches the max workspace memory then we should consider tuning this.
	Lock memory: This value indicates the total amount of dynamic memory, which the server has allocated for locks.	MB	If the memory allocated for locks is less and there is a contention/wait for a lock, try tuning the lock memory management parameters of the MS SQL server
	Grant workspace memory: This value indicates the total amount of memory granted for the execution of processes. This memory is used for hash, sort and create index operations.	MB	If the grant workspace memory is nearing the maximum workspace memory then the maximum workspace memory may have to be increased.
	Connection memory: This value indicates the total amount of dynamic memory, which the server is using for maintaining connections.	MB	If the memory allocated for connection is less, try tuning the memory management parameters of the MS SQL server.
	Memory grants pending: Indicates the total number of processes waiting for a workspace memory grant.	Number	In general, if you have any processes queuing waiting for memory, you should expect degraded performance. The ideal situation for a healthy server is no outstanding memory grants – i.e., the value of this measure should ideally be 0.

3.2.2 SQL Locks Test

This test monitors the locking activity of various transactions supported by an MS SQL server.

Purpose	This test measures statistics pertaining to locking activity of an MS SQL server
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. INSTANCE – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter.
5. USEPERFMON – By default, this flag is set to Yes, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to No. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to No.
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 enabled/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.
7. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every MS SQL server monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Avg wait time for locks:
This gives the average amount of wait time for each lock request that resulted in a wait.</td><td>Secs</td><td>A high value may indicate that there is contention for locks in the system. When the average wait time for locks is high, users may have to wait for their transactions to complete. This metric reports the average wait time for a variety of locks, including database, extent, key, page, RID, and table. When the average wait time is high, use the SQL profiler to identify which lock(s) is causing transaction delays.
The detailed diagnosis of the Avg wait time for locks measure, if enabled, provides the average wait time for each lock type.</td></tr>
</table>

	Lock requests: This value gives the number of new locks and lock conversions requested from the lock manager per second.	Reqs/Sec	A high value indicates that there is high locking activity in the system and may need close scrutiny for the type of locks being requested.
	Lock waits: This value indicates the number of lock requests per second that could not be satisfied immediately and required the caller to wait before being granted the lock.	Waits/Sec	A high value of waits can have an adverse impact on application performance. Possible reasons for this behavior could be: ➢ inadequate number of locks available in the database, ➢ unusually high locking behavior of applications accessing the database, ➢ improper database application design, etc.
	Deadlocks: Number of lock requests/sec that resulted in a deadlock	Deadlocks/Sec	A deadlock may arise due to various situations including bad design of queries and deficient coding practices. A deadlock is a situation where both/all the lock requestors are in a mutual or a multi-way tie. Any deadlocks are detrimental to database application performance.
	Lock timeouts: indicates the number of lock requests per second that timed out, including requests for NOWAIT locks.	Timeouts/Sec	The LOCK_TIMEOUT setting allows an application to set a maximum time that a statement waits on a blocked resource. When a statement has waited longer than the LOCK_TIMEOUT setting, the blocked statement is canceled automatically, and error message 1222 (`Lock request time-out period exceeded`) is returned to the application. Any transaction containing the statement, however, is not rolled back. A high value for this measure indicates that many statements were cancelled, as they could not acquire a lock on a resource for a long time. This may be due to another extent lock holding the resource (An extent lock is the one which locks multiple resources). In such a situation, use the detailed diagnosis of the **SQL Blocker Processes** test to know which statements have been blocked for a long time, and which statements are blocking them.

The detailed diagnosis of the *Lock requests* measure, if enabled, provides the rate of locks for each lock type (see Figure 3.14).

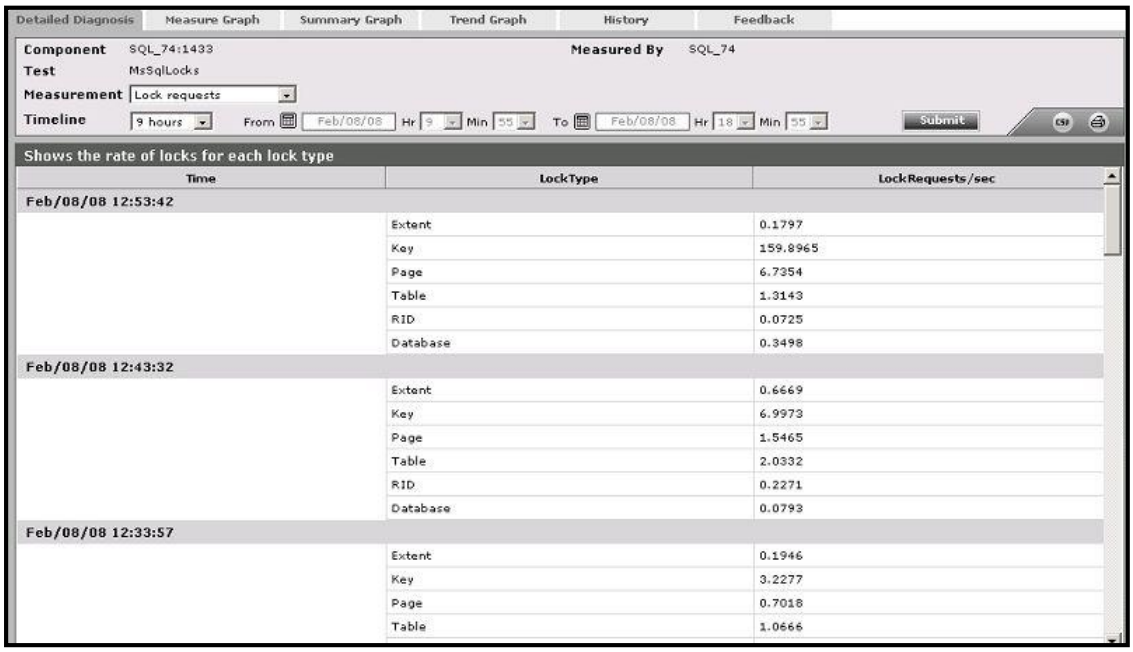


Figure 3.14: The detailed diagnosis of the Lock requests measure

The detailed diagnosis of the *Lock waits* measure, if enabled, displays the rate of lock waits for each lock type (see Figure 3.15).

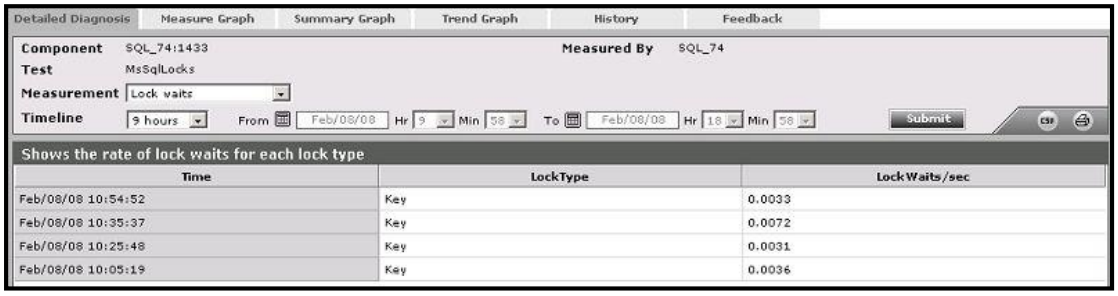


Figure 3.15: The detailed diagnosis of the Lock waits measure

3.2.3 SQL Latches Test

A latch is a "lightweight lock". A latch acts like a lock, in that its purpose is to prevent data from changing unexpectedly. For example, when a row of data is being moved from the buffer to the SQL Server storage engine, a latch is used by SQL Server during this move to prevent the data in the row from being changed during this very short time period. This not only applies to rows of data, but to index information as well, as it is retrieved by SQL Server.

Just like a lock, a latch can prevent SQL Server from accessing rows in a database, which can hurt performance. Because of this, latch wait time must be minimized.

The SQL Latches test makes the following measures of latch activity for an MS SQL database.

Purpose	Measures the latch activity for an MS SQL database
Target of the test	An MS SQL server

<table>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. INSTANCE – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter.
5. USEPERFMON – By default, this flag is set to Yes, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to No. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to No.
6. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every MS SQL server monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Latch wait rate:
This is the number of latch requests per second that could not be granted immediately.</td><td>Waits/Sec</td><td>The latch activity may vary from one server to another. Hence, it is important to get baseline numbers for this metric, so that you can compare "typical" latch activity against what is happening currently. If latch activity is higher than expected, this often indicates one of two problems. First, it may mean that the SQL Server could use more memory. If latch activity is high, check the buffer cache hit ratio is. If it is below 99%, the SQL server could probably benefit from more memory. If the hit ratio is above 99%, then it could be the I/O system that is contributing to the problem, and a faster I/O system might improve server's performance.</td></tr>
<tr><td>Total latch wait time:
This metric denotes the total wait time for latch requests that had to wait in the last second.</td><td>Secs</td><td>Ideally, this value should be close to 0. The larger this value is, the more contention there is for latches and worse the performance of the database. If the wait time is high, check the SYSPROCESSES table of the database to see which latches are seeing most contention.</td></tr>
</table>

3.2.4 SQL Cache Test

This internal test monitors the usage of buffer memory of an MS SQL server.

Purpose	This test measures usage of buffer of an MS SQL server		
Target of the test	An MS SQL server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **INSTANCE** – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter. 5. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**. 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 enabled/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. 9. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every MS SQL server monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Cache hit ratio:** This value indicates the percentage of pages found in the cache without having to read from disk.	Percent	The ratio is the total number of cache hits divided by the total number of cache lookups since SQL Server was started. Because reading from the cache is less expensive than reading from disk, you want the ratio to be high. The higher this value is, the better. Generally, you can increase the cache hit ratio by increasing the amount of memory available to the SQL Server.
	Objects in cache: This value indicates the number of objects found in the cache currently.	Number	The higher the count, the better the performance of the server, as it does not need to read the requested object from disk.
	Log cache hit ratio: This value indicates the percentage of log cache reads satisfied from the log cache.	Percent	A good cache hit ratio indicates that the log cache is performing well and a low value indicates that the server needs to be tuned.

The detailed diagnosis of the *Cache hit ratio* measure, if enabled, provides the cache hit ratio for each cache type (see Figure 3.16). A high cache hit ratio is an indicator of the SQL server's good health. The information provided by Figure 3.16 will therefore reveal the cache types that were effective and those that were not.

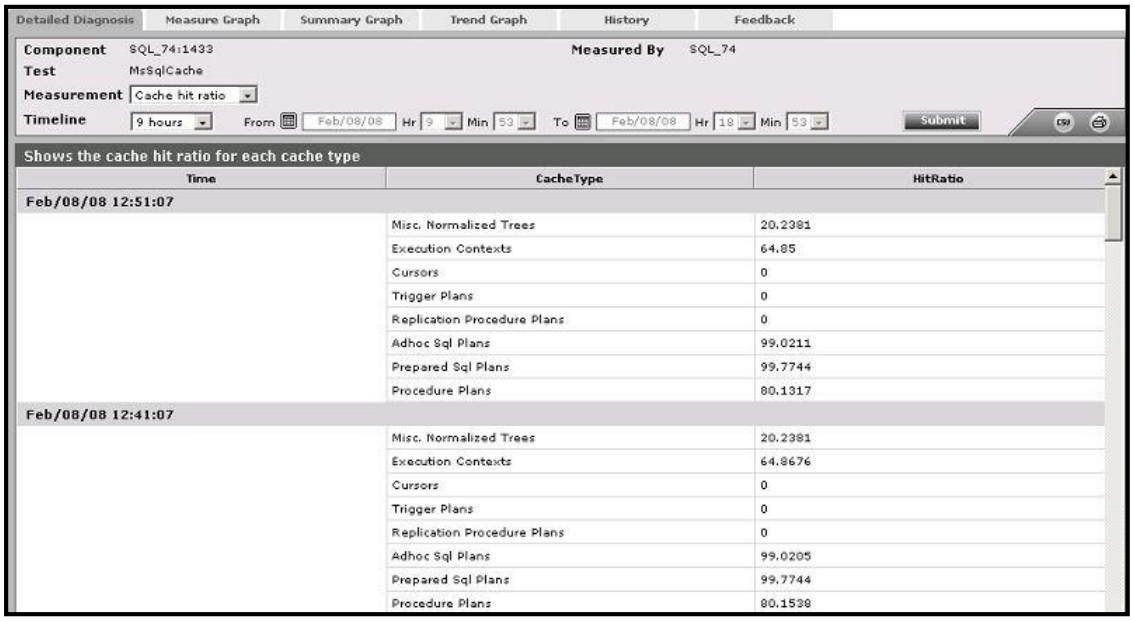


Figure 3.16: The detailed diagnosis of the Cache hit ratio measure

The detailed diagnosis of the *Objects in cache* measure , if enabled, provides the number of objects available in each cache type (see Figure 3.17). This again is an indicator of the effectiveness of the cache types.

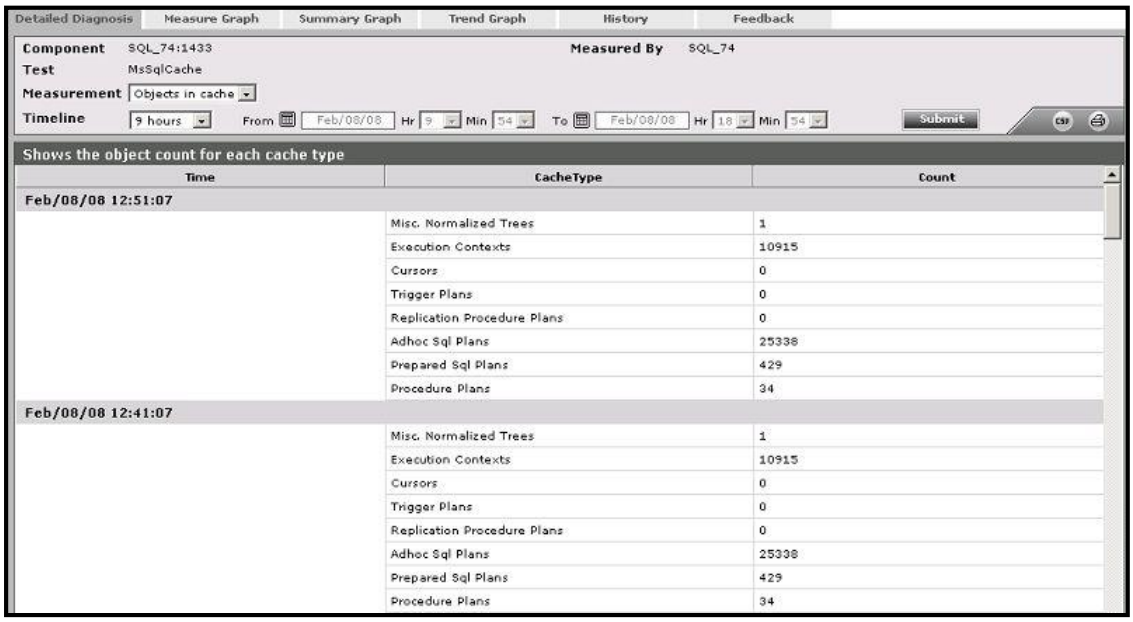


Figure 3.17: The detailed diagnosis of the Objects in cache measure

3.2.5 SQL Buffers Test

This internal test also monitors the usage of buffer memory of an MS SQL server.

Purpose	This test measures usage of buffer of an MS SQL server
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **INSTANCE** – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter. 5. **SSL** - By default, the **SSL** flag is set to **No**, indicating that the target MS SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled MS SQL server, set the **SSL** flag to **Yes**. 6. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**. 7. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every MS SQL server monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Buffer cache hit ratio: This value indicates the percentage of pages that are served from the server's buffer cache (i.e., without requiring a read from the disk).	Percent	This value is the ratio of the total number of cache hits to the total number of cache lookups since the server was started. Because reading from the cache is much less expensive than reading from disk, this ratio should ideally be high. Generally, one can increase the buffer cache hit ratio by increasing the amount of memory available to SQL Server.
	Page reads: This indicates the number of database pages that were physically read per second.	Pages/Sec	Because physical I/O is expensive, one may be able to minimize the cost, either by using a larger data cache, intelligent indexes, more efficient queries, or by changing the database design.
	Page writes: This indicates the number of database page writes that are issued per second.	Writes/Sec	Page writes are generally expensive. Reducing page-write activity is important for optimal tuning. One way to do this is to ensure that you do not run out of free buffers in the free buffer pool. If you do, page writes will occur while waiting for an unused cache buffer to flush.
	Target Pages: The ideal number of pages that should be in the buffer pool for optimal performance	Number	
	Total pages: The current total number of pages in the buffer pool. This value includes the number of pages in the database pool, free pool, and stolen list.	Number	For optimum performance, the total pages must be close to the target pages value. **This measure will not be available for Microsoft SQL Server 2012.**
	Free pages: The total number of pages currently in all free lists	Number	**This measure will not be available for Microsoft SQL Server 2012.**
	Stolen pages: The number of buffer pool pages used to satisfy other server memory requests	Number	**This measure will not be available for Microsoft SQL Server 2012.**

	Lazy writes rate: Indicates the number of lazy writes per second.	Writes/Sec	*Lazy writes* are buffers written by the lazy writer. The lazy writer is a system process that flushes out batches of dirty, aged buffers (buffers that contain changes that must be written back to disk before the buffer can be reused for a different page) and makes them available to user processes. The lazy writer eliminates the need to perform frequent checkpoints in order to create available buffers. Keeping the number of lazy writes low can enhance performance. A supply of buffers available for immediate use keeps the number of lazy writes low. Before a requested page can be brought into memory, a free buffer must be available in the buffer pool. If no free buffers are available, an existing buffer must be reused. When an existing buffer has to be reused, many buffer pages might have to be searched in order to locate a buffer to reclaim for use. If the buffer found is marked as dirty or modified, the buffer manager must first write the changes to disk before the page can be reused and assigned to the requesting process. This results in a wait for the requesting process. Waiting processes can degrade performance.
	Page life expectancy: Indicates the number of seconds a page will stay in the buffer pool without references.	Secs	A high value of this measure indicates that your page stays longer in the buffer pool (area of the memory cache), thereby leading to higher performance. This is because, every time a request comes in, the probability of that request been served by the cache is higher; this in turn significantly reduces direct disk accesses, which are relatively more expensive operations. A value too low for your system indicates that pages are being flushed from the buffer pool too quickly. The target on an OLTP system should be at least 300 seconds (5min). When under 300 seconds, this may indicate poor index design (leading to increased disk I/O and less effective use of memory) or, simply, a potential shortage of memory.

	Free list stalls: Indicates the number of requests per second that had to wait for a free page.	Stalls/Sec	Free list stall rates of 3 or 4 per second indicate too little SQL memory available.

3.2.6 SQL Buffer Nodes Test

This test is specific to MSSQL Server 2005 (or above).

Microsoft SQL Server 2005 (or above) is non-uniform memory access (NUMA) aware. As clock speed and the number of processors increase, it becomes increasingly difficult to reduce the memory latency required to use this additional processing power. NUMA architecture provides a scalable solution to this problem. SQL Server 2005 (or above) has been designed to take advantage of NUMA-based computers without requiring any application changes. This database server groups schedulers to map to the grouping of CPUs, based on the hardware NUMA boundary exposed by Windows. For example, a 16-way box may have 4 NUMA nodes, each node having 4 CPUs. This allows for a greater memory locality for that group of schedulers when tasks are processed on the node. With SQL Server 2005 (or above) you can further subdivide CPUs associated with a hardware NUMA node into multiple CPU nodes. This is known as soft-NUMA. There is one SQL Server memory node for each physical NUMA node. When a thread running on a specific hardware NUMA node allocates memory, the memory manager of SQL Server tries to allocate memory from the memory associated with the NUMA node for locality of reference. Similarly, buffer pool pages are distributed across hardware NUMA nodes. It is more efficient for a thread to access memory from a buffer page that is allocated on the local memory than to access it from foreign memory.

To understand the local vs. foreign memory distribution in SQL Server better, assume that the computer has 16 gigabytes (GB) of memory. Other applications including Windows have consumed some of the memory from each node, SQL Server has assigned some memory for its processes outside of the buffer pool, and SQL Server has 10 GB of memory to assign to the buffer pool. The buffer pool memory is divided among four physical NUMA nodes, N0, N1, N2, and N3, each with the following local memory available:

- N0 – 1 GB
- N1 – 3 GB
- N2 – 3 GB
- N3 – 3 GB

In the above configuration, all nodes will eventually allocate and use 2.5 GB of memory; however, node N0 will end up with 1.0 GB of its own memory, and 1.5 GB of memory from other nodes.

The SQL Buffer Nodes test reports information about buffer pool page distribution for each NUMA node on MSSQL Server 2005 (or above).

Purpose	Reports information about buffer pool page distribution for each NUMA node on MS SQL Server 2005 (or above)
Target of the test	An MS SQL server 2005 (or above)
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 5. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 6. **SSL** - By default, the **SSL** flag is set to **No**, indicating that the target MS SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled MS SQL server, set the **SSL** flag to **Yes**. 7. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.		
Outputs of the test	One set of results for every NUMA node on the MS SQL Server 2005 (or above)		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Database pages: Indicates the number of pages that are currently in the buffer pool with database content.	Number	
	Foreign pages: Indicates the number of pages that were currently received from other NUMA nodes.	Number	If the SQL server is running on non-NUMA hardware, then the value of this measure will be 0. Foreign pages will not be used during ramp-up because they can frequently be transferred to the owning node and become local to that node. When the value of **max server memory** is reached, some nodes may have foreign memory, but once the memory target is achieved, the buffer pool will treat local and foreign memory identically. For example, under memory pressure, the buffer pool will not make any effort to free up foreign memory pages before local memory pages. If the value of this measure increases consistently, it indicates that local memory on the node is inadequate.

	Free pages: Indicates the total number of pages that are currently free on this NUMA node.	Number	If the value of this measure dips consistently, it is indicative of insufficient memory on the node. You might want to consider resizing the buffer pool in this case.
	Page life expectancy: Indicates the number of seconds a page currently stayed in the buffer pool without references.	Secs	
	Stolen pages: Indicates the number of pages that were currently used for miscellaneous server purposes (stolen from the buffer pool) on this node.	Number	
	Target pages: Indicates the ideal number of pages in the buffer pool on this node.	Number	
	Total pages: Indicates the total number of committed pages that currently exist in the buffer pool on this node.	Number	

3.2.7 SQL Lock Waits Test

lock wait event occurs when a user requests for a resource that is already locked by another user, forcing the former to wait until the latter releases the lock. Lock wait events on a database need to be minimal. If a lock is held on a resource for too long a time, all other requests will be denied access to that resource, thereby causing critical operations to fail. Moreover, if the number of lock waits grows over time, it will consequently increase the length of the pending requests queue; a long request queue may not only cause the unnecessary erosion of valuable server resources, it may also choke the database server, thereby significantly impacting the quality of the user experience with the server. It is therefore imperative that the lock wait events are monitored, and issues related to such events immediately brought to the attention of administrators.

This test monitors the lock wait events for each lock type, and promptly alerts administrators to a sudden/steady increase in the number and duration of such events. The lock types which will form the descriptors of this test are discussed below:

Descriptor	Lock Type	Description
S	Shared locks	Shared locks are held on data being read under the pessimistic concurrency model. While a shared lock is being held other transactions can read but cannot modify locked data. After the locked data has been read the shared lock is released, unless the transaction is

		being run with the locking hint (READCOMMITTED, READCOMMITTEDLOCK) or under the isolation level equal or more restrictive than Repeatable Read.
U	Update locks	Update locks are a mix of shared and exclusive locks. When a DML statement is executed SQL Server has to find the data it wants to modify first, so to avoid lock conversion deadlocks an update lock is used. Only one update lock can be held on the data at one time, similar to an exclusive lock. But the difference here is that the update lock itself can't modify the underlying data. It has to be converted to an exclusive lock before the modification takes place.
X	Exclusive locks	Exclusive locks are used to lock data being modified by one transaction thus preventing modifications by other concurrent transactions. You can read data held by exclusive lock only by specifying a NOLOCK hint or using a read uncommitted isolation level. Because DML statements first need to read the data they want to modify you'll always find Exclusive locks accompanied by shared locks on that same data.
I	Intent locks	Intent locks are a means in which a transaction notifies other transaction that it is intending to lock the data. Thus the name. Their purpose is to assure proper data modification by preventing other transactions to acquire a lock on the object higher in lock hierarchy. What this means is that before you obtain a lock on the page or the row level an intent lock is set on the table. This prevents other transactions from putting exclusive locks on the table that would try to cancel the row/page lock.
Sch	Schema locks	There are two types of schema locks: • Schema stability lock (Sch-S): Used while generating execution plans. These locks don't block access to the object data. • Schema modification lock (Sch-M): Used while executing a DDL statement. Blocks access to the object data since its structure is being changed.
SIX	Shared with Intent Exclusive	A transaction that holds a Shared lock also has some pages/rows locked with an Exclusive lock
SIU	Shared with Intent Update	A transaction that holds a Shared lock also has some pages/rows locked with an Update lock
UIX	Update with Intent Exclusive	A transaction that holds an Update lock also has some pages/rows locked with an Exclusive lock

This test is applicable only to Microsoft SQL Server 2005 (and above).

Purpose	Monitors the lock wait events for each lock type, and promptly alerts administrators to a sudden/steady increase in the number and duration of such events

Target of the test	An MS SQL server 2005 (and above)
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. SSL – If the MS SQL server being monitored is an SSL-enabled server, then set the SSL flag to Yes. If not, then set the SSL flag to No.
5. INSTANCE - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the INSTANCE parameter.
6. USER – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the Sysadmin role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE, and PUBLIC roles in this text box.
7. PASSWORD - The password of the specified USER
8. CONFIRM PASSWORD - Confirm the password by retyping it.
9. DOMAIN - By default, none is displayed in the DOMAIN text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the DOMAIN can continue to be none. On the other hand, if 'Windows only' authentication has been enabled, then, in the DOMAIN text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the USER name and PASSWORD that you provide should be that of a user authorized to access the monitored SQL server.
10. ISNTLMV2 - In some Windows networks, NTLM (NT LAN Manager) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the ISNTLMV2 flag is set to No, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to Yes if NTLMv2 is enabled on the target host.
11. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
12. 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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for lock wait type on the MS SQL server monitored</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

Measurements made by the test	**Current lock waits:** Indicates the current number of lock wait events for this lock type.	Number	Ideally, this value should be very low. A high value or a consistent increase in the value may choke the database server and severely hamper its overall performance. Therefore, if the value of this measure is high, you might first need to identify what is causing the lock waits. For this purpose, you can use the detailed diagnosis of this measure. The detailed diagnosis leads you to the exact object the lock events are waiting on, the user who holds a lock on that object, and the query that initiated the lock. This way, inefficient queries can be identified and fine-tuned. Given below are some tips for minimizing lock waits: • Keep all Transact-SQL transactions as short as possible. • To reduce the amount of time that tables are locked, which hurts concurrency and performance, avoid interleaving reads and database changes within the same transaction. Instead, try to do all your reads first, then perform all of the database changes (UPDATES, INSERTS, DELETES) near the end of the transaction. This helps to minimize the amount of time that exclusive locks are held. • Use clustered indexes on heavily used tables. • Make appropriate use of non-clustered indexes • Try to avoid Transact-SQL statements that affect large numbers of rows at once, especially the INSERT and UPDATE statements.

			• Try to have your UPDATE and DELETE statements use an index. • When using nested transactions, avoid commit and rollback conflicts.
	Avg. wait time for locks: Indicates the duration of the lock wait events for this lock type.	Milliseconds	Ideally, this value should be very low. A high value or a consistent increase in the value may choke the database server and severely hamper its overall performance. Therefore, if the value of this measure is high, you might first need to identify what is causing the lock waits. For this purpose, you can use the detailed diagnosis of the *Current lock waits* measure. The detailed diagnosis leads you to the exact object the lock events are waiting on, the user who holds a lock on that object, and the query that initiated the lock. This way, inefficient queries can be identified and fine-tuned.

3.3 The MS SQL Workload Layer

The tests mapped to this layer help analyze the query, transaction, and process workload on the SQL server.

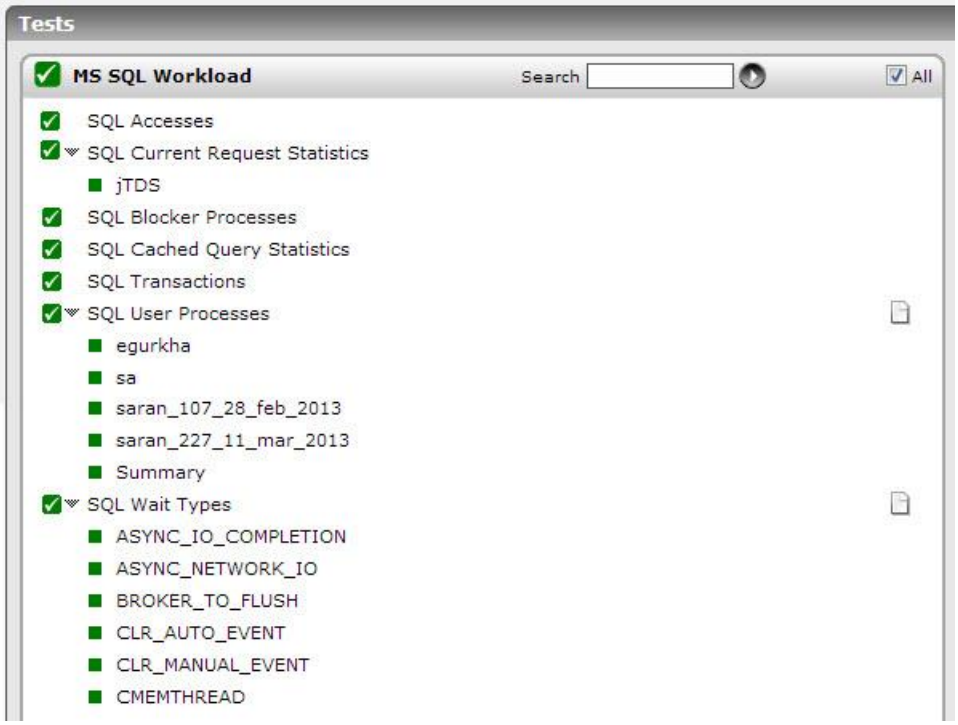


Figure 3.18: The tests mapped to the MS SQL Workload layer

3.3.1 SQL Accesses Test

This test monitors various critical metrics regarding accesses to the MS SQL database.

Purpose	Monitors various critical metrics regarding accesses to the MS SQL database
Target of the test	An MS SQL server
Agent deploying the test	An Internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. INSTANCE – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter.
5. SSL - By default, the SSL flag is set to No, indicating that the target MS SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled MS SQL server, set the SSL flag to Yes.
6. USEPERFMON – By default, this flag is set to Yes, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to No. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to No.
7. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every MS SQL server monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Full table scans:
This metric represents the number of unrestricted full scans being handled by the server. The scans can either be base table or full index scans and the value represents the sum of all scans for all the database instances being handled by the server.</td><td>Scans/Sec</td><td>Generally it is better to have fewer table scans for a database. In many cases, the SQL server itself will perform a few full table scans on a regular basis for internal use. To detect anomalies, check for random full table scans that may represent the behavior of applications using the database server. If an unusually high number of full table scans are noticed, use the Profiler and Index Tuning Wizard to determine what is causing these scans, and if adding any indexes can help reduce the table scans.</td></tr>
<tr><td>Page splits:
This value represents the rate of page splits occurring in a database server as the result of index pages overflowing.</td><td>Splits/Sec</td><td>If the rate of page splits is high, consider increasing the fill factor of the indexes.</td></tr>
</table>

	Sql compilations: This value is the rate of SQL compilations happening in the database.	Reqs/Sec	SQL compilations are a normal part of an SQL Server's operation. But because compilations take up CPU and other resources, the SQL server attempts to reuse as many execution plans in cache as possible (execution plans are created when compilations occur). The more that execution plans are reused, the less overhead there is on the server, and the faster overall performance is. A high number of SQL compilations indicates that the server is very busy. Compilations can be forced if object schema changes, if previously parallelized execution plans have to run serially, if statistics are recomputed, or if a number of other things occur. In some cases, it is possible to reduce the number of unnecessary compilations.
	Sql recompilations: This value is the rate of SQL recompilations happening in the database server.	Reqs/Sec	The lower the rate of SQL recompilations, the better the database server performance can be.
	Batch requests: This value is the rate of batch requests being handled by the database server.	Reqs/Sec	This metric is a good indicator of how busy an SQL Server is. This metric is generally in step with the server's CPU usage. If the number of batch requests processed is very high (1000 requests/sec or higher), check for CPU and network bottlenecks that can be caused by the high activity rate. The capacity of an MS SQL database server to handle batch requests will depend on the hardware capabilities (CPU, memory, network interface speeds, etc.).
	Work files rate: Indicates the number of workfiles that are created per second.	Workfiles/Sec	Workfiles are used for temp record storage during hash operations. They are created in memory but can overflow or spill to disk (to tempdb, not to separate OS-level files). A value greater than 20 for this measure indicates trash in the tempdb or poorly coded queries.

	Work tables rate: Indicates the number of worktables created per second.	Worktables/Sec	The relational engine may need to build a worktable to perform a logical operation specified in an SQL statement. Worktables are typically generated for certain GROUP BY, ORDER BY, or UNION queries. For example, if an ORDER BY clause references columns not covered by any indexes, the relational engine may need to generate a worktable to sort the result set into the order requested. Worktables are built in **tempdb** and are dropped automatically at the end of the statement.
	Cache requests for work table creation: Indicates the percentage of work tables created where the initial two pages of the work table were not allocated but were immediately available from the work table cache.	Percent	When a query execution plan is cached, the tempdb work tables required by the plan, if any, are often cached. When a work table is cached, the table is truncated (from the previous execution of the code) and up to nine pages remain in the cache for reuse. This improves the performance of the next execution of the query. A value less than 90% for this measure may indicate insufficient memory; this is because, when memory is low, the database server engine drops execution plans and their corresponding work tables from the cache. This may have caused the low cache hit ratio for work tables. On 32-bit systems, a low value for this measure may also hint at the need for upgrading to 64-bit.

	Free space scans: Indicates the percentage of scans that were initiated to search for free space in which to insert a new record fragment.	Percent	This measure represents inserts into a table with no physical ordering of the rows. A table with no ordering, without a clustered index, is known as a heap table. Inserts into heaps will require SQL Server to perform freespace scans to identify pages with free space to insert rows. A heap table also requires an additional, internal column called an uniquifier to be generated for each row inserted. Extra processing is required to define and store a heap table since SQL Server normally uses the clustered index as a storage mechanism for the table data. Freespace scans have an additional I/O expense for inserts and can possibly cause contention on the GAM, SGAM, and PFS pages when there are many connections inserting. It is usually recommended that you physically order the table rows by using a clustered index on the table. A low value is hence desired for this measure. If the value of this measure is very high, you have to quickly investigate the reasons for the same and rapidly figure out what needs to be done to control the free space scans. Towards this end, you first need to identify the I/O-intensive queries executing on the SQL database. For this, you can use the detailed diagnosis of the *Avg physical reads* measure of the **SQL Cached Queries** test. The detailed diagnosis of this test reveals which queries are serviced by direct database accesses (and not by the cache). Since such queries are bound to increase processing overheads, optimizing these queries will not only conserve resources, but will also reduce free space scans. Secondly, use the detailed diagnosis of the **SQL Missing Indexes** test to precisely pinpoint which columns on which tables are not indexed. Unindexed columns can cause queries to take more time and consume more I/O resources. Identifying such columns and indexing them can help reduce I/O and free space scans. Thirdly, you can use the detailed diagnosis of the **SQL Unused Indexes** test to know which indexes are not used effectively, and are hence increasing I/O overheads during query execution.

			By ensuring that such indexes are used, you can reduce I/O and free space scans.
	SQL cancelled rate: Indicates the number of SQL queries that were cancelled per second.	Cancelled/Sec	Users my cancel the currently executing query/batch request at any time. When such cancellations occur, an attention event occurs. An attention is a request by the client to end the currently running request. Typically, an attention event can occur when a cancel request, client-interrupt request, or a broken client connection has occurred. A high value for this measure may hence result in a large number of attention events. If the query or batch that was cancelled was in an explicit user transaction (BEGIN TRAN …. END TRAN), these attention events could result in open transactions and severe blocking problems.

3.3.2 SQL Blocker Processes Test

One common problem encountered with databases is blocking. Suppose that process A is modifying data that process B wants to use. Process B will be blocked until process A has completed what it is doing. This is only one type of blocking situation; others exist and are common. What matters to a database administrator is identifying when blocking is a problem and how to deal with it effectively. When blocking is bad enough, users will notice slowdowns and complain about it. With a large number of users, it is common for tens or hundreds of processes to be blocked when slowdowns are noticed. Killing these processes may or may not solve the problem because 10 processes may be blocked by process B, while process B itself is blocked by process A. Issuing 10 kill statements for the processes blocked by B probably will not help, as new processes will simply become blocked by B. Killing process B may or may not help, because then the next process that was blocked by B, which is given execution time, may get blocked by process A and become the process that is blocking the other 9 remaining processes. When you have lots of blocking that is not resolving in a reasonable amount of time you need to identify the root blocker, or the process at the top of the tree of blocked processes. Imagine again that you have 10 processes blocked by process B, and process B is blocked by process A. If A is not blocked by anything, but is itself responsible for lots of blocking (B and the 10 processes waiting on B), then A would be the root blocker. (Think of it as a traffic jam. Figure 3.19 will help) Killing A (via kill) is likely to unblock B, and once B completes, the 10 processes waiting on B are also likely to complete successfully. The SQL Blocker Processes test monitors the number of root blocker processes in a database.

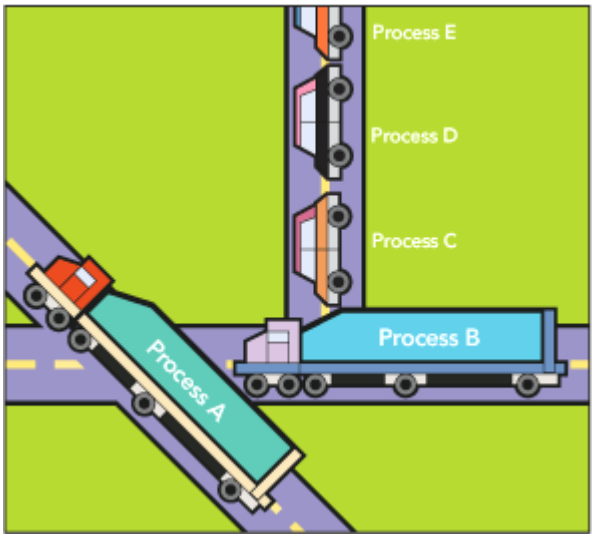


Figure 3.19: The traffic jam analogy representing blocking

Purpose	Monitors the number of root blocker processes in a database
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 11. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 12. **BLOCKED SESSION COUNT** – Specify the minimum number of sessions a process should block for this test to count that process as a root blocker. For instance, if you specify *10* here, it indicates that the *Number of rootblockers* measure of this test will include only those processes that are blocking 10 or more sessions. 13. **MAX BLOCKING TIME SECS** – If a process is blocked for or beyond the duration (in seconds) specified here, then this test will count that process as a process that has been blocked for the maximum time. The details of such processes will then be captured and displayed as part of the detailed diagnosis of the *Max waiting time* measure. For example, if you specify *120* seconds here, then the detailed diagnosis of the *Max waiting time* measure will display the details of all processes that were blocked for 2 minutes and above.

<table>
<tr><td></td><td colspan="3">14. DD FREQUENCY - The DD FREQUENCY refers to the frequency with which detailed diagnosis measures are to be generated. 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. Typically, detailed diagnosis frequencies are set globally, using the DIAGNOSIS CONFIGURATION page that appears when the Configure -> Diagnosis Settings menu sequence is followed. This global setting can be overridden at the test-level using the DD FREQUENCY parameter. To disable the detailed diagnosis capability for a test, you can set this parameter to 0:0.
15. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every MS SQL server monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Number of root blockers:
Indicates the number of root blocker processes.</td><td>Number</td><td>Usually, the number of root blocker processes should be low. If this value increases suddenly, this is a cause for concern. Likewise, if a root-blocker process has been blocking other processes for a long time, it is a reason for further investigation. The detailed diagnosis for this test, if enabled, provides details of the root blocker processes - their SPIDs, programs running these processes, and the queries being issued by these processes. It is usually the case that killing any root-blocker process that has been running for a long while will get the database running well again.</td></tr>
<tr><td>Blocked processes:
Indicates the number of processes that are blocked by the root blockers.</td><td>Number</td><td>Use the detailed diagnosis of this measure to know which processes are blocked.</td></tr>
</table>

	Max waiting time: Indicates the waiting time – i.e., blocked time – of that process(es) that was blocked for the maximum duration.	Secs	If the value of this measure matches or exceeds the **MAX BLOCKING TIME** configuration of this test, it indicates that one/more processes have been blocked for a very long time. You can then use the detailed diagnosis of this measure to identify these blocked processes and figure out who initiated such processes and their resource usage. Processes that are resource hogs can thus be identified.

3.3.3 SQL Transactions Test

This test reports the number of transactions active in an instance of the Database Engine, and the effects of those transactions on resources such as the snapshot isolation level row version store in **tempdb**.

Transactions are logical units of work; a set of operations that must either all succeed or all be erased from a database in order to maintain the logical integrity of the data. All modifications of data in SQL Server databases are made in transactions. When a database is set to allow snapshot isolation level, SQL Server must maintain a record of the modifications made to each row in a database. Each time a row is modified, a copy of the row as it existed before the modification is recorded in a row version store in **tempdb**. Many of the measures of the MsSqlTransTest monitor the size and rate of growth of the following row version stores in **tempdb**:

- The online index build version store is used for online index builds in all databases.
- The common version store is used for all other data modification operations in all databases.

This test again, is specific to MS SQL Server 2005 (or above).

Purpose	Reports the number of transactions active in an instance of the Database Engine, and the effects of those transactions on resources such as the snapshot isolation level row version store in **tempdb**.
Target of the test	An MS SQL server 2005 (or above)
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 5. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**. 6. **SSL** - By default, the **SSL** flag is set to **No**, indicating that the target MS SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled MS SQL server, set the **SSL** flag to **Yes**. 7. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every MS SQL Server 2005 (or above) that is being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Free space in tempdb: Indicates the amount of space that is currently free in tempdb.	KB	There must be enough free space in **tempdb** to hold both the snapshot isolation level version store and all new temporary objects created in this instance of the Database Engine. When the value of this measure decreases, the Database Engine forces the version stores to shrink. During the shrink process, the longest running transactions that have not yet generated row versions are marked as victims. A message 3967 is generated in the error log for each victim transaction. If a transaction is marked as a victim, it can no longer read the row versions in the version store. When it attempts to read row versions, message 3966 is generated and the transaction is rolled back. If the shrinking process succeeds, space becomes available in **tempdb**. Otherwise **tempdb** runs out of space and the following occurs: ➢ Write operations continue to execute but do not generate versions. An information message (3959) appears in the error log, but the transaction that writes data is not affected. ➢ Transactions that attempt to access row versions that were not generated because of a **tempdb** full rollback terminate with an error 3958.
	Version store size: Indicates the amount of space in **tempdb** that is currently being used to store snapshot isolation level row versions.	KB	This information helps determine the amount of space needed in the **tempdb** database for the version store. Monitoring this measure over a period of time provides a useful estimate of additional space needed for **tempdb**.
	Version generation rate: Indicates the rate at which new row versions are added to the snapshot isolation version store in **tempdb**.	KB/Sec	The values of the **Version generation rate** and **Version cleanup rate** measures can be used to predict **tempdb** space requirements.

	Version cleanup rate: Indicates the rate at which row versions are removed from the snapshot isolation version store in **tempdb**.	KB/Sec	Once every minute, a background thread removes row versions that are no longer needed and frees up the version space in **tempdb**. A long-running transaction prevents space in the version store from being released if it meets any of the following conditions: ➢ It uses row versioning-based isolation ➢ It uses triggers, MARS, or online index build operations ➢ It generates row versions
	Version store units: Indicates the number of allocation units currently active in the snapshot isolation version store in **tempdb**.	Number	
	Version store units creation: Indicates the number of new version store units created in the version store since the last measurement period.	Number	
	Version store units deletion: Indicates the number of version store units that were truncated since the last measurement period.	Number	A version store unit is truncated when SQL Server determines that none of the version rows stored in the version store unit are needed to run active transactions.
	Update conflict ratio: Indicates the percentage of transactions using the snapshot isolation level that have encountered update conflicts within the last second.	Percent	An update conflict occurs when a snapshot isolation level transaction attempts to modify a row that last was modified by another transaction that was not committed when the snapshot isolation level transaction started.

	Longest transaction running time: Indicates the length of time since the start of the transaction that has been active longer than any other current transaction.	Secs	Row versions are stored in tempdb for as long as an active transaction needs to access it. If the value of this measure is very high, then it indicates that a transaction has been running for an unreasonable period of time, and is thus preventing the database engine from freeing space from **tempdb**. If the **Free space in tempdb** measure touches alarmingly low levels, then you might have to identify the long running transaction and terminate it. Use **fn_transactions()** to identify the transaction.
	Active transactions: Indicates the number of currently active transactions.	Number	This measure is a good indicator of the current workload on the database server.
	Snapshot transactions: Indicates the number of currently active transactions using the snapshot isolation level.	Number	The value of this measure changes when the first data access occurs, not when the **BEGIN TRANSACTION** statement is issued. Also, note that this measure does not include system transactions.
	Update snapshot transactions: Indicates the number of currently active transactions using the snapshot isolation level that perform update operations.	Number	The sum of **Update snapshot transactions** and **Non snapshot version transactions** represents the total number of transactions that participate in version generation. The difference of **Snapshot transactions** and **Update snapshot transactions** reports the number of read-only snapshot transactions.
	Non snapshot version transactions: Indicates the total number of currently active non-snapshot transactions that generate version records.	Number	

	Temp tables creation rate: Indicates the number of temporary tables/table variables created per second.	Tables/Sec	Temporary tables are created in tempdb. They are backed by physical disk and are even logged into the transaction log. They act like regular tables in that you can query their data via SELECT queries and modify their data via UPDATE, INSERT, and DELETE statements. If created inside a stored procedure they are destroyed upon completion of the stored procedure. Furthermore, the scope of any particular temporary table is the session in which it is created; meaning it is only visible to the current user. You can create indexes and statistics on temporary tables. You can also apply Data Definition Language (DDL) statements against temporary tables to add constraints, defaults, and referential integrity such as primary and foreign keys. You can also add and drop columns from temporary tables. The syntax for creating table variables is quite similar to creating either regular or temporary tables. The only differences involve a naming convention unique to variables in general, and the need to declare the table variable as you would any other local variable in Transact SQL. Unlike temporary or regular table objects, table variables have certain clear limitations. • Table variables can not have Non-Clustered Indexes • You can not create constraints in table variables

			• You can not create default values on table variable columns • Statistics can not be created against table variables Temporary tables are usually preferred over table variables for a few important reasons: they behave more like physical tables in respect to indexing and statistics creation and lifespan.

3.3.4 SQL User Processes Test

This test reports the number and state of sessions of each user who is currently connected to the MS SQL server. Using the metrics reported by this test, administrators can promptly isolate idle sessions and suspended sessions, which are a drain on a server's resources.

Purpose	Reports the number and state of sessions of each user who is currently connected to the MS SQL server
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **EXCLUDEUSER** - In the **EXCLUDEUSER** text box, specify a comma-separated list of user names that need to be excluded from monitoring. By default, *none* is displayed here indicating that this test monitors connections initiated by all current users to the MS SQL server, by default. 11. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 12. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

<table>
<tr><td></td><td colspan="3">13. 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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each user currently connected to the MS SQL server monitored</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total processes:
Indicates the total number of sessions currently open on the server for this user.</td><td>Number</td><td></td></tr>
<tr><td>Running processes:
Indicates the number of sessions of this user that are currently active.</td><td>Number</td><td>The detailed diagnosis of this measure, if enabled, will provide the complete details of the active sessions of a particular user. Using this information, you can understand how each of the connections were made - i.e., using which program - and from where - i.e., from which host.</td></tr>
<tr><td>Sleeping processes:
Indicates the number of sessions initiated by this user that are currently idle.</td><td>Number</td><td>Ideally, the value of this measure should be low. A high value is indicative of a large number of idle sessions, which in turn causes the unnecessary consumption of critical server resources. Idle sessions also unnecessarily lock connections from the connection pool, thereby denying other users access to the server for performing important tasks.

The detailed diagnosis of this measure, if enabled, will provide the complete details of the idle sessions of a particular user. Using this information, you can understand how each of the idle connections were made - i.e., using which program - and from where - i.e., from which host.</td></tr>
</table>

	Suspended processes: Indicates the number of sessions initiated by this user that are currently suspended.	Number	A session can switch to a suspended state if one/more processes triggered in that session could not continue executing; a possible reason for this could be that the processes are waiting for blocked rows or a blocked table to be released. The detailed diagnosis of this measure, if enabled, will provide the complete details of the suspended sessions of a particular user.
	Background processes: Indicates the number of background processes currently running for this user.	Number	The detailed diagnosis of this measure, if enabled, provides the details pertaining to the background processes currently executing.

The detailed diagnosis of the *Sleeping processes* measure, if enabled, will provide the complete details of the idle sessions of a particular user. Using this information, you can understand how each of the idle connections were made - i.e., using which program - and from where - i.e., from which host.

Sleeping processes Status Details

Time	LoginName	No of Connections	Connections Status	ProgramName	HostName
Nov 04, 2009 18:53:04					
	chitra_68_oct05	4	sleeping	jTDS	EG100
	chitra_68_oct05	1	sleeping	SQL Query Analyzer	EG103
	chitra_68_oct05	1	sleeping	SQL Query Analyzer - Object Browser	EG103

Figure 3.20: The detailed diagnosis of the Sleeping processes measure

3.4 The MS SQL Databases Layer

The test associated with this layer (see Figure 3.21) monitors space usage on an MS SQL server database.

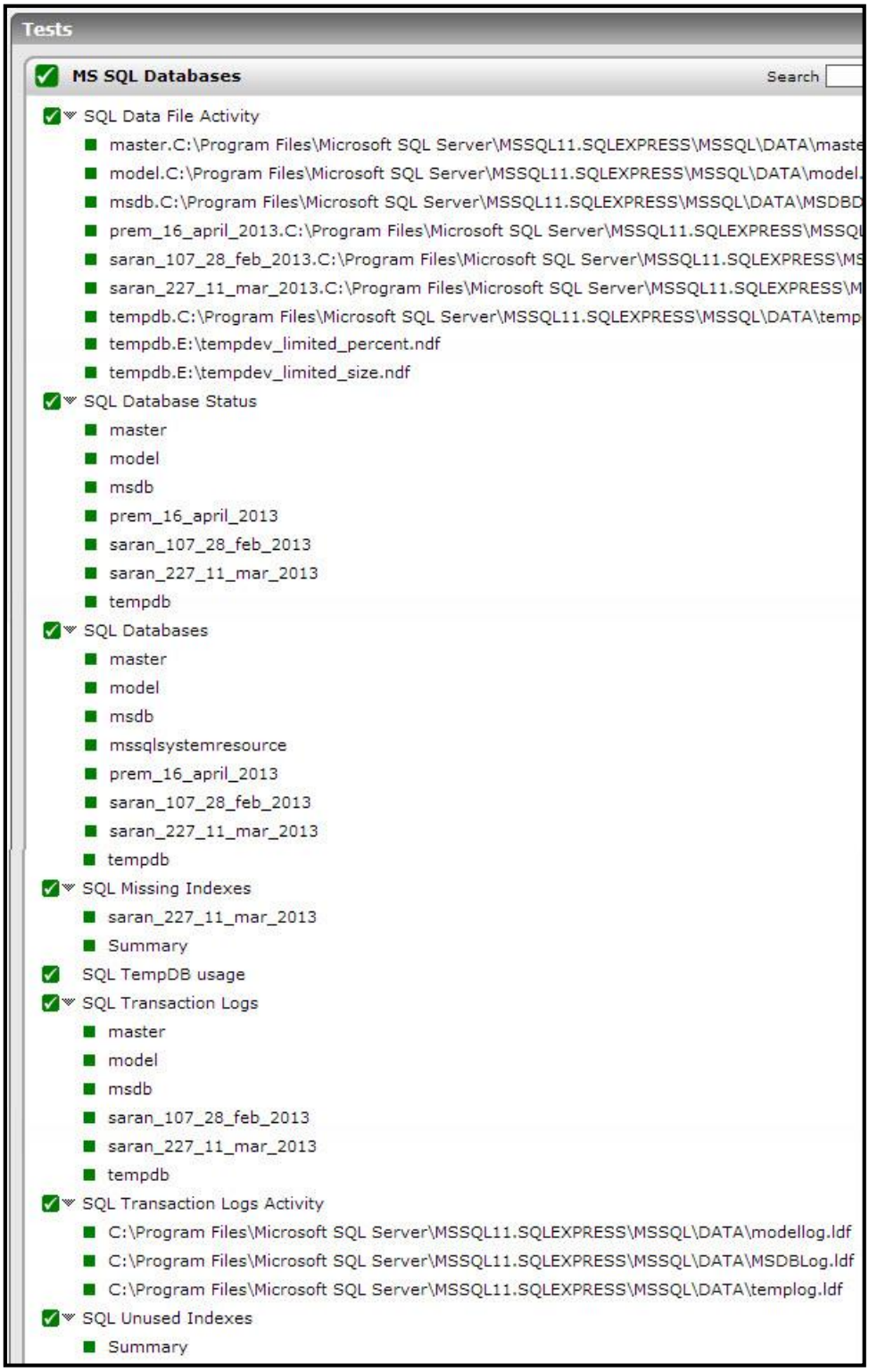


Figure 3.21: The tests associated with the MS SQL Databases Layer

3.4.1 SQL TempDB Usage Test

The **tempdb** system database is a global resource that is available to all users connected to the instance of SQL Server and is used to hold the following:

- Temporary user objects that are explicitly created, such as: global or local temporary tables, temporary stored procedures, table variables, or cursors.
- Internal objects that are created by the SQL Server Database Engine, for example, work tables to store intermediate results for spools or sorting.
- Row versions that are generated by data modification transactions in a database that uses read-committed using row versioning isolation or snapshot isolation transactions.
- Row versions that are generated by data modification transactions for features, such as: online index operations, Multiple Active Result Sets (MARS), and AFTER triggers.

Since **tempdb** is shared by multiple users/applications, if **tempdb** runs out of disk space, it can cause significant disruptions in the SQL Server production environment and can suspend operations of all applications that are using it. To prevent such an eventuality, you can use the **SQL TempDB Usage** test to continuously track **tempDB** usage, capture potential space contentions, and initiate measures to avert the contention.

Purpose	Continuously tracks **tempDB** usage, captures potential space contentions, and enables administrators to quickly initiate measures to avert the contention
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. SSL – If the MS SQL server being monitored is an SSL-enabled server, then set the SSL flag to Yes. If not, then set the SSL flag to No.
5. INSTANCE - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the INSTANCE parameter.
6. USER – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the Sysadmin role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE, and PUBLIC roles in this text box.
7. CONFIRM PASSWORD - Confirm the password by retyping it.
8. DOMAIN - By default, none is displayed in the DOMAIN text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the DOMAIN can continue to be none. On the other hand, if 'Windows only' authentication has been enabled, then, in the DOMAIN text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the USER name and PASSWORD that you provide should be that of a user authorized to access the monitored SQL server.
9. ISNTLMV2 - In some Windows networks, NTLM (NT LAN Manager) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the ISNTLMV2 flag is set to No, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to Yes if NTLMv2 is enabled on the target host.
10. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every MS SQL server database monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total size:
Indicates the total size of tempdb.</td><td>MB</td><td></td></tr>
</table>

	User objects: Indicates the amount of space allocated from uniform extents to user objects in the **tempdb** database.	MB	User objects include both user-defined tables and indexes, and system catalog tables and indexes. User-defined tables include the global temporary tables such as ##t, and local temporary tables such as #t. Both of these objects are session scoped but a global temporary table lives until all sessions that are using it expire or terminate. Local temporary tables, on the other hand, are destroyed when the scope (for example, stored procedure or session) they were created in expires or terminates. Local temporary tables also include table variables such as @t, the return value in table valued functions, and the mapping index for online clustered index build with the SORT_IN_TEMPDB option. A high value of this measure indicates that a lot of space has been used by user objects. Compare the value of this measure with that of the *Internal objects, Version store*, and *Mixed extent* measures to know where the maximum **tempdb** space has been spent.

	Internal objects: Indicates the amount of space allocated from uniform extents to internal objects in the **tempdb** database.	MB	Internal objects are created internally by SQL Server. These objects are used: a. To store intermediate runs for sort. b. To store intermediate results for hash joins and hash aggregates. c. To store XML variables or other large object (LOB) data type variables. The LOB data type includes all of the large object types: **text**, **image**, **ntext**,**var char(max)**, **varbinary(max)** , and all others. d. By queries that need a spool to store intermediate results. e. By keyset cursors to store the keys. f. By static cursors to store a query result. g. By Service Broker to store messages in transit. h. By INSTEAD OF triggers to store data for internal processing. Internal objects are also used by any feature that uses these operations. For example, DBCC CHECK internally uses a query that may need to spool intermediate results. Query notification and event notification use Service Broker, so they need space in **tempdb** as well. Compare the value of this measure with that of the *User objects, Version store*, and *Mixed extent* measures to know where the maximum **tempdb** space has been spent. If this measure reports the highest value it implies that query plans are making heavy use of the **tempdb**.
			This is not necessarily a problem, but you may want to look at the query plans to see if alternate query plans can be generated by creating indexes or by re-formulating the queries so as to minimize **tempdb** space usage.

	Version store: Indicates the amount of space allocated from uniform extents for the version store.	MB	Version stores are used to store row versions generated by transactions for features such as snapshot isolation, triggers, MARS (multiple active result sets), and online index build. There are two version stores in **tempdb** for the whole instance of SQL Server. The online index build version store is for row versions from tables that have online index build operations on them. The common version store is for row versions from all other tables in all databases. Compare the value of this measure with that of the *User objects, Internal objects,* and *Mixed extent* measures to know where the maximum **tempdb** space has been spent. If this comparative analysis reveals that this measure reports the highest value, it implies that version store cleanup cannot keep pace with version generation. See if a long-running transaction is preventing version store cleanup. Or, a high transaction throughput might be generating a large number of versions per minute. The background task cleans up versions every minute.
	Mixed extent: Indicates the amount of space that is used by objects of multiple types (user objects, internal objects, version store, Index Allocation Map (IAM) pages, etc.)	MB	Extents are the basic unit in which space is allocated to tables and indexes. To make its space allocation efficient, SQL Server does not allocate entire extents to tables with small amounts of data. SQL Server has two types of extents - Uniform extents are owned by a single object; all eight pages in the extent can only be used by the owner object. A mixed extent is a single extent that contains multiple tables; it can be shared by up to eight objects. A high value indicates that a large amount of **tempdb** space is occupied by mixed extents. Compare the value of this measure with that of the *User objects, Internal objects,* and *Version store* measures to know where the maximum **tempdb** space has been spent.
	Free space: Indicates the amount of unused space in the **tempdb** database.	MB	A high value is desired for this measure.

	Usage of allocated space: Indicates the percentage of allocated **tempdb** space that is currently in use.	Percent	A consistent rise in the value of this measure could indicate a gradual, but steady erosion of space in the **tempdb** database. A value close to 100% signals a potential space crunch in the **tempdb**, which can affect the performance of all the applications using the **tempdb**. Under such circumstances, it would be good practice to compare the values of the *User objects, Internal objects, Version store, and Mixed extents* measures to know which object type is consuming the most space in the **tempdb**. You can then use the DMVs in the SQL server to analyze which Transact-SQL statements are the top consumers of **tempdb** space. You can kill such tasks, where appropriate, to free space.
	Is auto-growth enabled? Indicates whether/not auto-growth is enabled for th **tempdb** database.		Auto-growth is the process by which the SQL Server engine expands the size of a database file when it runs out of space. The amount by which a database file grows is based on the settings that you have for the file growth options for your database. If this setting is enabled for one/more database files in the **tempdb** database, this measure will report the value **Yes**. If this setting is not enabled for any of the database files in the **tempdb** database, then this measure will report the value **No**. The numeric values that correspond to the above-mentioned measure values are listed in the table below: **Measure Value** / **Numeric Value** **Yes** / 100 **No** / 0 **Note:** Typically, this measure reports the **Measure Value**s listed in the table above as the status of the **Autogrowth** setting. In the graph of this measure however, the status will be represented using the numeric values – i.e., *100* and *0*.

	Max file size: Indicates the maximum size upto which the database files (for which the **Auto-growth setting** has been enabled) in the **tempdb** database can grow. **This measure will be reported only if the 'Is auto-growth enabled?' measure returns the value 'Yes'.**	MB	There are three different settings you can use to identify how your database files will grow. They can grow by a specific size, a percentage of the current size, or not grow at all. Additionally you can set your files to unrestricted growth, which means they will keep growing as they need more space or you run out of disk space. Or you can restrict the growth of a database file to grow no larger than a specified size. Each one of these different auto-grow settings have defaults, or you can set them for each database file. The default auto-growth settings for a database are rarely the ideal settings for how your database should grow. If you have an idea of the growth profile of your database when you first build it then you should set your auto-growth properties based on those growth projections. If you don't have any idea of how fast your database will grow then you should be monitoring for auto-growth events. Knowing how often your database grows will give you some ideas of the growth rate of your database.
	Free disk space: Indicates the amount of disk space that is currently available for use for **tempdb** database.	MB	A high value implies that the **tempdb** has adequate disk space for growth. If the value of this measure is low, then you may have to fine-tune your auto-growth settings accordingly.
	Usage as % of max size: Indicates the percentage of *Max file size* that is currently available for use for **tempdb** database. **This measure will be reported only if the 'Is auto-growth enabled?' measure returns the value 'Yes'.**	Percent	This measure reports the *Free disk space* value as a percentage of *Max file size*. In other words: *Free disk space/Max file size * 100* If your database is set to auto-grow till disk capacity is reached, then, a high value of this measure indicates that there is enough space for the **tempdb** database to grow. A low value indicates that there is very little disk space for the use of **tempdb**. You may then have to free up some disk space or fine-tune your auto-growth settings.

3.4.2 SQL Databases Test

This test monitors the transactions that occur on every database of an MS SQL server.

Purpose	This test monitors the transactions that occur on every database an MS SQL server.

<table>
<tr><td>Target of the test</td><td colspan="3">An MS SQL server</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An Internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. INSTANCE – The name of a specific MS SQL instance to be monitored. The default value of this parameter is “default”. To monitor an MS SQL instance named “CFS”, enter this as the value of the “instance” parameter.
5. SSL - By default, the SSL flag is set to No, indicating that the target MS SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled MS SQL server, set the SSL flag to Yes.
6. USEPERFMON – By default, this flag is set to Yes, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to No. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to No.
7. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as “Not applicable’ by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every database on the Ms Sql server monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Active transactions:
This value indicates the current number of active transactions yet to be committed to the database.</td><td>Number</td><td>A large number for this value may indicate a large number of active transactions. Alternately this may also indicate that due to some reasons the users are not able to complete the transactions.</td></tr>
<tr><td>Transaction rate:
This measure indicates the number of transactions that are started for the database per second.</td><td>Trans/Sec</td><td>A high value of this metric indicates a lot of transactional activity happening to the database</td></tr>
</table>

	Replication transaction rate: This value indicates the number of transactions per second read out of the transaction log of the publication database and delivered to the distribution database.	Trans/Sec	A high value indicates that there is more replicated transactions happening from the publication database and is being sent to the distribution database.
	Pending replication transactions: This value indicates the number of pending replication transactions in the database.	Number	This is the number of transactions in the transaction log of the publication database marked for replication, but not yet delivered to the distribution database.
	Data file size: This metric is the cumulative size of all the data files in the database server.	MB	The value of this metric provides an idea of the growth of the databases hosted by the database server.
	Log flush waits: This value indicates the number of transaction commits that are waiting in the log flush ready to be flushed.	Waits/Sec	A high value here may indicate non-optimal allocation of the log buffer related parameters.
	Write transaction rate: Indicates the number of transactions that wrote to the database and committed, in the last second.	Trans/Sec	

3.4.3 SQL Database Space Test

Typically, SQL Server databases have three types of files, as shown in the following table.

File	Description
Primary	The primary data file contains the startup information for the database and points to the other files in the database. User data and objects can be stored in this file or in secondary data files. Every database has one primary data file. The recommended file name extension for primary data files is .mdf.
Secondary	Secondary data files are optional, are user-defined, and store user data. Secondary files can be used to spread data across multiple disks by putting each file on a different disk drive. Additionally, if a database exceeds the maximum size for a single Windows file, you can use secondary data files so the database can continue to grow. The recommended file name extension for secondary data files is .ndf.
Transaction Log	The transaction log files hold the log information that is used to recover the database. There must be at least one log file for each database. The recommended file name extension for transaction logs is .ldf.

All data files are stored in the filegroups listed in the following table.

Filegroup	Description
Primary	The filegroup that contains the primary file. All system tables are allocated to the primary filegroup.
User-defined	Any filegroup that is specifically created by the user when the user first creates or later modifies the database.

If even a single file group in a database runs out of free space, serious performance degradations will be noticed in applications that depend on that file group for their data needs. To avoid this, administrators must track space usage both at the database-level and at the individual file group-level and proactively identify those file groups that are over-utilized. The **SQL Database Space** test provides administrators with both these usage insights. By monitoring the space usage in each SQL database, the test points administrators to those databases that are consuming too much space and reveals the type of data that is hogging space – data in tables? Or indexes? Alongside, the test also reports usage metrics for each file group in every SQL database, and accurately pinpoints those file groups that may soon fill up the disk! This way, the test turns the spotlight on databases and file groups that may have to be resized to ensure peak application performance.

Note that this test will report metrics for file groups only if Microsoft SQL Server 2008 R2 (and above) is monitored.

Purpose	By monitoring the space usage in each SQL database, the test points administrators to those databases that are consuming too much space and reveals the type of data that is hogging space – data in tables? Or indexes? Alongside, the test also reports usage metrics for each file group in every SQL database, and accurately pinpoints those file groups that may soon fill up the disk! This way, the test turns the spotlight on databases and file groups that may have to be resized to ensure peak application performance.
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – Provide the name of a SQL user with the **Sysadmin** role. However, if you do not want to expose the credentials of a **Sysadmin**, then create a special user for this purpose on each of the databases to be monitored, and make sure that you assign any of the following privileges to that user: • Assign the **db_datareader** privilege to that user in each of the databases to be monitored; (OR) • Assign the **PUBLIC** role to that user, and grant **execute** permission to that user for the **sp_spaceused** procedure in every database to be monitored **Note that the name of the special user should be the same in all the databases**. 7. **PASSWORD** – Provide the password of the specified **USER**. 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **EXCLUDEDB** - Specify a comma-separated . list of databases for which the space computation need not be done (e.g., *temp*). The default value is 'none'. 11. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 12. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**. 13. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every MS SQL server database and every file group in each database		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Total size: The space allocated to a database.	MB	The **Total size** does not include the size of the database log files. **This measure is reported only for a database.**
	Reserved space percent: The percentage of allocated space reserved for tables and indexes of a database	Percent	If the value of this measure reaches 100%, it indicates that the total space in the database has been completely allocated. New tables/indexes can be added to the database, only if its total size is increased. **This measure is reported only for a database.**
	Reserved space: The amount of allocated space reserved for the tables and indexes created on a database	MB	If the value of this measure becomes equal to that of the **Total size** measure, new tables/indexes can no longer be created on the database. To create new tables, you must increase the database size. **This measure is reported only for a database.**
	Data space: The amount of allocated space used by data in tables.	MB	The total allocated space that is in use in a database is the sum of the value of the *Data space* and *Index space* measures. **These measures are reported only for a database.**
	Index space: The amount of allocated space used by indexes.	MB	
	Unused space: The amount of allocated space that is available for use in the database	MB	This is the difference between the *Total size* and *Reserved space*. **These measures are reported only for a database.**

	Is auto-growth enabled? Indicates whether/not auto-growth is enabled for this file group.		**This measure is reported only for file groups.** Auto-growth is the process by which the SQL Server engine expands the size of a database file when it runs out of space. The amount by which a database file grows is based on the settings that you have for the file growth options for the data file. If this setting is enabled for even one file in a file group, this measure will report the value **Yes**. If this setting is not enabled for any of the database files in a file group, then this measure will report the value **No**. The numeric values that correspond to the above-mentioned measure values are listed in the table below: **Measure Value** / **Numeric Value** **Yes** / 100 **No** / 0 **Note:** Typically, this measure reports the **Measure Value**s listed in the table above as the status of the **Autogrowth** setting. In the graph of this measure however, the status will be represented using the numeric values – i.e., *100* and *0*.

	Max file size: Indicates the maximum size upto which the database files in this file group can grow.	MB	**This measure is reported only for file groups.** Each database file that is associated with your database has an auto-growth setting. There are three different settings you can use to identify how your database files will grow. They can grow by a specific size, a percentage of the current size, or not grow at all. Additionally you can set your files to unrestricted growth, which means they will keep growing as they need more space or you run out of disk space. Or you can restrict the growth of a database file to grow no larger than a specified size. Each one of these different auto-grow settings have defaults, or you can set them for each database file. If the auto-growth setting is not enabled at all for a file in a file group, then the amount of space that was originally allocated to that file will be regarded as the **Max file size** of that file. On the other hand, if the **Auto-growth** setting is enabled for a file in the file group, then the **Max file size** of that file will be one of the following: • If a specific size limit is explicitly set for the file, then this will be considered as the **Max file size** of that file. • If no size limit is set for the file, then the total capacity of the disk drive in which that file resides will be considered as the **Max file size** of that file. So, if a file group consists of a few data files for which auto-growth is enabled and a few others for which it is disabled, then the **Max file size** of that file group will be a sum total of the following: • The sum of the space allocated to each of the files for which auto-growth is not enabled; • The sum of the maximum size limits, if defined, for each file for which auto-growth is

			disabled; • The sum of the total capacity of the disks containing the auto-growth-enabled files for which no size limit is defined.
	Free disk space: Indicates the amount of disk space that is currently available for use for this file group.	MB	**This measure is reported only for file groups.** A high value implies that the database files in the file group have adequate disk space for growth. If the value of this measure is low, then you may have to fine-tune your auto-growth settings accordingly.
	Free disk space percent: Indicates the percentage of *Max file size* that is currently available for use for this file group.	Percent	**This measure is reported only for file groups.** This measure reports the *Free disk space* value as a percentage of *Max file size*. In other words: *Free disk space/Max file size * 100* If many files in a file group are set to auto-grow till disk capacity is reached, then, a high value of this measure indicates that there is enough space for the files to grow. A low value indicates that there is very little room for file growth.

3.4.4 SQL Data File Activity Test

By periodically monitoring the I/O activity on each datafile on the MS SQL server and observing the growth in size of the datafile, this test sheds light on the following:

- Datafiles that are experiencing I/O bottlenecks;
- Datafiles that are consuming too much disk space

Purpose	Periodically monitors the I/O activity on and the growth in the size of the log file associated with each database
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **EXCLUDE INFO** - By default, this is set to none, indicating that the test will monitor all the databases on the MS SQL server by default. To exclude specific databases from the monitoring scope of this test, provide a comma-separated list of databases in the **EXCLUDE INFO** text box. 11. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 12. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

	13. **SHOW DATAFILE PATH** - This test reports a set of results for each datafile on the target Oracle database server. This means that every datafile is a descriptor of this test. By default, while displaying the descriptors of this test, the eG monitoring console does not prefix the datafile names with the full path to the datafiles. This is why, the **SHOW DATAFILE PATH** flag is set to **No** by default. If you want the data file names to be prefixed by the full path to the data files, then, set the **SHOW DATAFILE PATH** flag to **Yes**. 14. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. For instance, if you set to 1:1, it means that detailed measures will be generated every time this test runs, and also every time the test detects a problem. 15. **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 enabled/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.		
Outputs of the test	One set of results for each database on the MS SQL server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Write rate: Indicates the rate at which writes occurred on this datafile.	Writes/Sec	
	Data write rate: Indicates the rate at which data was written to this datafile.	KB/Sec	
	I/O stall writes: Indicates the total time taken to write to this datafile.	Millisecs	A high value for this measure could indicate a bottleneck while writing to the datafile. By comparing the value of this measure across datafiles, you can identify the data file to which write operations are taking too long to complete.
	Read rate: Indicates the rate of reads from this datafile.	Reads/Sec	
	Data read rate: Indicates the rate at which data was read from this datafile.	KB/Sec	

	I/O stall reads: Indicates the total time taken to read from this datafile.	Millisecs	A high value for this measure could indicate a bottleneck while reading from the datafile. By comparing the value of this measure across datafiles, you can identify the datafile to which read operations are taking too long to complete.
	I/O stall: Indicates the total time taken for I/O to complete on this datafile.	Millisecs	A high value for this measure could indicate an I/O bottleneck on this datafile.
	Size on disk: Indicates the total size on disk of each datafile.	Bytes	This measure is used to determine the growth of the datafile. A low value is desired for this measure. A very high value, or a consistent increase in this value may adversely impact I/O operations. You may want to consider maintaining multiple datafiles of smaller sizes to improve I/O efficiency, and to speed up backup/restore operations.

3.4.5 SQL Database Status Test

If a user complaints of problems while accessing a database, the knowledge of the current state of that database will enable administrators to promptly diagnose the reason for such an occurrence. This test auto-discovers all the databases on an MS SQL server and reports the current state of each database, thereby enabling administrators to easily troubleshoot issues related to database access.

Purpose	Auto-discovers all the databases on an MS SQL server and reports the current state of each database
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 12. **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 enabled/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.
Outputs of the test	One set of results for every MS SQL server database monitored

<table>
<tr><th>Measurements made by the test</th><th>Measurement</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td>Database status:
Indicates the current state of this database.</td><td></td><td>The states that can be reported by this measure, the numeric value that corresponds to each state, and a brief description of the state are provided below:
<table>
<tr><th>Value</th><th>State</th><th>Description</th></tr>
<tr><td>0</td><td>ONLINE</td><td>Database is available for access. The primary filegroup is online, although the undo phase of recovery may not have been completed.</td></tr>
<tr><td>1</td><td>RESTORING</td><td>One or more files of the primary filegroup are being restored, or one or more secondary files are being restored offline. The database is unavailable in this case.</td></tr>
<tr><td>2</td><td>RECOVERING</td><td>Database is being recovered. The recovering process is a transient state; the database will automatically become online if the recovery succeeds. If the recovery fails, the database will become suspect. The database is unavailable in this case.</td></tr>
</table>
</td></tr>
</table>

			Value	**State**	**Description**
			3	RECOVERY_PENDING	SQL Server has encountered a resource-related error during recovery. The database is not damaged, but files may be missing or system resource limitations may be preventing it from starting. The database is unavailable. Additional action by the user is required to resolve the error and let the recovery process be completed.
			4	SUSPECT	At least the primary filegroup is suspect and may be damaged. The database cannot be recovered during startup of SQL Server. The database is unavailable. Additional action by the user is required to resolve the problem.

			Value	State	Description
			5	EMERGENCY	User has changed the database and set the status to EMERGENCY. The database is in single-user mode and may be repaired or restored. The database is marked READ_ONLY, logging is disabled, and access is limited to members of the sysadmin fixed server role. EMERGENCY is primarily used for troubleshooting purposes. For example, a database marked as suspect can be set to the EMERGENCY state. This could permit the system administrator read-only access to the database. Only members of the sysadmin fixed server role can set a database to the EMERGENCY state.
			6	OFFLINE	Database is unavailable. A database becomes offline by explicit user action and remains offline until additional user action is taken. For example, the database may be taken offline in order to move a file to a new disk. The database is then brought back online after the move has been completed.

			The detailed diagnosis of this measure reports the user access mode of the database, the database recovery model, and the log re-use wait state of the database.

3.4.6 SQL Transaction Logs Test

Every SQL Server database has at least two files associated with it: one data file that houses the actual data and one transaction log file. The transaction log is a fundamental component of a database management system. All changes to application data in the database are recorded serially in the transaction log. The information recorded includes the following:

- the beginning time of each transaction
- the actual changes made to the data and enough information to undo the modifications made during each transaction (accomplished using before and after images of the data)
- the allocation and reallocation of database pages
- the actual commit or rollback of each transaction

Using this information, the DBMS can track which transaction made which changes to SQL Server data. However, it is only during transaction rollbacks, commits, and database recovery operations that the transaction log serves its true purpose. When a transaction is rolled back, the SQL Server copies before images to the database for every modification made since the BEGIN TRANSACTION. During a recovery scenario you can use the transaction log to restore a database. This causes a roll forward of the transaction log. During a roll forward SQL Server will copy after images of each modification to the database. Using the logged data SQL Server ensures that each modification is applied in the same order that it originally occurred.

Lack of adequate space in the transaction log would hence have serious repercussions on the way the SQL server carries out database updations; sometimes, critical changes to the database could get lost due to a space crunch. The **SQL Transaction Logs** test enables administrators to constantly track the space consumption by the transaction log, so that administrators are instantly notified of inadequacies, and are prompted to act fast.

Purpose	Constantly tracks the space consumption by the transaction log
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. SSL – If the MS SQL server being monitored is an SSL-enabled server, then set the SSL flag to Yes. If not, then set the SSL flag to No.
5. INSTANCE - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the INSTANCE parameter.
6. USER – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the Sysadmin role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE, and PUBLIC roles in this text box.
7. PASSWORD - The password of the specified USER
8. CONFIRM PASSWORD - Confirm the password by retyping it.
9. DOMAIN - By default, none is displayed in the DOMAIN text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the DOMAIN can continue to be none. On the other hand, if 'Windows only' authentication has been enabled, then, in the DOMAIN text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the USER name and PASSWORD that you provide should be that of a user authorized to access the monitored SQL server.
10. ISNTLMV2 - In some Windows networks, NTLM (NT LAN Manager) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the ISNTLMV2 flag is set to No, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to Yes if NTLMv2 is enabled on the target host.
11. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every MS SQL server database monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Log size:
Indicates the current size of the transaction log attached to this database.</td><td>MB</td><td></td></tr>
</table>

<table>
<tr><td></td><td>Usage of allocated space:
Indicates the percentage of transaction log space that is currently in use.</td><td>Percent</td><td>Ideally, this value should be low. A high value or a value close to 100% requires immediate attention, as it indicates that the transaction log is suffering from severe space constraints. There is hence the danger of subsequent database modifications going unrecorded. Typically, excessive space usage can be attributed to too many changes been written to the log, but very little/none to the database. Further diagnosis can alone reveal the root-cause of this deviant behavior.</td></tr>
<tr><td></td><td>Is auto-growth enabled?
Indicates whether/not auto-growth is enabled for the transaction logs.</td><td></td><td>Turning on the autogrowth setting of the transaction logs enables the transaction log files to automatically grow by a configured percentage of the current log file size when more data space is required.
If this setting is enabled for the transaction logs, this measure will report the value Yes. If this setting is not enabled, then this measure will report the value No.
The numeric values that correspond to the above-mentioned measure values are listed in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Yes</td><td>100</td></tr><tr><td>No</td><td>0</td></tr></table>
Note:
Typically, this measure reports the Measure Values listed in the table above as the status of the Autogrowth setting. In the graph of this measure however, the status will be represented using the numeric values – i.e., 100 and 0.
This measure is available for Microsoft SQL Server 2008 R2 and above only.</td></tr>
<tr><td></td><td>Auto-growth percent:
Indicates the percentage by which the transaction log files have been configured to automatically grow when more data space is required.</td><td>Percent</td><td>The growth increment of your transaction log must be large enough to stay ahead of the needs of your transaction units. Also, the growth increment must be large enough to avoid the following performance</td></tr>
</table>

	Auto growth size: Indicates the size (in MB) by which the transaction log files have been configured to automatically grow when more data space is required.	MB	penalities: i. If you run a transaction that requires more log space than is available, and you have turned on the **autogrow** option for the transaction log of that database, then the time it takes the transaction to complete will include the time it takes the transaction log to grow by the configured amount. If the growth increment is large or there is some other factor that causes it to take a long time, the query in which you open the transaction might fail because of a timeout error. j. If you run a large transaction that requires the log to grow, other transactions that require a write to the transaction log will also have to wait until the grow operation completes. k. If you have many file growths in your log files, you may have an excessively large number of virtual log files (VLF). This can lead to performance problems with database startup/online operations, replication, mirroring, and change data capture (CDC). Additionally, this can sometimes cause performance problems with data modifications.

			Besides considering the above factors when deciding on an autogrowth value to set, you also need to decide whether to set this value as a percentage or in MB. Typically, when you use percentage as an auto growth factor and the transaction log is smaller in size, you will probably encounter many repeated growth instances. At large file size or percentages you may encounter a timeout or long period of blocking while the file is grown. Typically, it is recommended to set the **AUTOGROW** value to an optimum value in MB instead of percentages. A general rule of thumb to you can use for testing is to set your **AUTOGROW** setting to about one-eight the size of the file. **These measures are available for Microsoft SQL Server 2008 R2 and above only.**
	Max size of log: Indicates the maximum size to which the transaction log can grow.	MB	You need to turn on the **MAXSIZE** setting for each file to prevent any one file from growing to a point where it uses up all available disk space. **This measure is available for Microsoft SQL Server 2008 R2 and above only.**

	Usage as % of max size: Indicates the percentage of maximum log file size that is currently being used.	Percent	If the value of this measure keeps growing close to 100%, it indicates that the log is growing rapidly and may soon run out of space! In this case, do the following to make sure that the transaction log file does not become full: • In case of a database that has been configured with the *Simple Recovery* model, you need to ensure that the transaction log is automatically cleared on all checkpoints to ensure that there is adequate space to log subsequent transactions. You should also check for long-running transactions and search for possible flaws in the applications using the database. • In case of a database that has been configured with the *Full Recovery* model, check whether the transaction log backups have been defined and have succeeded. Also, verify whether the log backups occur often enough. Here again, keep your eyes open for long-running transactions and the reasons for their longevity. On the other hand, if the value of this measure touches 100%, it implies that the log file cannot grow more. An *error 9002* is then raised. This means, that operations that change the state of the database cannot be successfully executed until sufficient free space is available in the transaction log. But, this also means that there's no guarantee that all the necessary information was recorded into the transaction log. The previous check list applies also this time but since the log is really full, you should also check the auto growth setting of the log file(s). Is it properly defined or should the log be let to grow more?
			Regardless of the decision to let the log grow more or not, the log can be truncated in a situation like this. **This measure is available for Microsoft SQL Server 2008 R2 and above only.**

3.4.7 SQL Missing Indexes Test

SQL Server allows you to put **indexes** on table columns, to speed up **`WHERE`** and **`JOIN`** statements on those columns. If a SQL query takes longer (much longer) to complete, it could be because one/more of these indexes are 'missing'. When the query optimizer optimizes a query, it identifies those **indexes** it would have liked to have used but were not available - these are called 'missing indexes'.

With the help of the **SQL Missing Indexes** test, you can be promptly alerted when the query optimizer finds one/more 'missing indexes'. Besides reporting the count of the missing indexes, the test also reveals which queries require these indexes, thus enabling you to quickly initiate index creation and query optimization.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the count of the missing indexes and also reveals which queries require these indexes
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 6. **PASSWORD** - The password of the specified **USER** 7. **CONFIRM PASSWORD** - Confirm the password by retyping it. 8. **INSTANCE** – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **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 enabled/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. 12. **ISPASSIVE –** If the value chosen is **Yes**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every database on the monitored MS SQL server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Number of Missing Indexes: Indicates the total number of missing indexes found in the queries that are currently executing on this database.	Number	Use the detailed diagnosis of this measure to know which queries are missing one/more indexes. Knowledge of these queries can greatly aid database administrators in optimizing them, which in turn will increase the speed of database accesses and consequently, improve overall database performance.

3.4.8 SQL Unused Indexes Test

One of the balancing acts of SQL Server is the use of indexes. Too few indexes can cause scans to occur which hurts performance and too many indexes causes overhead for index maintenance during data updates and also a bloated database. Administrators hence need to continuously track which indexes are used and how they are used. **Unused indexes** in particular can have a negative impact on performance. This is because when the underlying table data is modified, the index may need to be updated also. This takes additional time and can even increase the probability of blocking. To avoid this, administrators need to rapidly isolate the unused indexes and drop them. The **Unused Indexes** test facilitates this. Using this test, administrators can quickly determine the number of unused indexes in each database, and also understand how badly that index impacts database performance.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Helps quickly determine the number of unused indexes in each database, and also understand how badly that index impacts database performance
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 6. **PASSWORD** - The password of the specified **USER** 7. **CONFIRM PASSWORD** - Confirm the password by retyping it. 8. **INSTANCE** – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **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. 12. The option to selectively enabled/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. 13. **ISPASSIVE –** If the value chosen is **Yes**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
Outputs of the test	One set of results for every database on the monitored MS SQL server

Measurements made by the test	Measurement	Measurement Unit	Interpretation

	Number of Unused Indexes: Indicates the total number of unused indexes currently found in this database .	Number	Use the detailed diagnosis of this measure to know which indexes in a database are unused. The detailed diagnosis also reveals how often each such index has been updated, so that you can assess the unnecessary overheads that the database may have incurred in the process.

3.4.9 SQL Transaction Logs Activity Test

Every database created on an MS SQL server is associated with a database file (.mdf) and a log data file (.ldf). All transactions to the database are logged in the log data file. The server will use the logged transactional information to restore the database after a crash, or when a transaction runs into issues and may have to be rolled back.

This test periodically monitors the I/O activity on and the growth in the size of the log file associated with each database. Using the metrics reported by this test, you can proactively detect bottlenecks while reading from or writing to the log file and excessive disk space usage by the log file.

By default, this test is disabled. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Periodically monitors the I/O activity on and the growth in the size of the log file associated with each database
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **EXCLUDE INFO** - By default, this is set to none, indicating that the test will monitor all the databases on the MS SQL server by default. To exclude specific databases from the monitoring scope of this test, provide a comma-separated list of databases in the **EXCLUDE INFO** text box. 11. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 12. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 13. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. For instance, if you set to 1:1, it means that detailed measures will be generated every time this test runs, and also every time the test detects a problem.

	14. **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 enabled/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.		
Outputs of the test	One set of results for each database on the MS SQL server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Write rate: Indicates the rate at which writes occurred on this log file.	Writes/Sec	
	Data write rate: Indicates the rate at which data was written to this log file.	KB/Sec	Ideally, the value for this measure should be low. If the value for this measure is high, use the detailed diagnosis of the Num of waits measure to identify the queries that are causing the waits to remain for a long time. You may want to finetune the queries to reduce wait time.
	I/O stall writes: Indicates the total time taken to write to this log file.	Millisecs	A high value for this measure could indicate a bottleneck while writing to the log file.By comparing the value of this measure across log files, you can identify the log file to which write operations are taking too long to complete.
	Read rate: Indicates the rate of reads from this log file.	Reads/Sec	
	Data read rate: Indicates the rate at which data was read from this log file.	KB/Sec	
	I/O stall reads: Indicates the time taken to read from this log file.	Millisecs	A high value for this measure could indicate a bottleneck while reading from the log file. By comparing the value of this measure across log files, you can identify the log file to which read operations are taking too long to complete.
	I/O stall: Indicates the total time taken for I/O to complete on this log file.	Millisecs	A high value for this measure could indicate an I/O bottleneck on this log file.

	Size on disk: Indicates the total size on disk of each logfile.	Bytes	This measure is used to determine the growth of the logfile.

3.5 SQL Mirroring Status

Database mirroring is a solution for increasing the availability of a SQL Server database. Mirroring is implemented on a per-database basis and works only with databases that use the full recovery model.

Database mirroring maintains two copies of a single database that must reside on different server instances of SQL Server Database Engine. Typically, these server instances reside on computers in different locations. Starting database mirroring on a database, initiates a relationship, known as a *database mirroring session*, between these server instances.

One server instance serves the database to clients (the *principal server*). The other instance acts as a hot or warm standby server (the *mirror server*), depending on the configuration and state of the mirroring session. When a database mirroring session is synchronized, database mirroring provides a hot standby server that supports rapid failover without a loss of data from committed transactions. A Witness is an optional instance of SQL Server that enables the mirror server to recognize when to initiate an automatic failover. Unlike the two failover partners, the witness does not serve the database. Supporting automatic failover is the only role of the witness.

When the session is not synchronized, the mirror server is typically available as a warm standby server (with possible data loss).

It is hence evident that to prevent any data loss during failover, a database mirroring session should be in the *synchronized state* and a Witness should be up and running. If one or both the aforesaid conditions are not fulfilled, data loss is bound to occur. This is why, administrators should continuously track the state of every database mirroring session and witness on a SQL server instance. This is where the **SQL Mirroring Status** test helps.

For each database on a SQL server instance on which database mirroring is enabled, this test reports the current status of the mirroring session of that database, reveals what role that database plays in the mirroring session, and the current state of the witness. This way, administrators are promptly alerted when any mirroring session or witness switches to an abnormal state.

Purpose	For each database on a SQL server instance on which database mirroring is enabled, this test reports the current status of the mirroring session of that database, reveals what role that database plays in the mirroring session, and the current state of the witness.
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for each database on the MS SQL server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**

<table>
<tr>
<td></td>
<td>Mirroring state:
Indicates the mirroring state of this database mirroring session.</td>
<td></td>
<td>The values that this measure can report and their corresponding numeric values are as follows:
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>DISCONNECTED</td><td>1</td></tr>
<tr><td>SYNCHRONIZED</td><td></td></tr>
<tr><td>SYNCHRONIZING</td><td></td></tr>
<tr><td>PENDING_FAILOVER</td><td></td></tr>
<tr><td>SUSPENDED</td><td></td></tr>
<tr><td>UNSYNCHRONIZED</td><td></td></tr>
<tr><td>SYNCHRONIZED</td><td></td></tr>
</table>
Note:
By default, this measure reports the Measure Values listed in the table above to indicate the current status of a database mirroring session. In the graph of this measure however, the same is represented using the numeric equivalents only.</td>
</tr>
<tr>
<td></td>
<td>Mirroring role state:
Indicates the role that is currently played by this database in the mirroring session.</td>
<td></td>
<td>The values that this measure can report and their corresponding numeric values are as follows:
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>PRINCIPAL</td><td>1</td></tr>
<tr><td>MIRROR</td><td>2</td></tr>
</table>
Note:
By default, this measure reports the Measure Values listed in the table above to indicate the database role. In the graph of this measure however, the same is represented using the numeric equivalents only.</td>
</tr>
<tr>
<td></td>
<td>Mirroring witness sate:
Indicates the current state of the witness in the database mirroring session.</td>
<td></td>
<td>The values that this measure can report and their corresponding numeric values are as follows:
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>UNKNOWN</td><td>1</td></tr>
<tr><td>CONNECTED</td><td>2</td></tr>
<tr><td>DISCONNECTED</td><td>3</td></tr>
</table>
Note:
By default, this measure reports the Measure Values listed in the table above to indicate the state of the witness. In the graph of this measure however, the same is represented using the numeric equivalents only.</td>
</tr>
</table>

3.5.1 SQL Mirroring Transactions Test

The principal and mirror servers communicate and cooperate as *partners* in a *database mirroring session*. The two partners perform complementary roles in the session: the *principal role* and the *mirror role*. At any given time, one partner performs the principal role, and the other partner performs the mirror role. Each partner is described as *owning* its current role. The partner that owns the principal role is known as the *principal server*, and its copy of the database is the current principal database. The partner that owns the mirror role is known as the *mirror server*, and its copy of the database is the current mirror database. When database mirroring is deployed in a production environment, the principal database is the *production database*.

Database mirroring involves *redoing* every insert, update, and delete operation that occurs on the principal database onto the mirror database as quickly as possible. Redoing is accomplished by sending a stream of active transaction log records to the mirror server, which applies log records to the mirror database, in sequence, as quickly as possible. Unlike replication, which works at the logical level, database mirroring works at the level of the physical log record. Beginning in SQL Server 2008, the principal server compresses the stream of transaction log records before sending it to the mirror server. This log compression occurs in all mirroring sessions.

If transaction log records are not sent quickly by principal server or are not applied quickly by the mirror server, then the data in the principal and mirror databases will be out of sync; this will cause significant data loss during a failover. To avoid this, administrators must keep track of the log record traffic between the principal and mirror servers, proactively detect potential slowness in mirroring, figure out the probable source of the bottleneck, and clear it to ensure synchronization between the principal and mirror databases. This is where the **SQL Mirroring Transactions** test helps.

This test tracks the transactions started on a SQL server instance, measures the rate at which transaction log data is sent to the mirror server for synchronization, and the time taken by the mirror server to apply the data. In the process, the test pinpoints bottlenecks in database mirroring and where exactly the bottlenecks lie.

Purpose	Tracks the transactions started on a SQL server instance, measures the rate at which transaction log data is sent to the mirror server for synchronization, and the time taken by the mirror server to apply the data. In the process, the test pinpoints bottlenecks in database mirroring and where exactly the bottlenecks lie.
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for each database on the MS SQL server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Queued log size: Indicates the total number of kilobytes of log that have not yet been sent to the mirror server.	KB	A high value for this measure could indicate a bottleneck on the principal server or a network congestion obstructing data flow to the mirror server. Ideally, the value of this measure should be low.
	Log bytes sent: Indicates the the rate at which log data was sent.	Bytes/sec	A high value is desired for this measure. A consistent drop in this value could indicate a processing bottleneck on the principal server.

	Log compressed bytes sent: Indicates the rate at which compressed bytes of log data was sent.	Bytes/Sec	
	Log send flow control time: Indicates the duration for which log stream messages waited for send flow control, in the last second.	Msecs	Sending log data and metadata to the mirroring partner is the most data-intensive operation in database mirroring and might monopolize the database mirroring and Service Broker send buffers. Use this counter to monitor the use of this buffer by the database mirroring session. A high value of this measure indicates that the queue in the actual layer sending the messages on the network is full. Hence this would indicate a network issue.
	Transactions: Indicates the rate at which transactions were started for the database.	Transactions/Sec	
	Transaction delay: Indicates delay in waiting for unterminated commit acknowledgement.	Msecs	High values in this counter can be a clear indicator of a bottleneck that is affecting performance and that end users are seeing a delay in their transactions.
	Avg transaction delay: Indicates ratio between transaction delay and transaction per sec.	Msecs/transaction	
	Log harden time: Indicates the time for which log blocks waited to be hardened to disk, in the last second.	Msecs	If transactions are not hardened on the log drive on the mirror fast enough and you are using high safety, the principal might have to wait for the mirror to acknowledge hardening of log records before transactions can commit, resulting in degraded performance. A high value for this measure is therefore a cause for concern.
	Log bytes received: Indicates the rate at which log bytes were received.	Bytes/Sec	A high value is desired for this measure. A consistent drop in this value could indicate a processing bottleneck on the mirror server.
	Log compressed bytes received: Indicates the rate at which compressed log data was received.	Bytes/Sec	
	Redo bytes: Indicates the number of bytes of log rolled forward on te mirror database per second.	Bytes/Sec	

	Redo queue: Indicates the total number of kilobytes of hardened log that currently remain to be applied to the mirror database to roll it forward. This is sent to the Principal from the Mirror.	KB	A low value is ideal for this measure.
	Send/Receive ack time: Indicates the time for which messages waited for acknowledgement from the partner, in the last second.	Msecs	This counter is helpful in troubleshooting a problem that might be caused by a network bottleneck, such as unexplained failovers, a large send queue, or high transaction latency. In such cases, you can analyze the value of this counter to determine whether the network is causing the problem.
	Mirrored write transactions: Indicates the number of transactions that wrote to the mirrored database and waited for the log to be sent to the mirror in order to commit, in the last second. This counter is incremented only when the principal server is actively sending log records to the mirror server.	Transactions/Sec	This counter is incremented only when the principal server is actively sending log records to the mirror server.

3.6 The MS SQL Service Layer

The **MS SQL Service** layer tracks the health of the services associated with an MS SQL server (see Figure 3.22). The following sections provide more information on these tests and the measures reported by them.

Figure 3.22: Tests mapping to the MS SQL Service layer

3.6.1 SQL Long Running Queries Test

This test reports the time taken by each database for executing queries, and also reveals which query is taking too long to execute. This way, resource-intensive queries to a database can be quickly isolated.

This test will execute only on MS SQL Server 2005 (or above).

Purpose	Reports the time taken by each database for executing queries, and also reveals which query is taking too long to exe
Target of the test	An MS SQL server 2005 (or above)
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **EXCLUDEDB** - By default, the **EXCLUDEDB** text box will be set to a comma-separated list of databases that will be excluded from the monitoring scope of this test by default. You can append more databases to this list to exclude more databases, or can remove one/more databases from the list to include them in the monitoring purview. 11. **TOP N QUERY** - By default, the number '10' is displayed in the **TOP N QUERY** text box. This implies that, by default, the detailed diagnosis of this test will display the top-10 queries to the database, in terms of resource usage. If you want the detailed diagnosis to display more or less number of top queries, set the **TOP N QUERY** parameter to a number of your choice. 12. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 13. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

<table>
<tr><td></td><td colspan="3">14. AVG ELAPSED PER EXEC - By default, to compute the Avg elapsed time and Max elapsed time of queries, this test considers the execution time of only those queries that have been running for over 10 seconds. Also, by default, the detailed diagnosis of this test will provide the details of only those queries that have been running for over 10 seconds. This is why, the AVG ELAPSED PER EXEC parameter is set to 10, by default. You can change this default setting, if required.
15. 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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each database on the MS SQL server monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Avg elapsed time:
Indicates the average time taken by queries to execute on this database.</td><td>Secs</td><td>If the value of this measure is very high, it could either indicate that the database is unable to process the queries quickly or that one/more queries to the database are taking too long to execute. Improper indexing and fragmented tables in the database are common causes for slowdowns at the database-level. Besides the above, queries that are improperly structured can also take time to execute. The longer a query executes on the database, higher would be the resource consumption of that query. It is therefore imperative that such resource-intensive queries are quickly isolated and fine-tuned, so as to prevent degradations in the performance of the database server. Using the detailed diagnosis of this measure, you can rapidly identify the resource-intensive queries to the database.</td></tr>
<tr><td>Max elapsed time:
Indicates the maximum time taken by the queries to this database.</td><td>Secs</td><td></td></tr>
</table>

3.6.2 SQL Network Test

This test monitors the availability and response time from clients by an Ms Sql database server from an external perspective.

Purpose	This test measures the status of a MS SQL server.
Target of the test	An MS SQL server
Agent deploying the test	An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the target Microsoft SQL server is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the target Microsoft SQL server is listening.

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Ms Sql server. 3. **PORT** – The port number through which the Ms Sql server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **DATABASE** - The name of the database to connect to. The default is "master". To monitor multiple databases, ensure that the database names are provided as a colon-separated list. Alternatively, you can use the semi-colon as the separator for the database names. 11. **QUERY** – The select query to execute. The default is "select * from master.dbo.spt_monitor". If the target MS SQL database server is installed as case sensitive, then the value of query parameter must be case sensitive. If multiple databases are specified in the **DATABASE** text box, then you will have to provide multiple queries here separated by a semi-colon (;) - for eg., *select * from master.dbo.spt_monitor;select * from alarm*. Every **DATABASE** being monitored, should have a corresponding **QUERY** specification. 12. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.

	13. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every Ms Sql server monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	SQL server availability: Indicates the availability of the server.	Percent	The availability is 100% when the server is responding to a request and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the database server, or because the server has not been started. The availability is 100% when the instance is responding to a request and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the database instance, or because the instance is using an invalid user account. Besides the above, this measure will report that the server is unavailable even if a connection to the database instance is unavailable, or if a query to the database fails. In this case, you can check the values of the *DB connection availability* and *Query processor availability* measures to know what is exactly causing the database instance to not respond to requests - is it owing to a connection unavailability? or is it due to a query failure?
	SQL response time: The time taken by the database to respond to a user query. This is the sum total of the connection time and query execution time.	Seconds	A sudden increase in response time is indicative of a bottleneck at the database server.
	DB connection availability: Indicates whether the database connection is available or not.	Percent	If this measure reports the value 100 , it indicates that the database connection is available. The value 0 on the other hand indicates that the database connection is unavailable. A connection to the database may be unavailable if the database is down or if the database is listening on a port other than the one configured for it in the eG manager or owing to a poor network link. If the *SQL availability* measure reports the value 0, then, you can check the value of this measure to determine whether/not it is due to the unavailability of a connection to the server.

	Query processor availability: Indicates whether the database query is executed successfully or not.	Percent	If this measure reports the value 100, it indicates that the query executed successfully. The value 0 on the other hand indicates that the query failed. In the event that the *SQL availability* measure reports the value *0*, check the value of this measure to figure out whether the failed query is the reason why that measure reported a server unavailability.
	Connection time to database server: Indicates the time taken by the database connection.	Secs	A high value could indicate a connection bottleneck. Whenever the *SQL response time* of the measure soars, you may want to check the value of this measure to determine whether a connection latency is causing the poor responsiveness of the server.
	Query execution time: Indicates the time taken for query execution.	Secs	A high value could indicate that one/more queries to the database are taking too long to execute. Inefficient/badly designed queries to the database often run for long periods. If the value of this measure is higher than that of the *Connection time* measure, you can be rest assured that long running queries are the ones causing the responsiveness of the server to suffer.
	Records fetched: Indicates the number of records fetched from the database.	Number	The value 0 indicates that no records are fetched from the database

3.6.3 SQL Sessions Test

This test monitors the availability of the MS SQL server from an internal perspective. This test returns measurements like Login/Sec, Logout/Sec, and number of user connections.

Purpose	This test measures the session related measures of an MS SQL server
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 6. **PASSWORD** - The password of the specified **USER** 7. **CONFIRM PASSWORD** - Confirm the password by retyping it. 8. **INSTANCE** – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**. 12. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every MS SQL server monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Logins:** This value indicates the total number of logins per second.	Logins/Sec	A high value here indicates an increase in the rate of user logins into the SQL server. An unusual increase in the login rate may be an indicator of abnormal activity of database applications.
	Logouts: Indicates the total number of logouts per second.	Logouts/Sec	A high value here indicates increase in rate of users logging out of SQL server. An unusually large number of logins and logouts can occur due to application retries being caused by errors during database access.
	Current connections: Indicates the number user connections to the server at an instant.	Number	As each user connection consumes some memory, a large number of user connections could affect throughput. By tracking the history of user connections, a database administrator can set the maximum expected number of concurrent users accordingly.
	Logical connections: Indicates the number of logical connections to the server.	Number	The main purpose of logical connections is to service multiple active result sets (MARS) requests. For MARS requests, every time that an application makes a connection to SQL Server, there may be more than one logical connection that corresponds to a physical connection. When MARS is not used, the ratio between physical and logical connections is 1:1. Therefore, every time that an application makes a connection to SQL Server, logical connections will increase by 1

3.6.4 SQL Backup Details Test

To prevent the data loss that may occur due to the sudden failure of a SQL database, administrators are often advised to schedule the automatic backup of the databases on the Microsoft SQL server. It is recommended that these backup jobs are scheduled to occur frequently (say, once a day), so that the backup is always in sync with the data backed up. Failed or unusually fast backup jobs should also be detected quickly and marked for closer scrutiny, as this can cause data non-sync and increase the risk of data loss when disaster strikes. This is why, administrators will find the **SQL Backup Details** test very helpful! This test monitors the backup jobs configured for every SQL database. In the process, the test reports the type of backup that is configured, when the last backup job ran, and how long it took. This way, administrators can quickly identify databases that are not backed up as frequently as they would like them to be, rapidly detect backup jobs that may have failed to run as per schedule, and can pinpoint those databases where the last backup was suspiciously fast. With the help of these inferences, administrators can fine-tune backup schedules and can troubleshoot backup failures.

Purpose	Monitors the backup jobs configured for every SQL database. In the process, the test reports the type of backup that is configured, when the last backup job ran, and how long it took. This way, administrators can quickly identify databases that are not backed up as frequently as they would like them to be, rapidly detect backup jobs that may have failed to run as per schedule, and can

	pinpoint those databases where the last backup was suspiciously fast. With the help of these inferences, administrators can fine-tune backup schedules and can troubleshoot backup failures.		
Target of the test	An MS SQL server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 6. **PASSWORD** - The password of the specified **USER** 7. **CONFIRM PASSWORD** - Confirm the password by retyping it. 8. **INSTANCE** – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every database on the MS SQL server monitored		
	Measurement	**Measurement Unit**	**Interpretation**

<table>
<tr><td rowspan="3">Measurements made by the test</td><td>Backup type:
Indicates the type of backup that is configured for this database.</td><td></td><td>The values that this measure can report and their corresponding numeric values are listed below:<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Full</td><td>1</td></tr><tr><td>Diff-Database</td><td>2</td></tr><tr><td>T-Log</td><td>3</td></tr><tr><td>File-File Group</td><td>4</td></tr><tr><td>Diff-File</td><td>5</td></tr><tr><td>Partial</td><td>6</td></tr><tr><td>Diff-Partial</td><td>7</td></tr></table>Note:
By default, this measure will report one of the Measure Values listed in the table above to indicate the backup type of a database. However, in the graph of this measure, the same is represented using the numeric equivalents only.</td></tr>
<tr><td>Time taken for last backup operation:
Indicates the time it took for the last backup operation on this database to complete.</td><td>Mins</td><td>If the value of this measure is very low for a database, it warrants a probe, as it could be owing to the failure of the backup job.</td></tr>
<tr><td>Interval since last backup:
Indicates the number of days that have elapsed since this database was last backed up.</td><td>Days</td><td>If the value of this measure is over 1 day, it could imply one of the following:
The database is not been backed up regularly, or;
The last backup job on the database failed
Either way, further investigation may be required to determine the real reasons.</td></tr>
</table>

3.6.5 Tests Disabled by Default

The tests that have been discussed in this section have been disabled by default for this layer. To enable any of these tests, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose the test that you want to enable from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

3.6.5.1 SQL Query Wait Activity Test

This test monitors the wait types on the MS SQL server, and reports the number and duration of waits of each type.

This test is applicable only to Microsoft SQL Server 2005 (and above).

Purpose	Monitors the wait types on the MS SQL server, and reports the number and duration of waits of each type
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **EXCLUDE INFO** - By default, this is set to none, indicating that the test will monitor all the wait types active on the MS SQL server by default. To exclude specific wait types from the monitoring scope of this test, provide a comma-separated list of wait types in the **EXCLUDE INFO** text box. 11. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 12. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 13. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. For instance, if you set to 1:1, it means that detailed measures will be generated every time this test runs, and also every time the test detects a problem.

	14. **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 enabled/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.
Outputs of the test	One set of results for each wait type on the MS SQL server instance being monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Number of waits: Indicates the total number of waits for this wait type.	Number	The detailed diagnosis of this measure provides the query that caused the waits; you may want to fine-tune the queries to reduce the number of waits.
	Avg wait time: Indicates the average wait time for this wait type.	Secs	Ideally, the value for this measure should be low. If the value for this measure is high, use the detailed diagnosis of the Num of waits measure to identify the queries that are causing the waits to remain for a long time. You may want to finetune the queries to reduce wait time.
	Max wait time: Indicates the maximum wait time for this wait type.	Secs	Comparing the value of this measure across wait types will enable you to accurately isolate the wait type that is responsible for the longest waits.

3.6.5.2 SQL Open Cursors Test

Cursor is a database object used by applications to manipulate data in a set on a row-by-row basis, instead of the typical SQL commands that operate on all the rows in the set at one time.

If cursors executing on an MS SQL server take too long a time to execute, it could drain critical server resources, and could severely hamper server performance. With the help of this test, you can easily track the number of cursors that each application is executing on the MS SQL server and the time they take to execute, so that resource-intensive cursors and the applications executing them can be isolated.

Purpose	Tracks the number of cursors that each application is executing on the MS SQL server and the time they take to execute
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **EXCLUDE INFO** - By default, this is set to none, indicating that the test will monitor all the applications that are executing cursors on the MS SQL server by default. To exclude specific applications from the monitoring scope of this test, provide a comma-separated list of applications in the **EXCLUDE INFO** text box. 11. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 12. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

	13. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. For instance, if you set to 1:1, it means that detailed measures will be generated every time this test runs, and also every time the test detects a problem. 14. **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 enabled/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.		
Outputs of the test	One set of results for each application that is executing cursors on the MS SQL server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Number of cursors: Indicates the total number of cursors that this application is currently executing on the server.	Number	Compare the value of this measure across applications to know which application is executing the maximum number of cursors on the server. Use the detailed diagnosis of this measure to know the details of the cursors. This includes the name of each cursor, which user created the cursor, when, how long has it been since the cursor was created, the number of disk reads and writes performed by the cursor, and the worker time of the cursor. Long running cursors and I/O intensive cursors can thus be isolated.
	Avg cursor opentime: The average time for which this application was executing cursors on the server.	Secs	Ideally, the value of this measure should be low. A high value indicates that the cursors executed by this application have been running for too long a time on the server; besides eroding server resources, this phenomenon can greatly degrade server performance. Compare the value of this measure across applications to identify which application is executing long running cursors.
	Max cursor opentime: Indicates the maximum time for which cursors executed by this application have remained open.	Secs	

3.6.5.3 SQL Session Activity Test

This test monitors the sessions initiated by each application on the MS Sql server, and reports the level of activity each application has imposed on the server.

Purpose	Monitors the sessions initiated by each application on the MS Sql server, and reports the extent of load that each application has imposed on the server
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **EXCLUDE INFO** - By default, this is set to none, indicating that the test will monitor all the applications that are using the MS SQL server by default. To exclude specific applications from the monitoring scope of this test, provide a comma-separated list of applications in the **EXCLUDE INFO** text box. 11. **TOP-N** - By default, this is set to 10, indicating that the detailed diagnosis of this test will list only the top-10 sessions based on the elapsed time of the sessions. To view more or less number of top-n records in the detailed diagnosis, provide any other number of your choice in the **TOP-N** text box. 12. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 13. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 14. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. For instance, if you set to 1:1, it means that detailed measures will be generated every time this test runs, and also every time the test detects a problem.

	15. **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 enabled/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.		
Outputs of the test	One set of results for each application using the MS SQL server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Number of processes: Indicates the total number of processes executed currently by this application on the server.	Number	The detailed diagnosis of this measure is used to view the complete details of the session. This includes the session ID, the user who has initiated the session, the current status of the session, the elapsed time, CPU time, open cursor count,active request count,blocking/blocked request count, the count of disk reads and writes, and the count of logical reads performed by the session. These information enables you to know how resource intensive the session is and how much I/O load has been generated by the session.
	Open transaction count: Indicates the total number of transaction executed currently by the application on the server.	Number	In terms of a server overload, you can compare the value of this measure across the applications to identify the sessions that are responsible for increasing the transaction load on the server.
	Open cursor count: Indicates the total number of cursors executed currently by the application on the server.	Number	In terms of a server overload, you can compare the value of this measure across the applications to identify the sessions that are responsible for increasing the transaction load on the server.
	Active request count: Indicates the rate at which the application is sending requests to the server.	Reqs/Sec	
	Blocking request count: Indicates the number of requests of this application that are currently being blocked.	Number	Ideally, the value of this measure should be 0. If the measure reports a non-zero value, then you can use the detailed diagnosis of the *Blocked processes* measure of the *MsSqlSysProcesses* test to identify the exact query that is responsible for the blocking.

	Open resultset count: Indicates the number of resultsets that are currently open on the server for this application.	Number	
	Blocked request count: Indicates the number of requests of the application that are currently blocking.	Number	Ideally, the value of this measure should be 0. If the measure reports a non-zero value, then you can use the detailed diagnosis of the *Blocked processes* measure of the *MsSqlSysProcesses* test to identify the exact query that is responsible for the blocking.

3.6.5.4 SQL Error Log Test

This test reports the number and type of errors logged in the SQL server error logs.

Purpose	Reports the number and type of errors logged in the SQL server error logs
Target of the test	An MS SQL server 2005 (or above)
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. FILEPATH - Enter the full path to the log file to be monitored.
5. ISUTF16 - If the error log file to be monitored is encoded with UTF-16, then, set the ISUTF16 flag to Yes. By default, this flag is set to No.
6. ISPASSIVE – If the value chosen is YES, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
7. DD FREQUENCY - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. For this test, the DD FREQUENCY is set to 1:1 by default. This indicates that, by default, detailed measures will be generated every time this test runs and also every time the test detects a problem.
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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the MS SQL server being monitored</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

	DeadLocks: Indicates the number of deadlocks on the MS SQL server since the last measurement period.	Number	The SQL error log will capture deadlock conditions only if trace is enabled on the SQL server. This means that if the value of this measure is 0, it could imply one of the following: • Tracing is not enabled, and therefore, the SQL error log is not able to capture any deadlock conditions; if you want, you can enable tracing by runing the following command from the SQL prompt: *DBCC TRACEON(1222,-1)* • Tracing is enabled, but no deadlock has occurred. **This is a sign of good health.** If this measure reports a non-zero value on the other hand, it is a cause for concern, as it indicates that one/more deadlocks have occurred. In this case, you can use the detailed diagnosis of this test to know more information about the deadlocks.
	Informational messages: Indicates the number of informational messages that were captured by the error log during the last measurement period.	Number	Messages with a severity level of 0 to 10 are informational messages and not actual errors.
	User errors: Indicates the number of user errors that were captured by the error log during the last measurement period.	Number	The value of this measure indicates the number of errors with a severity level between 11 and 16. Such errors are generated as a result of user problems and can be fixed by the user.
	Software errors: Indicates the number of software errors captured by the error log during the last measurement period.	Number	All errors with severity levels 17 to 19 will be counted as software errors. Severity level 17 indicates that SQL Server has run out of a configurable resource, such as locks. Severity error 17 can be corrected by the DBA, and in some cases, by the database owner. Severity level 18 messages indicate non-fatal internal software problems. Severity level 19 indicates that a nonconfigurable resource limit has been exceeded.

	Fatal or system errors: Indicates the number of fatal or system errors experienced by the target MS SQL server during the last measurement period.	Number	Errors with severity levels 20 to 25 are typically categorized as fatal/system. Severity level 20 indicates a problem with a statement issued by the current process. Severity level 21 indicates that SQL Server has encountered a problem that affects all the processes in a database. Severity level 22 means a table or index has been damaged. To try to determine the extent of the problem, stop and restart SQL Server. If the problem is in the cache and not on the disk, the restart corrects the problem. Otherwise, use DBCC to determine the extent of the damage and the required action to take. Severity level 23 indicates a suspect database. To determine the extent of the damage and the proper action to take, use the DBCC commands. Severity level 24 indicates a hardware problem. Severity level 25 indicates some type of system error.
	Warning messages Indicates the number of warning messages found in the SQL error logs during the last measurement period.		
	Other errors: Indicates the number of other errors captured in the MS SQL server during the last measurement period.	Number	

3.6.5.5 SQL Job Details Test

This test measures the status of all the jobs executing on an MS SQL server.

Purpose	Measures the status of all the jobs executing on an MS SQL server
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide in this text box, the name of a SQL user with the **CONNECT SQL** and **PUBLIC** roles and who has access to the *SysJobs* and *SysJobHistory* tables of the **MSDB** database. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **DDFORSUCCESS** - By default, detailed diagnosis information will be available for the successful, failed and cancelled jobs reported by this test. Accordingly, the **DDFORSUCCESS** test flag is set to **Yes** by default. If you do not want detailed diagnosis for the successful jobs measures then set this flag to **No**. 12. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

	13. **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 enabled/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.
Outputs of the test	One set of results for the MS SQL server monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Total SQL jobs: Indicates the total number of jobs that are currently running.	Number	
	Successful SQL jobs: Indicates the jobs that have been successfully executed.	Number	
	Failed SQL jobs: Indicates the jobs that have not been successfully executed.	Number	Ideally, the value for this measure should be zero. On other hand, if this measure reports a non-zero value, then you can use detailed diagnosis to identify the name of the job that has failed, the server name and the step at which the job failed.
	Cancelled SQL jobs: Indicates the jobs that were terminated by user interference.	Number	Ideally, the value for this measure should be zero. However, if this measure reports a non-zero value, then you can use detailed diagnosis to identify the name of the job that has been cancelled, server name and the step at which the job was cancelled.

3.6.5.6 SQL Job Status Test

This test reports the current status of configured SQL jobs.

Purpose	Reports the current status of configured SQL jobs
Target of the test	An MS SQL server

Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide in this text box, the name of a SQL user with the **CONNECT SQL** and **PUBLIC** roles and who has access to the *SysJobs* and *SysJobHistory* tables of the **MSDB** database. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **JOBNAME** - Specify the job to be monitored in the **JOBNAME** text box. If multiple jobs are to be monitored, then provide a comma-seperated list of job names. 11. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 12. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

<table>
<tr><td></td><td colspan="3">13. 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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every job that has been configured for monitoring</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Number of run times for the SQL job:
Indicates the number of times the job has run.</td><td>Number</td><td></td></tr>
<tr><td>Avg duration of SQL job:
Indicates the average time taken by the job to execute.</td><td>Secs</td><td>By comparing the value of this metric across the current jobs, you can accurately identify the job that is taking too long to complete. Also, note that if the value of the Number of run times for the SQL job measure is equal to zero, then the Avg duration of the SQL job measure will not appear in the eG user interface.</td></tr>
<tr><td>Status of the SQL job:
Indicates the status of this job.</td><td>Number</td><td>If the value of this measure is 0, it means that the job has failed. In such a case, use the detailed diagnosis of the Number of run times for the SQL job measure to determine the root-cause of the failure. The value 3 for this measure indicates that the job was cancelled, and the value 1 denotes that the job executed successfully. However, note that if the value of the Number of run times for the SQL job measure is equal to zero, then the Status of the SQL job measure will not appear in the eG user interface.</td></tr>
</table>

3.6.5.7 SQL Applications Test

Sometimes the database performs poorly due, not to blocking, but to particularly heavy loads. Often the DBA will determine that the database simply cannot support the work that it is being asked to do and maintain adequate performance. This does not necessarily mean it is time to create more indexes or throw more hardware at the problem. One cannot always assume that periods of high utilization represent legitimate work. There could be problems in the

applications that are running, or even problems caused by the user. Maybe the application has a data paging functionality, but the user has opted to receive the entire 100,000 row DataSet every time, even though she/he has applied a sort which gives her the one row she needs first with each query. Regardless, it is important to identify performance issues and eliminate them. The MsSqlApplications test identifies which program has more connections open to (i.e., processes running in) the SQL database. Simply looking at the CPU cycles taken up by a process will not indicate which of these processes has been most active recently. For example, the SQL Server internal processes may have been running for days and will probably always show up as the processes that have taken the most CPU time since the database booted up. Hence, it is more helpful to find processes that have used lots of CPU for the majority of the time that they have been connected. This value represents how "expensive" a process is with respect to the SQL database server. For each program that connects to the database server, the MsSqlApplications test reports the total CPU cycles for each second that the program is connected to the database. This value, represented by the *CPU cycles rate* measure, is an aggregate of all the CPU cycles consumed by every instance of the program while it is connected to the database server. The *Avg CPU cycles rate* measure represents the *CPU cycles rate* averaged across the number of processes in the database for the program under consideration. The *Avg CPU cycles rate* quantifies how bad a program is compared to the others, by dividing the *CPU cycles rate* by the number of connected instances. A high value for this value would indicate that every instance of the program was CPU-intensive. A lower value would indicate that the program may have some instances that cause performance problems, but also has instances that are mostly idle.

Purpose	To determine the most expensive programs/processes in an SQL database server
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 7. **PASSWORD** - The password of the specified **USER** 8. **CONFIRM PASSWORD** - Confirm the password by retyping it. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

<table>
<tr><td></td><td>
11. EXCLUDEPATTERN - Provide a comma-separated list of programs/processes on the SQL server that need to be excluded from monitoring. The default value is none, indicating that all processes are monitored by default. To make sure that the test ignores a few processes, specify the process names as a comma-separated list. For example: SQL_Query_Analyzer,jTDS. You can also use wild card patterns in your specification - for instance, SQL*,*TDS,Microsoft*.

12. ISNTLMV2 - In some Windows networks, NTLM (NT LAN Manager) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 (“NTLMv2”) was concocted to address the security issues present in NTLM. By default, the ISNTLMV2 flag is set to No, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to Yes if NTLMv2 is enabled on the target host.

13. DD FREQUENCY - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is 2:1. 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.

14. 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 enabled/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.
</td></tr>
<tr><td>Outputs of the test</td><td>One set of results for every program on the MS SQL server monitored</td></tr>
<tr><td>Measurements made by the test</td><td>
<table>
<tr><th>Measurement</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td>Number of processes:
Indicates the number of database processes associated with a specific program.</td><td>Number</td><td>A comparison of this value across programs will indicate which program is initiating most connections to the database. Comparison of this value over time can provide indications of potential changes in database activity characteristics of a program.</td></tr>
<tr><td>CPU cycles rate:
Indicates the number of CPU cycles consumed by all processes of a program, per minute of login.</td><td>Cycles/sec</td><td>The higher the value, the more CPU resources that the program is taking in the database.</td></tr>
</table>
</td></tr>
</table>

	Avg CPU cycles rate: Indicates the number of CPU cycles consumed by a process of a program, per minute of login.	Cycles/Conn	This value is the ratio of the *CPU cycles rate* to the number of processes for a program.

The detailed diagnosis of the *CPU cycles rate* measure of this test provides details of the most expensive queries to the database - i.e., what host is a program running from, who is running it, and what application is running it, which database the program is accessing, etc (see Figure 3.23).

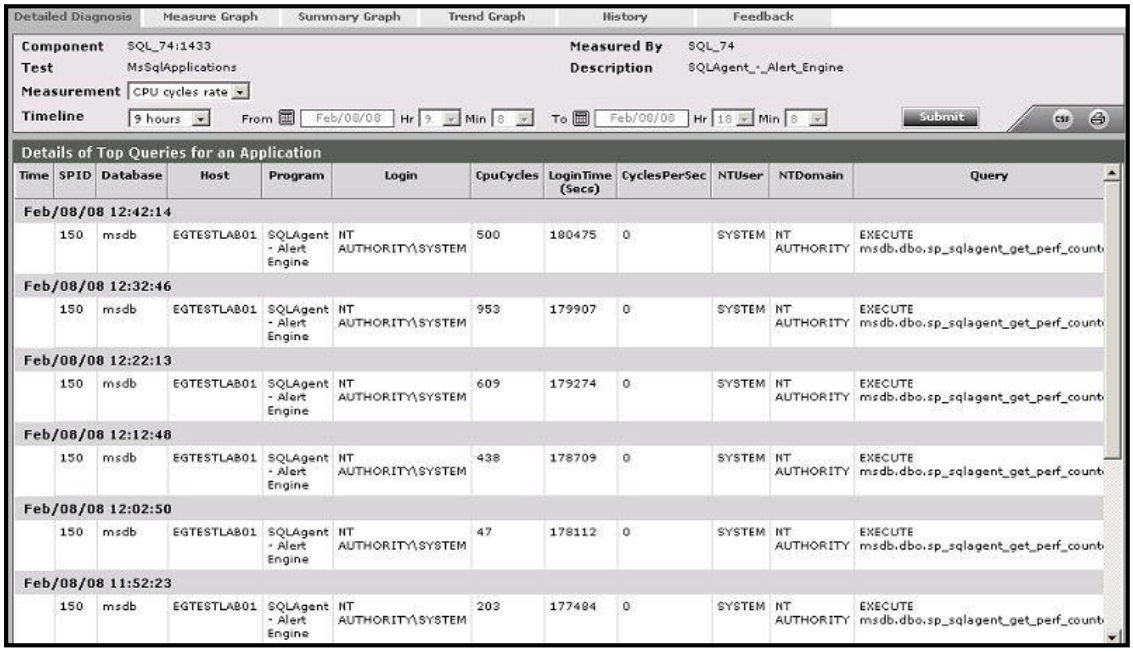


Figure 3.23: The detailed diagnosis of the CPU cycles rate measure

3.6.5.8 SQL Waits Test

This test reports key statistics pertaining to wait status. This test is specific to MS SQL Server 2005 (or above), and will hence not report any measure for any of the other versions of the MS SQL server.

Purpose	Reports key statistics pertaining to wait status
Target of the test	An MS SQL server 2005 (or above)
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** – The IP address of the MS SQL server. 3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433. 4. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 5. **SSL** - By default, the **SSL** flag is set to **No**, indicating that the target MS SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled MS SQL server, set the **SSL** flag to **Yes**. 6. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**. 7. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every type of wait on the MS SQL Server 2005 (or above) being monitored.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Avg wait time: Indicates the average duration of this wait type.	Secs	If a particular wait type is found to have persisted for a long time, it could indicate a processing overhead.
	Waits in progress: Indicates the number of processes currently waiting on this wait type.	Number	Closely monitoring *Waits in progress* along with *Avg wait time* over a period of time will reveal wait types that are locking critical system resources.
	Waits started: Indicates the number of waits started per second of this wait type.	Waits/sec	
	Cumulative waits: Indicates the percentage of time during the last measurement period wait events of this type occurred.	Percent	Compare the value of this measure across wait types to know which type of waits have occurred frequently during the last measurement period.

3.6.5.9 SQL Database Log Test

This test monitors the usage of the database logs on the MS SQL server.

<table>
<tr><td>Purpose</td><td colspan="3">Reports key statistics pertaining to wait status</td></tr>
<tr><td>Target of the test</td><td colspan="3">An MS SQL server</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST – The IP address of the MS SQL server.
3. PORT - The port number through which the MS SQL server communicates. The default port is 1433.
4. INSTANCE - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the INSTANCE parameter.
5. SSL - By default, the SSL flag is set to No, indicating that the target MS SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled MS SQL server, set the SSL flag to Yes.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for an MS SQL server instance being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>MS SQL log file size:
Indicates the size of all the log files in the database during the last measurement period.</td><td>KB</td><td>An unusually high value may indicate a sudden increase in the size of the log file of the SQL server.</td></tr>
<tr><td>MS SQL log file usage:
Indicates the percentage of log file space that is in use.</td><td>Percent</td><td></td></tr>
</table>

3.6.5.10 SQL Index Fragmentation Test

Fragmentation exists when indexes have pages in which the logical ordering, based on the key value, does not match the physical ordering inside the data file. All leaf pages of an index contain pointers to the next and the previous pages in the index. This forms a doubly linked list of all index/data pages. Ideally, the physical order of the pages in the data file should match the logical ordering. Overall disk throughput is increased significantly when the physical ordering matches the logical ordering of the data. This leads to much better performance for certain types of queries. When the physical ordering does not match the logical ordering, disk throughput can become less efficient, because the disk head must move back and forth to gather the index pages instead of scanning forward in one direction. This is how fragmentation affects I/O performance.

The first step to resolving the performance threat posed by fragmented indexes is to identify which indexes are fragmented. The **SQL Index Fragmentation** test helps in this regard. This test scans the indexes on an MS SQL server for high and very high levels of fragmentation, and reports the count of fragmented indexes. Using the detailed diagnosis capability of the test, you can also quickly drill down to the specific indexes that have been fragmented. You can thus proceed to defragment/rebuild the affected indexes, so as to increase disk throughput and improve overall SQL performance.

Purpose	Scans a pre-configured index sample for high and very high levels of fragmentation and reports the count of fragmented indexes.
Target of the test	An MS SQL server
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed. **As this test, if executed frequently, may increase the processing overheads of the eG agent, It is recommended that you run this test less frequently - say, once a day (24 hrs).** 2. **HOST** – The host for which the test is to be configured 3. **PORT** - The port on which the server is listening 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 6. **PASSWORD** - The password of the specified **USER** 7. **CONFIRM PASSWORD** - Confirm the password by retyping it. 8. **INSTANCE** – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter. 9. **DOMAIN** - By default, *none* is displayed in the DOMAIN text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the DOMAIN can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the DOMAIN text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the USER name and PASSWORD that you provide should be that of a user authorized to access the monitored SQL server. 10. **OBJECT NAME** - Specify a comma-separated list of tables, the indexes of which need to be checked for fragmentation. Every table name should be specified in the following format: *<DisplayName>:<schema_name>.<table_name>*, where *schema_name* refers to the name of the table owner, and *table_name* refers to the name of the table. The *DisplayName* in your specification will appear as the descriptor of this test. For instance, to monitor the indexes of the *alarm* and *history* tables owned by user *admin*, your specification would be: *AlarmMon1:admin.alarm,AlarmMon2:admin.history*. To monitor all tables in a schema, the specification would be of the following format: *<DisplayName>:<schema_name>.**. For example, to monitor all the tables in the *admin* schema, your specification would be: *AlarmMon:admin.**.

	11. **QUERYTIMEOUT** - Specify the time period upto which a query has to wait to obtain the required result set from the database in the **QUERYTIMEOUT** text box. If the query is not successful or if the query waits for a time period exceeding the specified time limit, the test will automatically kill the query. 12. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 13. **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 enabled/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. 14. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every *DisplayName* configured for the **OBJECT NAME** parameter of this test		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**

	Highly fragmented SQL indexes: Indicates the number of highly fragmented indexes.	Number	If 30% - 49% of an index is found to be fragmented, then such an index is counted as a highly fragmented index. Ideally, the value of this measure should be 0. A high value indicates high index fragmentation. High levels of fragmentation can cause disk I/O to mount, queries to run for long periods, and the overall performance of the database server to deteriorate. Use the detailed diagnosis of this measure to identify highly fragmented indexes. Once the affected indexes are isolated, take the necessary steps to correct the fragmentation. Towards this end, SQL provides the following statements: l. DROP INDEX followed by CREATE INDEX m. CREATE INDEX WITH DROP_EXISTING n. DBCC INDEXDEFRAG o. DBCC DBREINDEX

	Very highly fragmented SQL indexes: Indicates the number of indexes that are very highly fragmented.	Number	If over 50% of an index is found to be fragmented, then such an index is counted as a highly fragmented index. Ideally, the value of this measure should be 0. A high value indicates high index fragmentation. High levels of fragmentation can cause disk I/O to mount, queries to run for long periods, and the overall performance of the database server to deteriorate. Use the detailed diagnosis of this measure to identify highly fragmented indexes. Once the affected indexes are isolated, take the necessary steps to correct the fragmentation. Towards this end, SQL provides the following statements: p. DROP INDEX followed by CREATE INDEX q. CREATE INDEX WITH DROP_EXISTING r. DBCC INDEXDEFRAG s. DBCC DBREINDEX

3.6.5.11 SQL Table Size Test

When faced with a disk space crunch on their critical SQL servers, administrators may want to know which databases are hogging the disk space, and which tables on each database have grown beyond permissible limits. The **SQL Table Size** test provides administrators with this information. The test auto-discovers the databases with tables that exceed a configured size limit, and reports the count of such 'large sized tables' for each database. You can use the detailed diagnosis of the test to know which tables in a database are of a large size.

Purpose	Auto-discovers the databases with tables that exceed a configured size limit, and reports the count of such 'large sized tables' for each database
Target of the test	An MS SQL server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 5. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box. 6. **PASSWORD** - The password of the specified **USER** 7. **CONFIRM PASSWORD** - Confirm the password by retyping it. 8. **INSTANCE** – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter. 9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 10. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 11. **TABLE SIZE GB** - The test will report the count of those tables that are of a size greater than the value (in GB) specified here. 12. **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 enabled/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. 13. **ISPASSIVE –** If the value chosen is **Yes**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every database containing tables that are of size greater than the configured **TABLE SIZE GB**		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Table Count: Indicates the number of tables in this database that are currently of a size greater than the value (in GB) configured against **TABLE SIZE GB**.	Number	Use the detailed diagnosis of this measure to know which tables in a database are consuming the maximum disk space.

3.7 The MS SQL Application Dashboard

In order to ascertain how well an application is/has been performing, analysis of the performance of the **System** and **Network** layers of that application alone might not suffice. A closer look at the health of the **Application Layers** is also necessary, so as to promptly detect instantaneous operational issues with the target application, and also proactively identify persistent problems or a consistent performance degradation experienced by the application. To provide administrators with such in-depth insights into overall application performance and to enable them to accurately isolate the root-cause of any application-level slowdown, eG Enterprise offers the **Application Dashboard**. Each of the critical applications monitored by eG Enterprise is accompanied by an exclusive application dashboard. The contents of the dashboard will therefore primarily vary depending upon the application being monitored. Figure 3.24 for instance depicts the **Application Dashboard** of a **MS SQL application**.

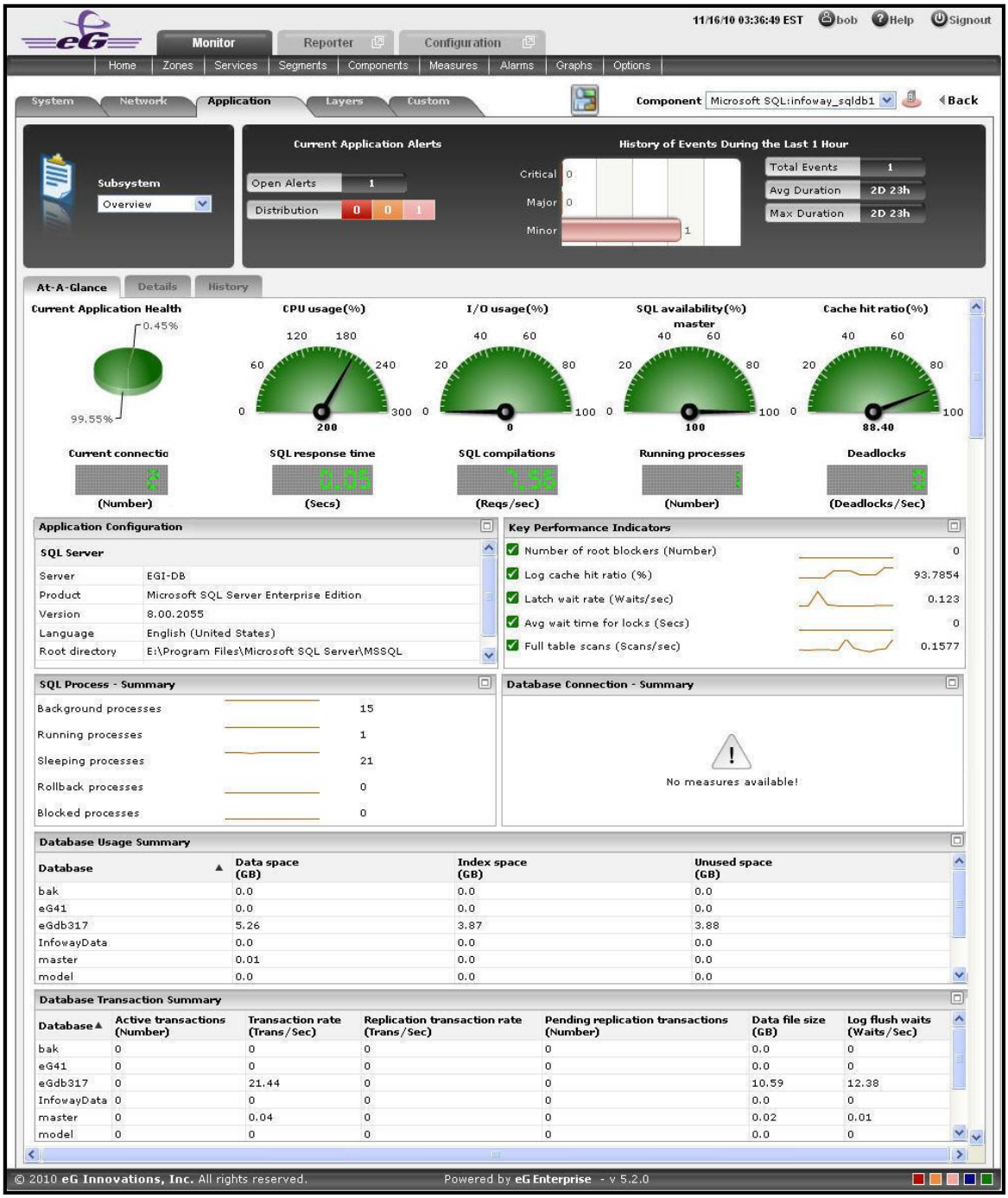


Figure 3.24: The Application Dashboard of a MS SQL application

In addition, like the **System** and **Network** dashboards, the contents of the **Application** dashboard too are further governed by the **Subsystem** chosen from Figure 3.24. By default, the **Overview** option is chosen from the **Subsystem** list. If need be, you can change this default setting by picking a different option from the **Subsystem** list. The sections that follow will discuss each of the **Subsystems** offered by the sample **MS SQL application dashboard** shown in Figure 3.24 above.

3.7.1 Overview

The **Overview** dashboard of a MS SQL application provides an all-round view of the health of the MS SQL application being monitored, and helps administrators pinpoint the problem areas. Using this dashboard therefore, you can determine the following quickly and easily:

- Has the application encountered any issue currently? If so, what is the issue and how critical is it?
- How problem-prone has the application been during the last 24 hours? Which application layer has been badly hit?
- Has the administrative staff been able to resolve all past issues? On an average, how long do the administrative personnel take to resolve an issue?
- Are all the key performance parameters of the application operating normally?
- What is the Application configuration of the MS SQL application?
- How many SQL Processes are running?
- What is the Database Usage? What is the Data space and how much Unused space is available in the MS SQL application with respect to each Database?
- How effective is the Database Transaction? What is the Transaction rate of each Database? Are the transactions for each Database behaving normally or is there any abnormal transactional behavior that has been reported during a particular time period?

The contents of the **Overview Dashboard** have been elaborated on hereunder:

1. The **Current Application Alerts** section of Figure 3.24 reveals the number and type of issues currently affecting the performance of the MS SQL application that is being monitored. To know more about the current issues, click on any cell against **Distribution** that represents the problem priority of interest to you; the details of the current problems of that priority will then appear as depicted by Figure 3.25.

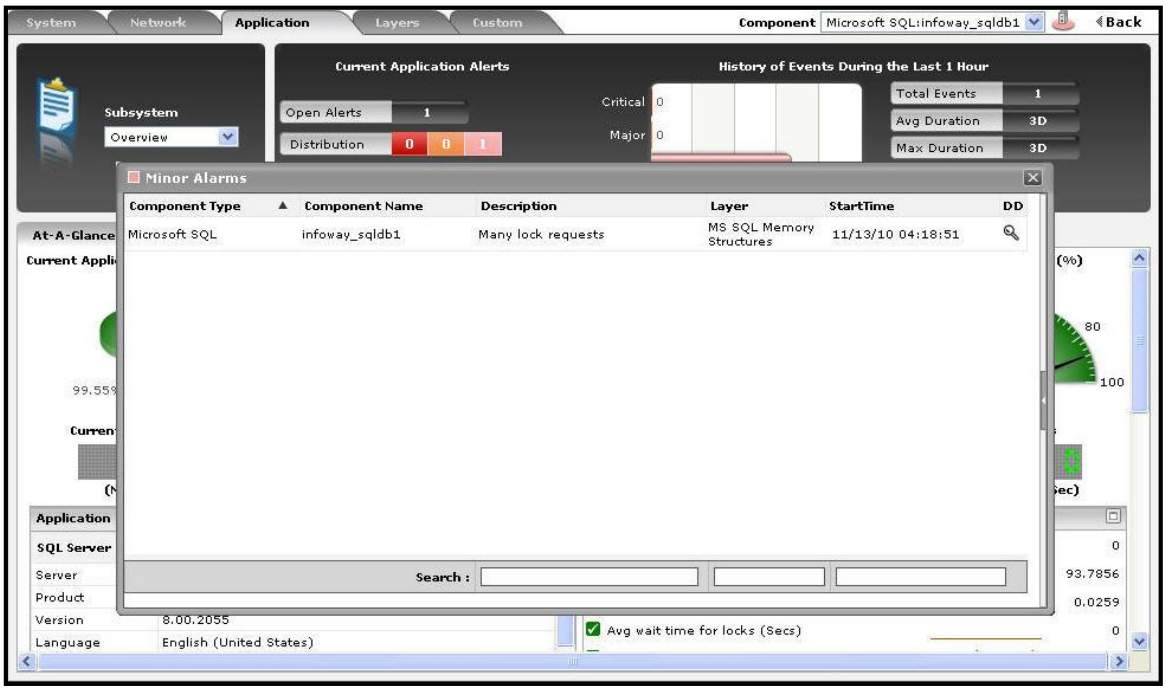


Figure 3.25: Viewing the current application alerts of a particular priority

2. If the pop-up window of Figure 3.25 reveals too many problems, you can use the **Search** text boxes that have been provided at the end of the **Description**, **Layer**, and **StartTime** columns to run quick searches on the contents of these columns, so that the alarm of your interest can be easily located. For instance, to find the alarm with a specific description, you can provide the whole/part of the alarm description in the text box at the end of the **Description** column in Figure 3.25; this will result in the automatic display of all the alarms with descriptions that contain the specified search string.

3. To zoom into the exact layer, test, and measure that reported any of the listed problems, click on a particular alarm in the **Alarms** window of Figure 3.25. Doing so will introduce an **Alarm Details** section into the **Alarms** window (see Figure 3.26), which provides the complete information related to the problem clicked on. These details include the **Site** affected by the problem for which the alarm was raised, the test that reported the problem, and the last measure that was reported will be reported in the **Last Measures**.

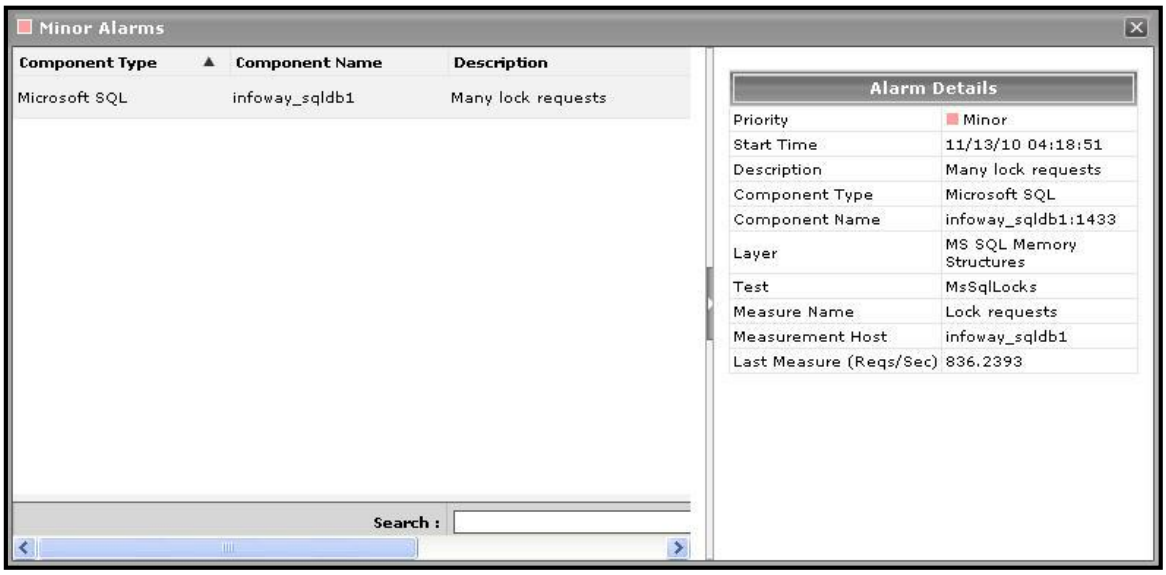


Figure 3.26: Additional alarm details

4. While the list of current issues faced by the application serves as a good indicator of the current state of the application, to know how healthy/otherwise the application has been over time, a look at the problem history of the application is essential. Therefore, the dashboard provides the **History of Events** section; this section presents a bar chart, where every bar indicates the number of problems of a particular severity, which was experienced by the MS SQL application during the last 1 hour (by default). Clicking on a bar here will lead you to Figure 3.27 which provides a detailed history of problems of that priority. Alongside the bar chart, you will also find a table displaying the average and maximum duration for problem resolution; this table helps you determine the efficiency of your administrative staff.

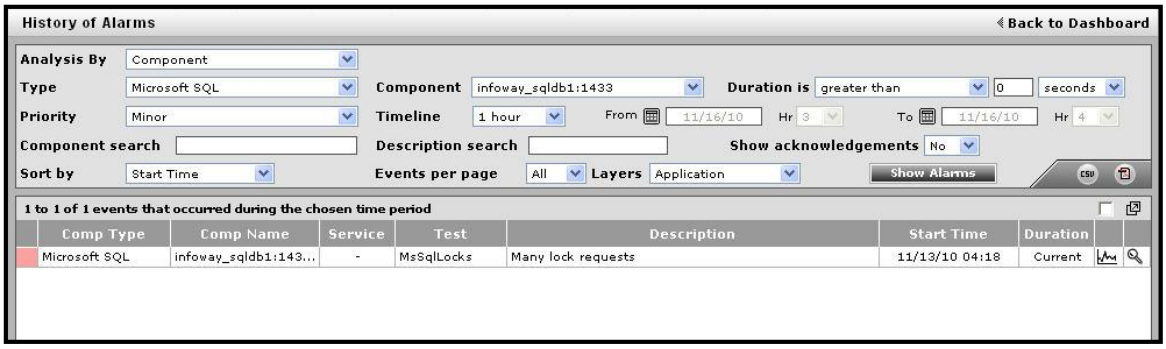


Figure 3.27: The problem history of the target application

5. If required, you can override the default time period of 1 hour of the event history, by following the steps below:
   - Click the button at the top of the dashboard to invoke the **Dashboard Settings** window.
   - Select the **Event History** option from the **Default timeline for** list.
   - Set a different default timeline by selecting an option from the **Timeline** list.
   - Finally, click the **Update** button.
6. Back in the dashboard, you will find that the **History of Events** section is followed by an **At-A-Glance** section; this section, using pie charts, digital displays and gauge charts, reveals, at a single glance, the current status of some of the critical metrics and key components of the MS SQL application. For instance, the **Current Application Health**

pie chart indicates the current health of the application by representing the number of application-related metrics that are in various states. Clicking on a slice here will take you to Figure 3.27 that provides a detailed problem history.

7. The dial and digital graphs that follow provide you with quick updates on the status of a pre-configured set of resource usage-related metrics pertaining to the MS SQL application. If required, you can configure the dial graphs to display the threshold values of the corresponding measures along with their actual values, so that deviations can be easily detected. For this purpose, do the following:

   - Click the  button at the top of the dashboard to invoke the **Dashboard Settings** window.
   - Set the **Show Thresholds** flag in the window to **Yes**.
   - Finally, click the **Update** button.

   You can customize the **At-A-Glance** tab page further by overriding the default measure list for which dial/digital graphs are being displayed in that tab. To achieve this, do the following:

   - Click on the  icon at the top of the **Application Dashboard**. In the **Dashboard Settings** window that appears, select **Application** from the **Module** list, and **Overview** from the **Sub-System** list.
   - To add measures for the dial graph, pick the **Dial Graph** option from the **Add/Delete Measures for** list. Upon selection of the **Dial Graph** option, the pre-configured measures for the dial graph will appear in the **Existing Value(s)** list. Similarly, to add a measure to the digital display, pick the **Digital Graph** option from the **Add/Delete Measures for** list. In this case, the **Existing Value(s)** list box will display all those measures for which digital displays pre-exist.
   - Next, select the **Test** that reports the said measure, pick the measure of interest from the **Measures** list, provide a **Display** name for the measure, and click the **Add** button to add the chosen measure to the **Existing Value(s)** list. **Note that while configuring measures for a dial graph the 'Measures' list will display only those measures that report percentage values**.

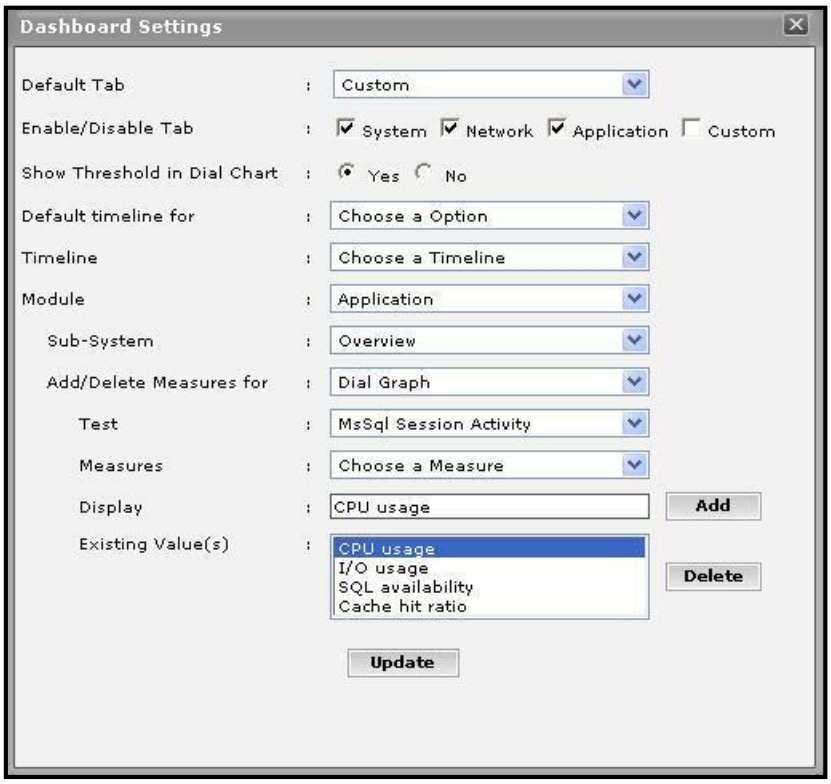


Figure 3.28: Configuring measures for the dial graph

- If you want to delete one/more measures from the dial/digital graphs, then, as soon as you choose the **Dial Graph** or **Digital Graph** option from the **Add/Delete Measures for** list, pick any of the displayed measures from the **Existing Value(s)** list, and click the **Delete** button.
- Finally, click the **Update** button to register the changes.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

- Clicking on a dial/digital graph will lead you to the layer model page of the MS SQL application; this page will display the exact layer-test combination that reports the measure represented by the dial/digital graph.

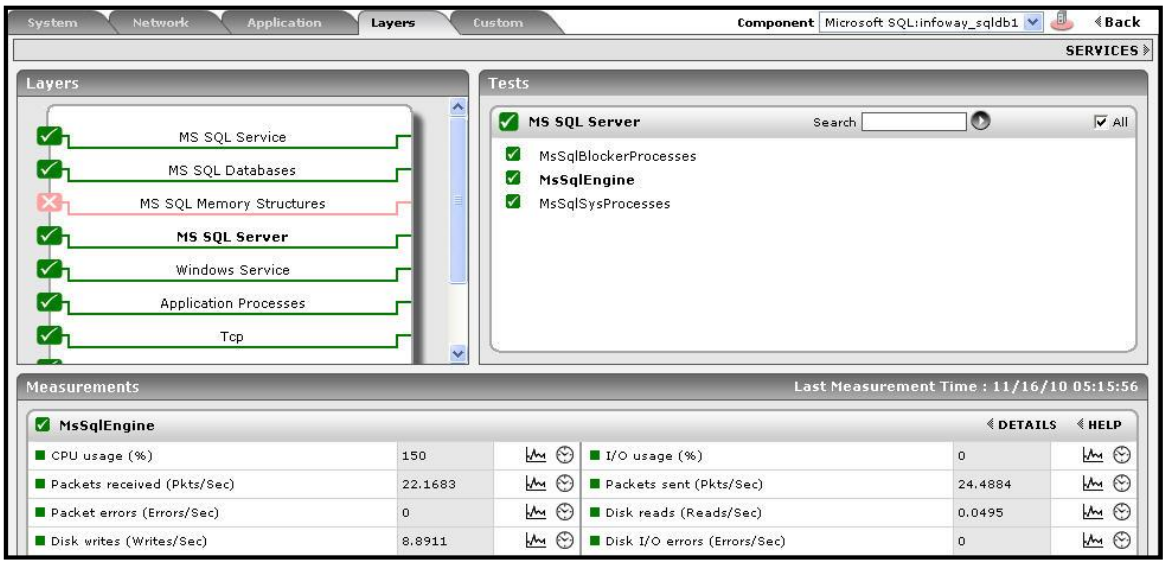


Figure 3.29: The page that appears when the dial/digital graph in the Overview dashboard of the MS SQL Application is clicked

8. If your eG license enables the **Configuration Management** capability, then, an **Application Configuration** section will appear here (as shown in Figure 3.24) providing the basic configuration of the application. You can configure the type of configuration data that is to be displayed in this section by following the steps below:

   - Click on the icon at the top of the **Application Dashboard**. In the **Dashboard Settings** window that appears, select **Application** from the **Module** list, and **Overview** from the **Sub-System** list.
   - To add more configuration information to this section, first, pick the **Application Configuration** option from the **Add/Delete Measures for** list. Upon selection of this option, all the configuration measures that pre-exist in the **Configuration Management** section will appear in the **Existing Value(s)** list.
   - Next, select the config **Test** that reports the said measure, pick the measure of interest from the **Measures** list, provide a **Display** name for the measure, and click the **Add** button to add the chosen measure to the **Existing Value(s)** list.
   - If you want to delete one/more measures from this section, then, as soon as you choose the **Application Configuration** option from the **Add/Delete Measures for** list, pick any of the displayed measures from the **Existing Value(s)** list, and click the **Delete** button.
   - Finally, click the **Update** button to register the changes.

9. Next to this section, you will find a pre-configured list of **Key Performance Indicators** of the MS SQL application. Besides indicating the current state of and current values reported by a default set of resource usage metrics, this section also reveals 'miniature' graphs of each measure, so that you can instantly study how that measure has behaved during the last 1 hour (by default) and thus determine whether the change in state of the measure was triggered by a sudden dip in performance or a consistent one. Clicking on a measure here will lead you to Figure 3.30, which displays the layer and test that reports the measure.

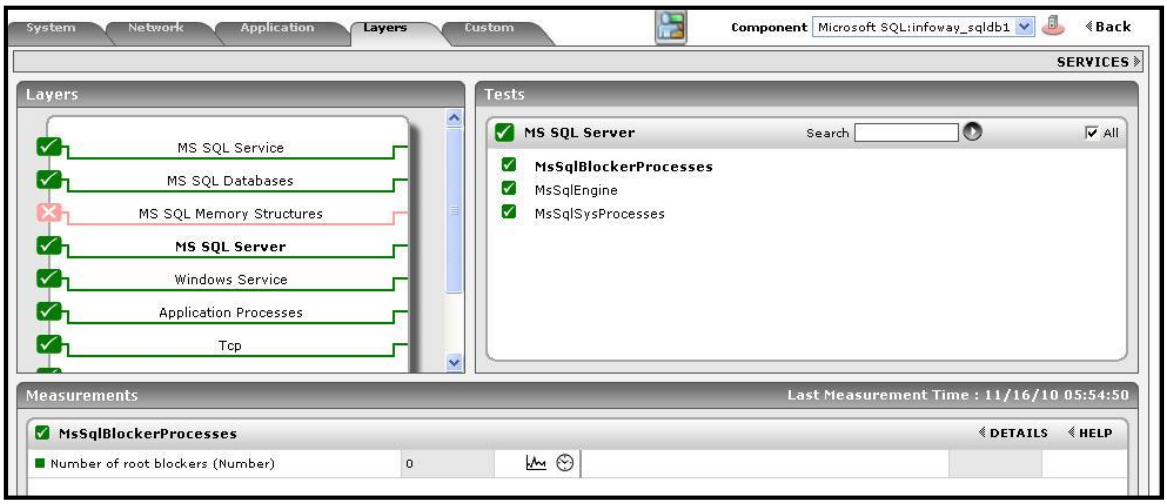


Figure 3.30: Clicking on a Key Performance Indicator

You can, if required, override the default measure list in the **Key Performance Indicators** section by adding more critical measures to the list or by removing one/more existing ones from the list. For this, do the following:

- Click on the icon at the top of the **Application Dashboard**. In the **Dashboard Settings** window that appears, select **Application** from the **Module** list, and **Overview** from the **Sub-System** list.
- To add more metrics to the **Key Performance Indicators** section, first, pick the **Performance Indicator** option from the **Add/Delete Measures for** list. Upon selection of this option, all the measures that pre-exist in the **Key Performance Indicators** section will appear in the **Existing Value(s)** list.
- Next, select the **Test** that reports the said measure, pick the measure of interest from the **Measures** list, provide a **Display** name for the measure, and click the **Add** button to add the chosen measure to the **Existing Value(s)** list.
- If you want to delete one/more measures from this section, then, as soon as you choose the **Key Performance Indicators** option from the **Add/Delete Measures for** list, pick any of the displayed measures from the **Existing Value(s)** list, and click the **Delete** button.
- Finally, click the **Update** button to register the changes.

10. Clicking on a 'miniature' graph that corresponds to a key performance indicator will enlarge the graph (see Figure 3.31), so that you can view and analyze the measure behavior more clearly, and can also alter the **Timeline** and dimension (**3D**/ **2D**) of the graph, if need be.

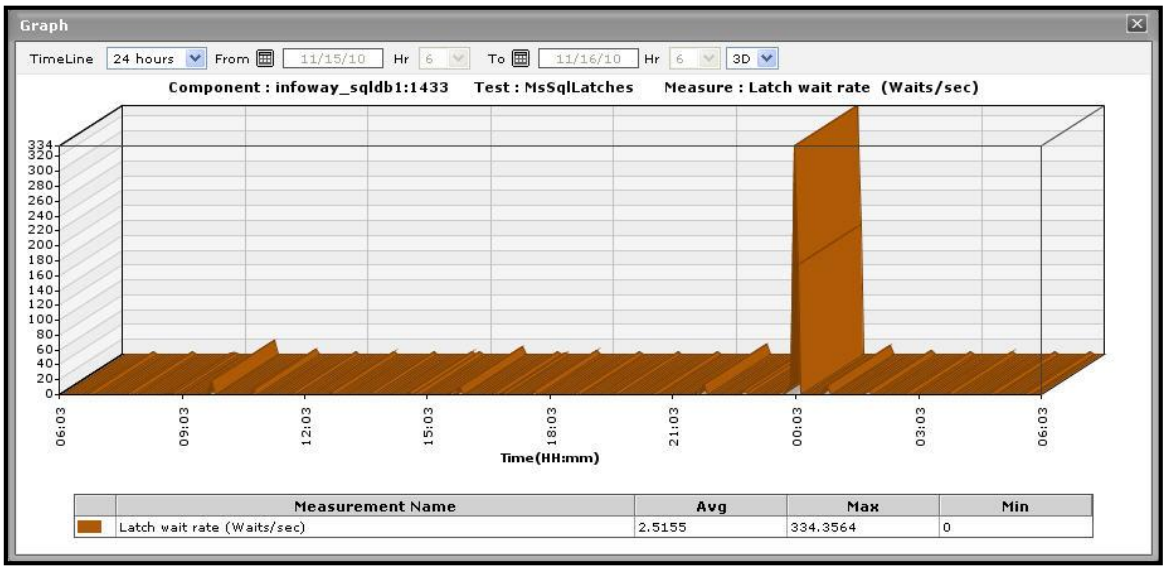


Figure 3.31: Enlarging the Key Performance Indicator graph

11. This way, the first few sections of the **At-A-Glance** tab page help understand what issues are currently affecting the application health, and when they actually originated. To diagnose the root-cause of these issues however, you would have to take help from the remaining sections of the **At-A-Glance** tab page. For instance, the **Key Performance Indicators** section may indicate a sudden/steady increase in the Log cache hit ratio of the MS SQL application. However, to determine whether the rise in the Log cache hit ratio was a result of one/more high SQL processes executing on the MS SQL application or a couple of resource-intensive SQL applications, you need to focus on the **SQL Process - Summary** section. This **SQL Process - Summary** section for starters reveals the number of Processes that are in varying states of activity. With the help of this section therefore, you can quickly figure out whether there are currently any:

    - Processes that are being processed in the background;
    - Processes that are being blocked by other processes;
    - Processes that are not utilized at present by the MS SQL application;
    - Processes that are currently running for the target MS SQL application, etc.

    Say, you notice that too many processes are currently running in a BACKGROUND state. Immediately, you might want to know whether this is a sudden occurrence, or has that problem occurred over a course of time. To enable you to determine this, every process that is displayed in the **SQL Process - Summary** section is accompanied by a 'miniature' graph, which tracks the changes in the corresponding process during the last 1 hour (by default). To enlarge the graph, click on it; this will invoke Figure 3.32. The enlarged graph allows you to change the **Timeline** for analysis, and also the graph dimension.

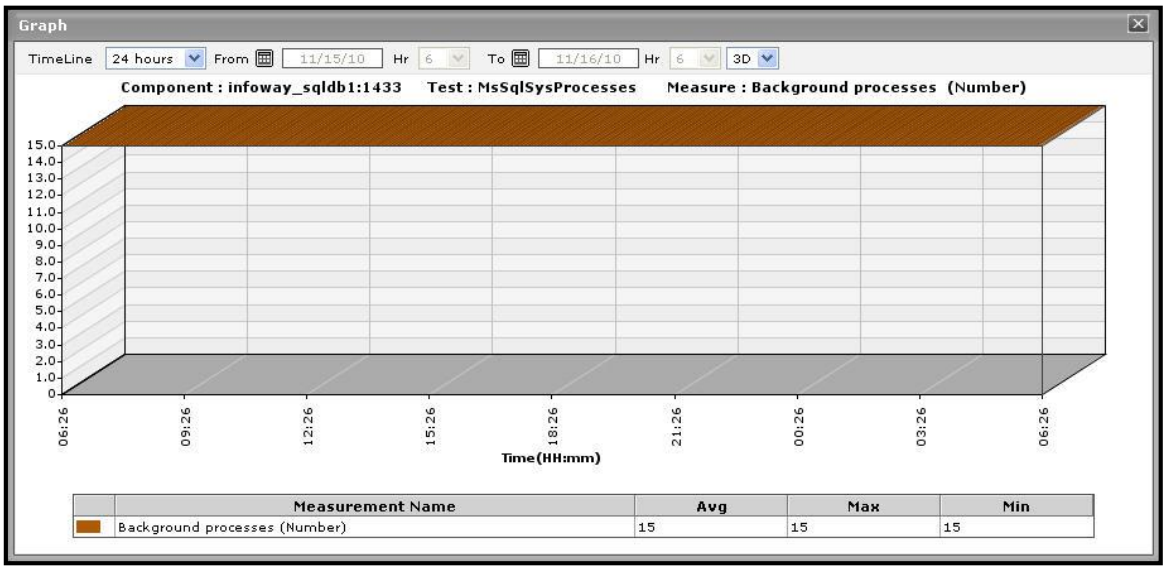


Figure 3.32: The enlarged processes graph

12. The **Database Usage Summary** section reveals how well the databases are being managed by the target MS SQL application. For every database, this section reveals the space that is utilized by the data files, the space that is used for indexing the data, and the space that is currently unused in the database. From this information, you can infer which database is utilizing the maximum amount of allocated resources. By default, the Database list provided by this section is sorted in the alphabetical order of the names of the databases. If need be, you can change the sort order so that the databases are arranged in, say, the descending order of values displayed in the **Data Space** column - this column displays the space that is utilized by the data files. To achieve this, simply click on the column heading **Data Space**. Doing so tags the **Data Space** label with a **down arrow** icon - this icon indicates that the **Database Usage Summary** table is currently sorted in the descending order of the space used by the data files. To change the sort order to 'ascending', all you need to do is just click again on the **Data Space** label or the **down arrow** icon. Similarly, you can sort the table based on any column available in it.

13. The **Database Transaction Summary** section, on the other hand, provides the transaction details of each of the databases that are currently available in the MS SQL application. By default, the database list provided by this section is sorted in the alphabetical order of the process names. If need be, you can change the sort order so that the databases are arranged in, say, the descending order of values displayed in the **Active transactions** column - this column displays the number of active transactions made by each database that is available in the MS SQL application. To achieve this, simply click on the column heading – **Active transactions**. Doing so tags the **Active transactions** label with a **down arrow** icon - this icon indicates that the database list is currently sorted in the descending order of the active transaction count. To change the sort order to 'ascending', all you need to do is just click again on the **Active transactions** label or the **down arrow** icon. Similarly, you can sort the process list based on any column available in the **Database Transaction Summary** section.

14. While the **At-A-Glance** tab page reveals the current state of the databases and the overall resource usage of the MS SQL application, to perform additional diagnosis on problem conditions highlighted by the **At-A-Glance** tab page and to accurately pinpoint their root-cause, you need to switch to the **Details** tab page (see Figure 3.33) by clicking on it. For instance, the **At-A-Glance** tab page may indicate the number of processes that are currently blocked, but to know which process has been blocked for the longest time, you will have to use the **Details** tab page.

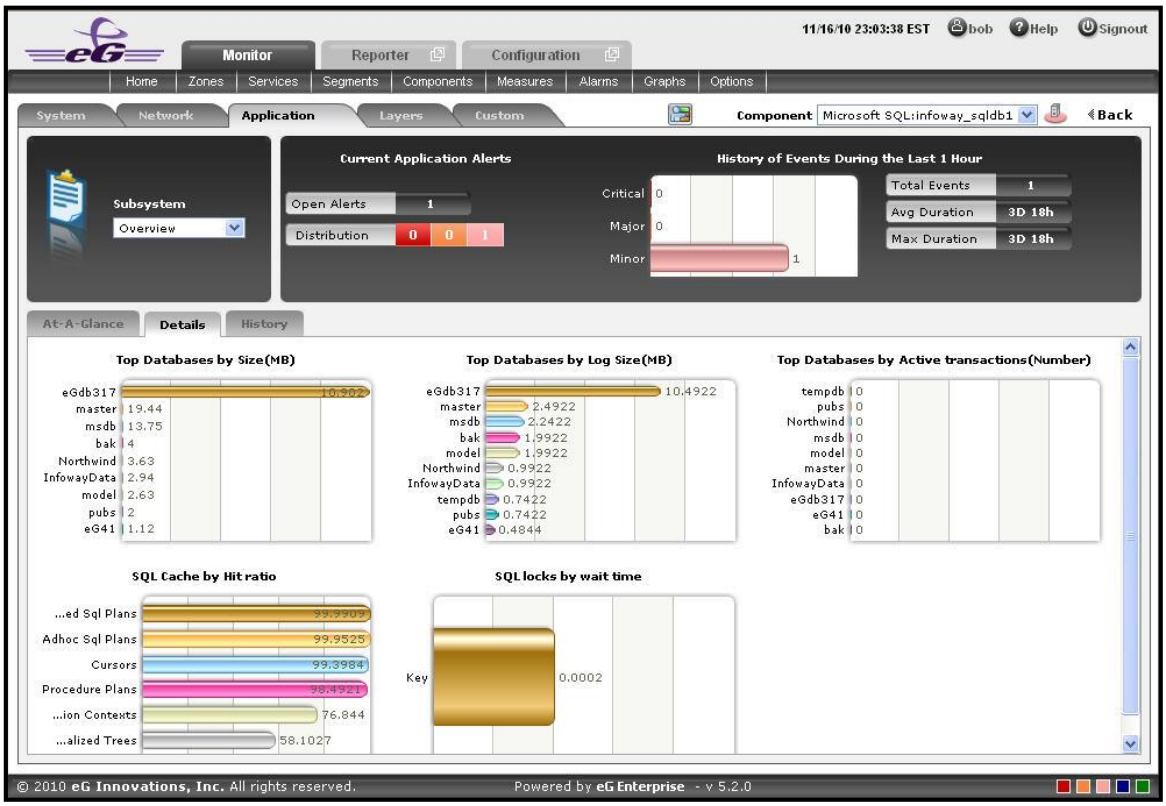


Figure 3.33: The Details tab page of the MS SQL Application Overview Dashboard

15. The **Details** tab page comprises of a default set of comparison bar graphs using which you can accurately determine the following:

- The size of the databases.
- The Log size of the databases.
- How many Active transactions are made on each database?
- What is the SQL Cache hit ratio of each database?

If required, you can configure the **Details** tab page to include comparison graphs for more measures, or can even remove one/more existing graphs by removing the corresponding measures. To achieve this, do the following:

- Click on the icon at the top of the **Application Dashboard**. In the **Dashboard Settings** window that appears, select **Application** from the **Module** list, and **Overview** from the **Sub-System** list.
- To add measures for comparison graphs, first, pick the **Comparison Graph** option from the **Add/Delete Measures for** list. Upon selection of this option, the pre-configured measures for comparison graphs will appear in the **Existing Value(s)** list.
- Next, select the **Test** that reports the said measure, pick the measure of interest from the **Measures** list, provide a **Display** name for the measure, and click the **Add** button to add the chosen measure to the **Existing Value(s)** list.

Figure 3.34: Configuring measures for the dial graph

- If you want to delete one/more measures for which comparison graphs pre-exist in the **details** tab page, then, as soon as you choose the **Comparison Graph** option from the **Add/Delete Measures for** list, pick any of the displayed measures from the **Existing Value(s)** list, and click the **Delete** button.
- Finally, click the **Update** button to register the changes.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

16. By default, the comparison bar graphs list the top-10 databases only. To view the complete list of databases, simply click on the corresponding graph in Figure 3.33. This enlarges the graph as depicted by Figure 3.35.

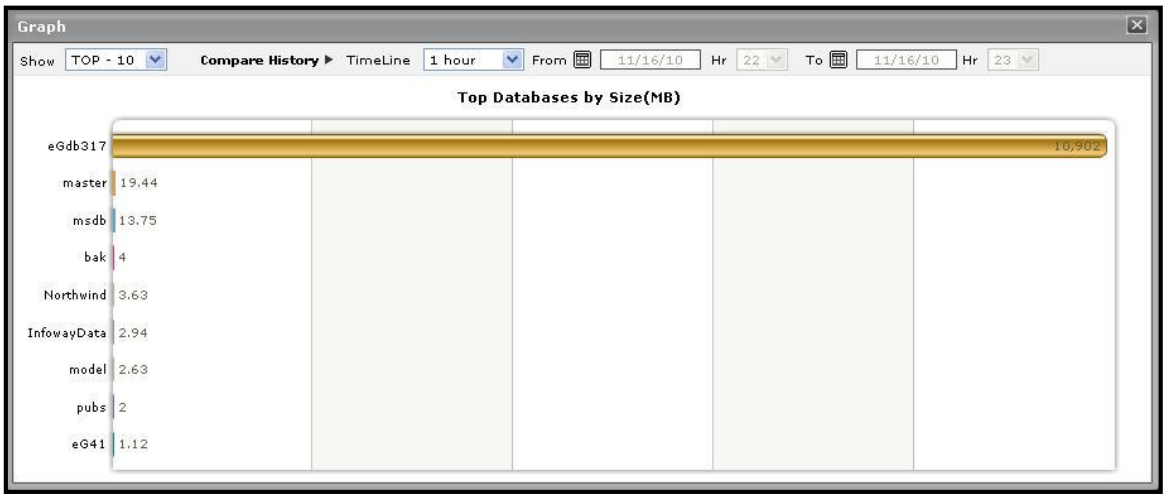
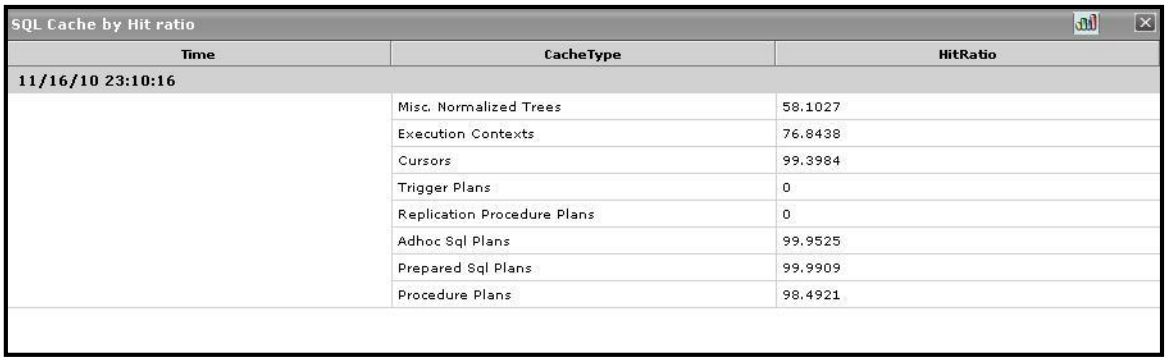


Figure 3.35: The expanded top-n graph in the Details tab page of the MS SQL Application Overview Dashboard

17. Though the enlarged graph lists all the databases in this case by default, you can customize the enlarged graph to display the details of only a few of the larger/smaller databases by picking a **TOP-N** or **LAST-N** option from the **Show** list in Figure 3.35.

18. Another default aspect of the enlarged graph is that it pertains to the current period only. Sometimes however, you might want to know what occurred during a point of time in the past; for instance, while trying to understand the reason behind a sudden increase or decrease in the size of the database usage on a particular day last week, you might want to first determine the database whose size has abnormally increased / decreased on the same day. To figure this out, the enlarged graph allows you to compare the historical performance of the databases. For this purpose, click on the **Compare History** link in Figure 3.35 and select the **TimeLine** of your choice.

19. Where detailed diagnosis is applicable, you can quickly view the detailed measures that correspond to a comparison graph by clicking on the icon at the right, top corner of the enlarged graph. This will invoke Figure 3.36, using which you can arrive at the root-cause of a problem.



Time	CacheType	HitRatio
11/16/10 23:10:16		
	Misc. Normalized Trees	58.1027
	Execution Contexts	76.8438
	Cursors	99.3984
	Trigger Plans	0
	Replication Procedure Plans	0
	Adhoc Sql Plans	99.9525
	Prepared Sql Plans	99.9909
	Procedure Plans	98.4921

Figure 3.36: The detailed diagnosis that appears when the DD icon in the enlarged comparison bar graph is clicked

20. For detailed time-of-day / trend analysis of the historical performance of a MS SQL application, use the **History** tab page. By default, this tab page (see Figure 3.37) provides time-of-day graphs of critical measures extracted from the target MS SQL application, using which you can understand how performance has varied during the default period of 24 hours. In the event of a problem, these graphs will help you determine whether the problem occurred suddenly or grew with time. To alter the timeline of all the graphs simultaneously, click on the **Timeline** link at the right, top corner of the **History** tab page of Figure 3.37.

You can even override the default timeline (of 24 hours) of the measure graphs, by following the steps below:

- Click on the icon at the top of the **Application Dashboard**.
- In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.
- Then, choose a **Timeline** for the graph.
- Finally, click the **Update** button.

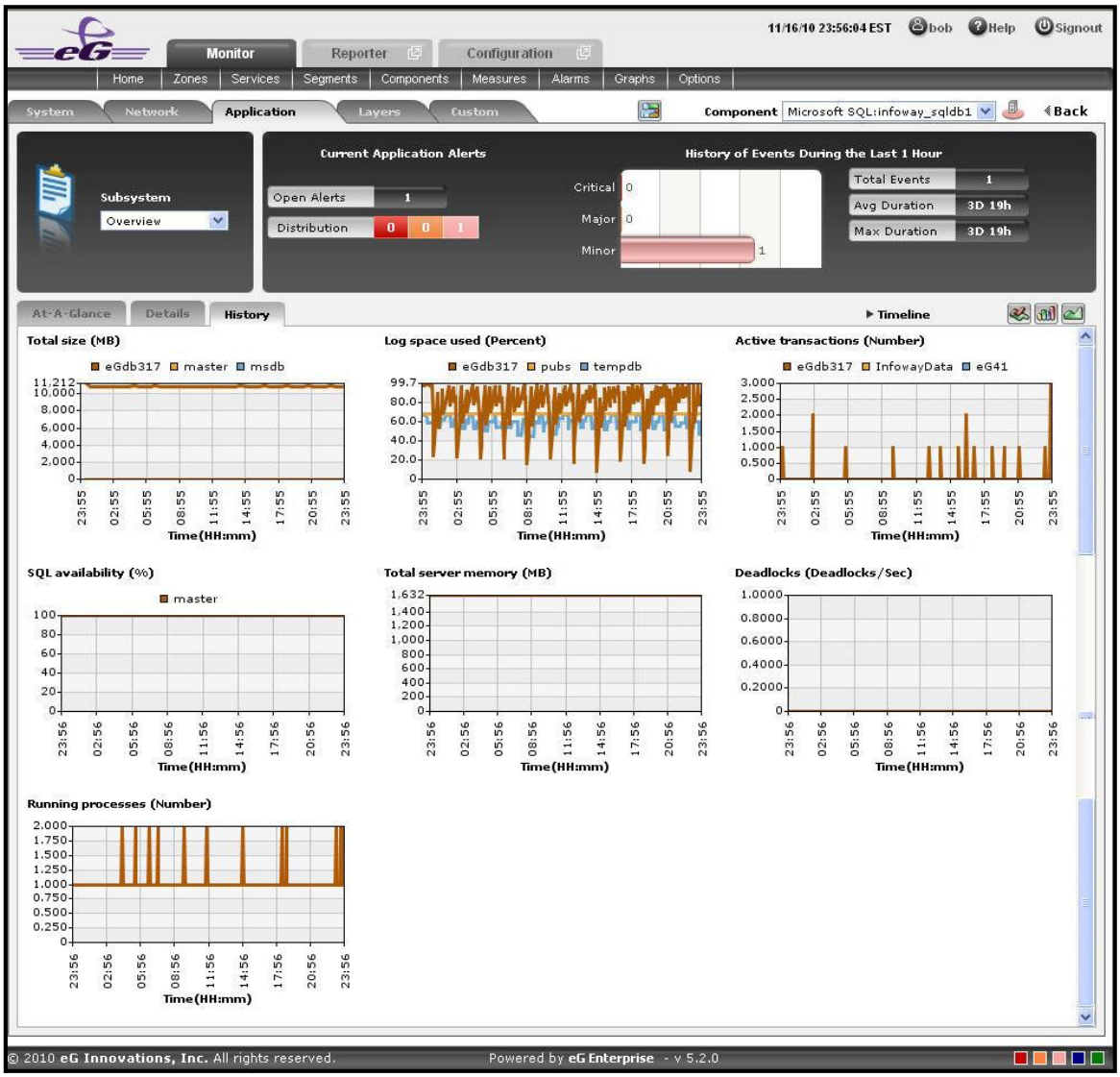


Figure 3.37: Time-of-day measure graphs displayed in the History tab page of the Application Overview Dashboard

21. You can click on any of the graphs to enlarge it, and can change the **Timeline** of that graph in the enlarged mode as shown in Figure 3.38.

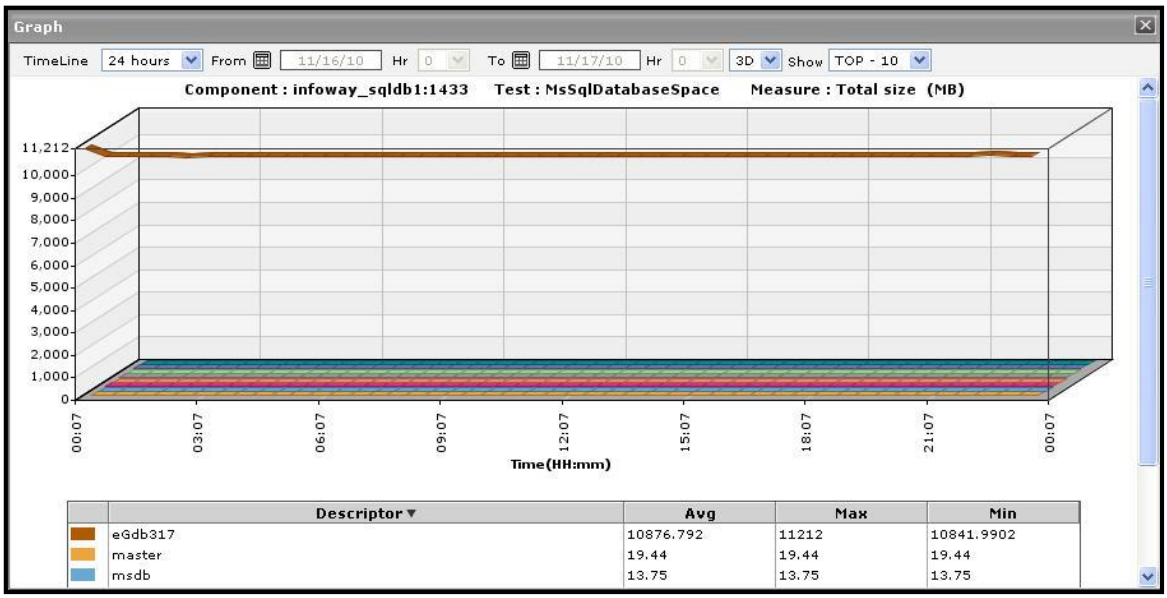


Figure 3.38: An enlarged measure graph of a MS SQL Application

22. In case of tests that support descriptors, the enlarged graph will, by default, plot the values for the **TOP-10** descriptors alone. To configure the graph to plot the values of more or less number of descriptors, select a different **TOP-N** / **LAST-N** option from the **Show** list in Figure 3.38.

23. If you want to quickly perform service level audits on the MS SQL application, then summary graphs may be more appropriate than the default measure graphs. For instance, a summary graph might come in handy if you want to determine the variation of Total size of a database with respect to the percentage of time during the last 24 hours. Using such a graph, you can determine whether the database size has been constant or varied, and if not, how frequently the application faltered in this regard. To invoke such summary graphs, click on the icon at the right, top corner of the **History** tab page. Figure 3.39 will then appear.

Figure 3.39: Summary graphs displayed in the History tab page of the Application Overview Dashboard

24. You can alter the timeline of all the summary graphs at one shot by clicking the **Timeline** link at the right, top corner of the **History** tab page of Figure 3.37. You can even alter the default timeline (of 24 hours) for these graphs, by following the steps given below:

    - Click on the icon at the top of the **Application Dashboard**.

- In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
- Then, choose a **Timeline** for the graph.
- Finally, click the **Update** button.

25. To change the timeline of a particular graph, click on it; this will enlarge the graph as depicted by Figure 3.40. In the enlarged mode, you can alter the **Timeline** of the graph. Also, though the graph plots hourly summary values by default, you can pick a different **Duration** for the graph in the enlarged mode, so that daily/monthly performance summaries can be analyzed.

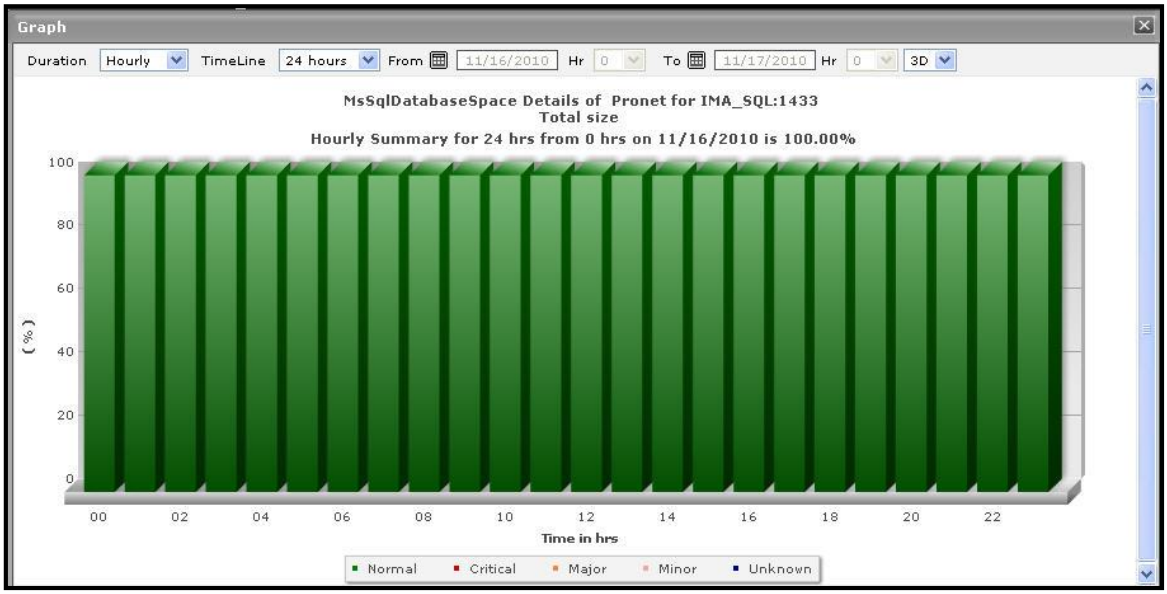


Figure 3.40: An enlarged summary graph of the MS SQL Application

26. To perform effective analysis of the past trends in performance, and to accurately predict future measure behavior, click on the icon at the right, top corner of the **History** tab page. These trend graphs as shown in Figure 3.41 typically show how well and how badly a measure has performed every hour during the last 24 hours (by default). For instance, the Total size trend graph of each database of a MS SQL application will help you figure out the total size of the database that was available in the application every hour during the last 24 hours. If the gap between the minimum and maximum values is marginal, you can conclude that the size of the database has been more or less constant during the designated period; this implies that the size of the database has neither increased nor decreased steeply during the said timeline. On the other hand, a wide gap between the maximum and minimum values is indicative of an erratic change in the size of the database, and may necessitate further investigation.

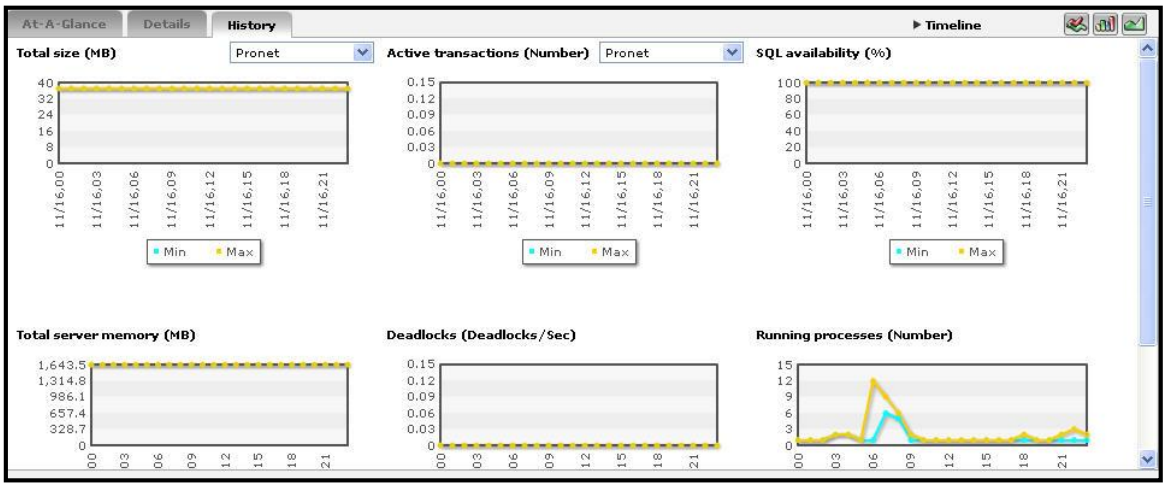


Figure 3.41: Trend graphs displayed in the History tab page of the Application Overview Dashboard

27. To analyze trends over a broader time scale, click on the **Timeline** link at the right, top corner of the **History** tab page, and edit the **Timeline** of the trend graphs. Clicking on any of the miniature graphs in this tab page will enlarge that graph, so that you can view the plotted data more clearly and even change its **Timeline**.

    To override the default timeline (of 24 hours) of the trend graphs, do the following:

    - Click on the icon at the top of the **Application Dashboard**.
    - In the **Dashboard Settings** window that appears, select **Trend Graph** from the **Default Timeline for** list.
    - Then, choose a **Timeline** for the graph.
    - Finally, click the **Update** button.

28. Besides the timeline, you can even change the **Duration** of the trend graph in the enlarged mode. By default, **Hourly** trends are plotted in the trend graph. By picking a different option from the **Duration** list, you can ensure that **Daily** or **Monthly** trends are plotted in the graph instead.

29. Also, by default, the trend graph only plots the minimum and maximum values registered by a measure. Accordingly, the **Graph** type is set to **Min/Max** in the enlarged mode. If need be, you can change the **Graph** type to **Avg** (see Figure 3.42), so that the average trend values of a measure are plotted for the given **Timeline**. For instance, if an average trend graph is plotted for the *Total size* measure, then the resulting graph will enable administrators to ascertain whether the size of a particular database has been constant during a specified timeline.

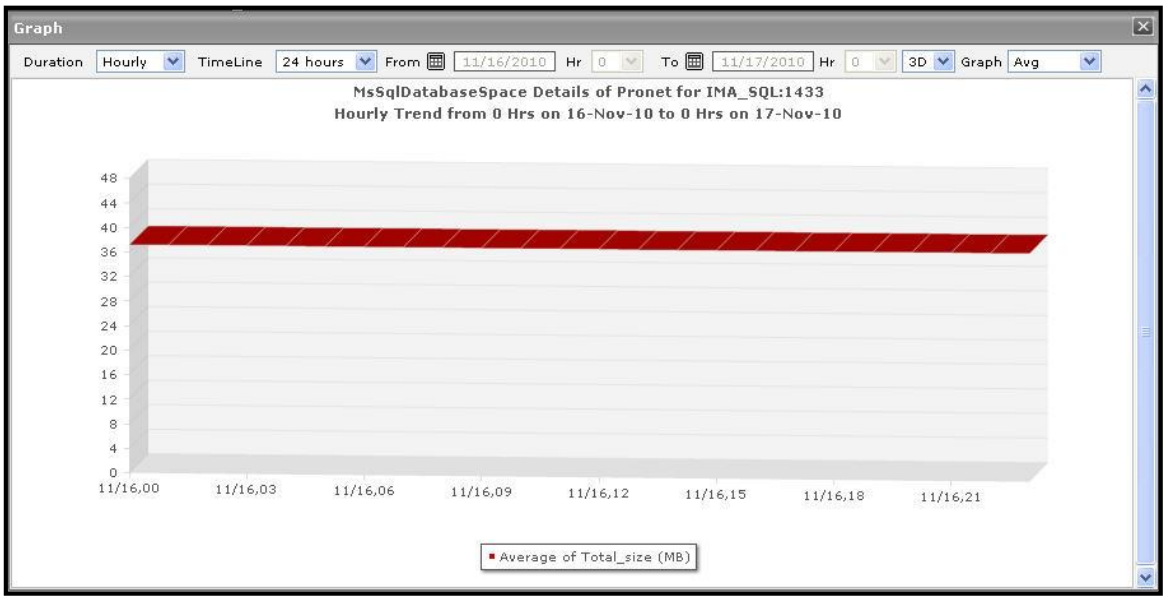


Figure 3.42: Viewing a trend graph that plots average values of a measure for a database available in the MS SQL application

30. Likewise, you can also choose **Sum** as the **Graph** type to view a trend graph that plots the sum of the values of a chosen measure for a specified timeline. For instance, if you plot a 'sum of trends' graph for the measure that reports the Total size of a database available in the MS SQL application, then, the resulting graph will enable you to analyze, on an hourly/daily/monthly basis (depending upon the **Duration** chosen), whether there was any change in the size of the database.

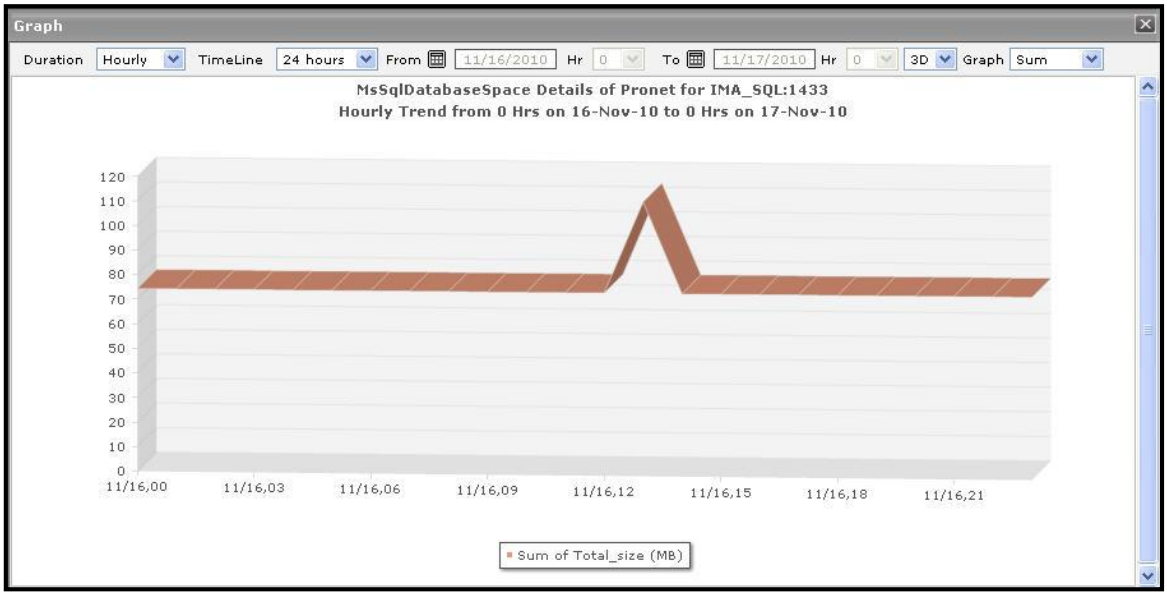


Figure 3.43: A trend graph plotting sum of trends for a database available in the MS SQL application

Note:

In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.

31. At any point in time, you can switch to the measure graphs by clicking on the button.
32. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:
    - Click the button at the top of the dashboard.
    - The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.44, pick **Application**, choose **Overview** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.

Figure 3.44: Adding a new graph to the **History** tab page

- The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the

measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.

- To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
- Next, select the **Measure** of interest.
- Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
- This will add a new measure, summary, and trend graph for the chosen measure, to the **History** tab page.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

3.7.2 SQLServer

If you want to assess how efficiently the SQL server manages the databases available on it, and thus promptly detect the server related discrepancies, select the **SQLServer** option from the **Subsystem** list.

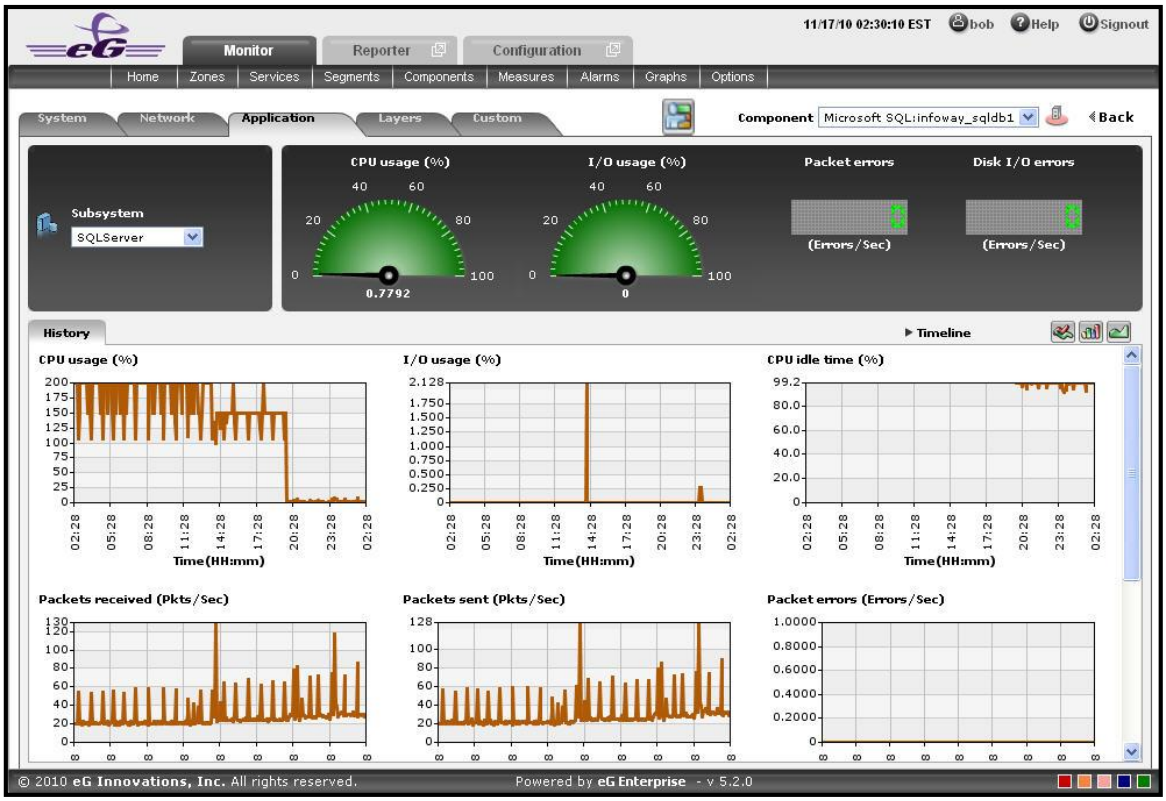


Figure 3.45: The SQLServer Dashboard

The contents of the **SQLServer** dashboard that then appears (see Figure 3.45) are as follows:

1. The dashboard begins with a dial and digital graphs section, which enables you to visually track the changes that are happening in the measures related to the SQL server of the MS SQL application. For instance, the CPU usage of the SQL server can be viewed at a single glance. Clicking on a dial/digital graph will lead you to the layer model page of the MS SQL Application; this page will display the exact layer-test combination that reports the measure represented by the dial/digital graph.
2. The **History** tab page displays measure graphs that depict how the server related measures such as CPU usage has been varying over time. In the event of any retaliation in the measures, this time-bound analysis will help you to easily differentiate between a sudden spike in the CPU usage and a consistent rise in the same.

   By default, these historical graphs track the time-of-day variations in memory usage during the last 24 hours. You can override this default timeline by following the steps discussed below:

   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.
3. To change the timeline of all the measure graphs at one shot, just click on the **Timeline** link at the right, top corner of the **History** tab page. To alter the timeline for a single graph, just click on that graph - this will enlarge the graph. You can change the **Timeline** of the graph in the enlarged mode.

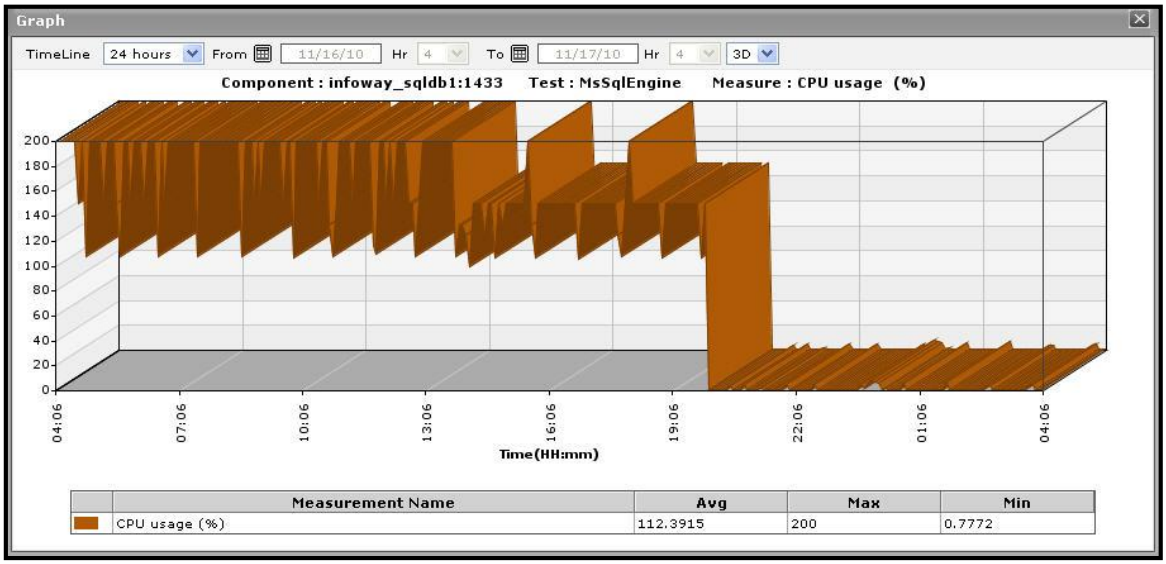


Figure 3.46: An enlarged measure graph in the History tab page of the SQLServer dashboard

4. Instead of these measure graphs, you can, if required, view summary graphs of the memory-related measures in the **History** tab page. For this, click on the icon at the right, top corner of the **History** tab page. Summary graphs help you figure out the percentage of time during the last 24 hours (by default) the MS SQL application was hogged by the server-related issues. While monitoring mission-critical applications that are governed by rigid service level agreements, summary graphs will help you determine whether the guaranteed CPU usage levels were fulfilled or not, and if not, how often did the usage levels slip.

   You can override the default timeline (of 24 hours) of the summary graphs by following the steps discussed below:

- Click on the icon at the top of the **Application Dashboard**.
- In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
- Then, choose a **Timeline** for the graph.
- Finally, click the **Update** button.

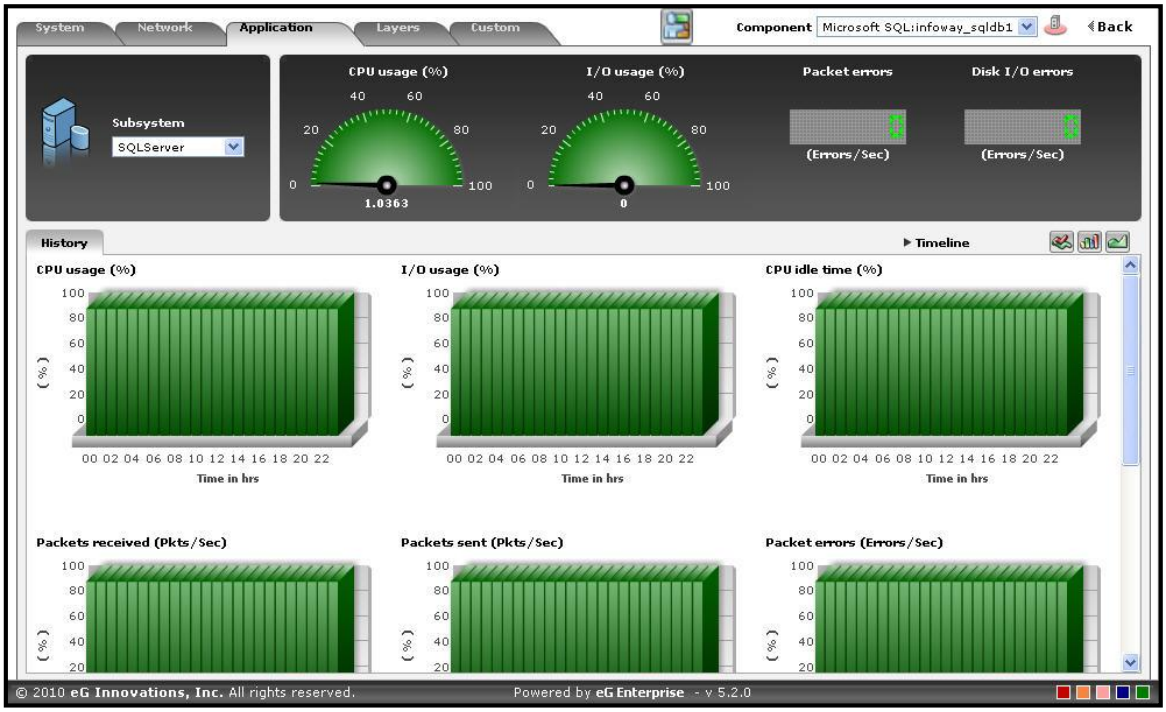


Figure 3.47: Summary graphs displayed in the History tab page of the SQLServer Dashboard

5. Here again, you can change the **Timeline** of all the summary graphs by clicking on the **Timeline** link in Figure 3.47, or click on a graph, enlarge it, and change its **Timeline** in the enlarged mode. Also, though the graph plots hourly summary values by default, you can pick a different **Duration** for the graph in the enlarged mode, so that daily/monthly performance summaries can be analyzed.
6. You can click on the icon at the right, top corner of the **History** tab page to view trend graphs of the memory usage-related measures. By default, these trend graphs plot the maximum and minimum memory usage values for every hour of the last 24 hours (by default). The default timeline of 24 hours can be overridden by following the steps discussed below:
   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Trend Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

Using these trend graphs, you can understand the variations in the CPU usage of the SQL server during the last

24 hours (by default), deduce the future usage trends, and accordingly recommend changes to the server size.

Figure 3.48: Trend graphs displayed in the History tab page of the SQLServer Dashboard

7. Here again, you can change the **Timeline** of all the trend graphs by clicking on the **Timeline** link in Figure 3.48, or click on a graph, enlarge it, and change its **Timeline** in the enlarged mode. Also, though the graph plots hourly trend values by default, you can pick a different **Duration** for the graph in the enlarged mode, so that daily/monthly performance trends can be analyzed.

8. Also, by default, the trend graph only plots the minimum and maximum values registered by a measure. Accordingly, the **Graph** type is set to **Min/Max** in the enlarged mode. If need be, you can change the **Graph** type to **Avg**, so that the average trend values of a measure are plotted for the given **Timeline**. Such a graph will enable you to assess whether the memory resources were utilized effectively or not, over time.

9. Likewise, you can also choose **Sum** as the **Graph** type to view a trend graph that plots the sum of the values of a chosen measure for a specified timeline. For instance, a 'sum of trends' CPU usage will enable you to analyze, on an hourly/daily/monthly basis (depending upon the **Duration** chosen), how the CPU usage of a server has varied during the specified timeline.

Note:

In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.

10. At any point in time, you can switch to the measure graphs by clicking on the button.

11. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If

you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:

- Click the  button at the top of the dashboard.
- The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.44, pick **Application**, choose **SQL Server** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.
- The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.
- To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
- Next, select the **Measure** of interest.
- Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
- This will add a new measure, summary, and trend graph for the chosen measure to the **History** tab page.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

3.7.3 SQLMemory Test

If you want to assess how efficiently the MS SQL application uses the memory resources available to it, and thus promptly detect issues related to the memory-intensive measures, select the **SQLMemory** option from the **Subsystem** list.

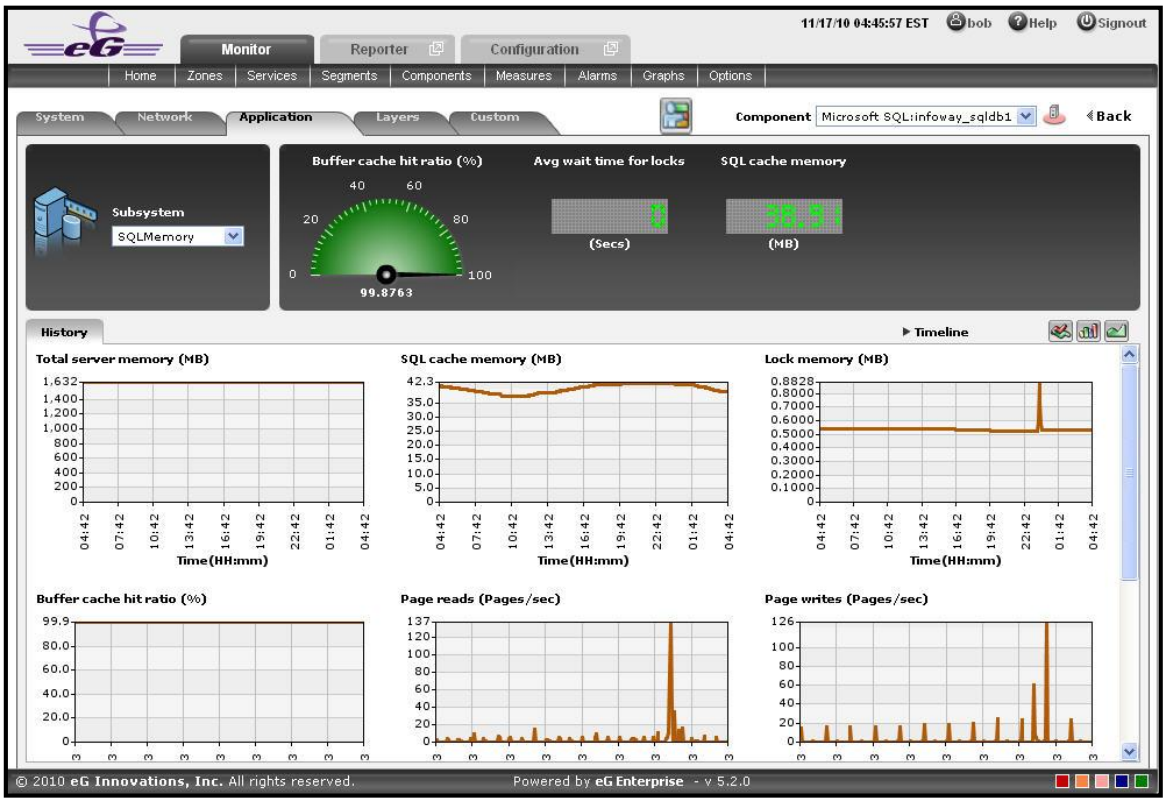


Figure 3.49: The SQLMemory Dashboard

The contents of this dashboard are discussed hereunder:

1. The dashboard begins with a dial and digital graphs section, which enables you to visually track the changes that are happening in the measures related to the memory related measures that are available in the MS SQL application. For instance, the Buffer cache hit ratio pertaining to the SQL Memory can be viewed at a single glance. Clicking on a dial/digital graph will lead you to the layer model page of the MS SQL Application; this page will display the exact layer-test combination that reports the measure represented by the dial/digital graph.
2. The **History** tab page displays time-of-day graphs for all the memory-related measures for a default time duration of 24 hours. You can override this default timeline (of 24 hours) by following the steps below:
   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

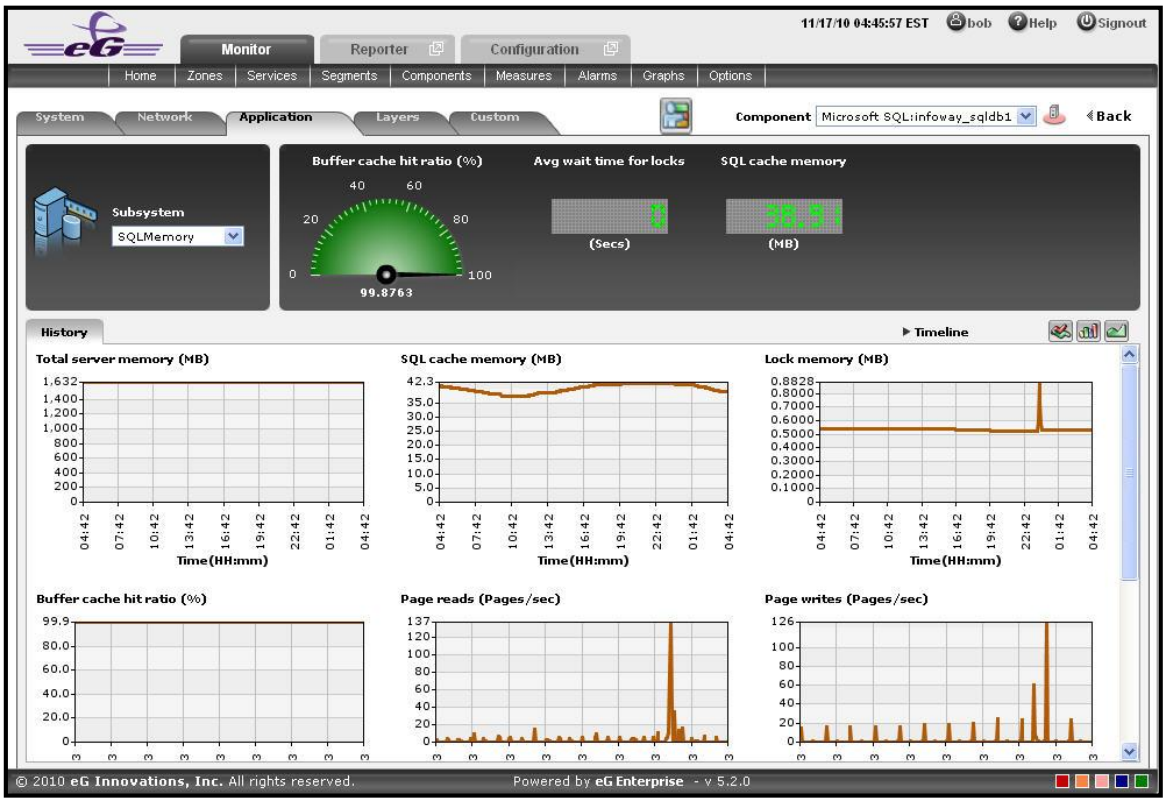


Figure 3.50: The History tab page of the SQLMemory Dashboard

3. Say, you suddenly notice that the Page reads measure has increased; in such a case, you can use these measure graphs to figure out when during the last 24 hours there was an increase in the number of pages that was read per second. If required, you can even look beyond the last 24 hours - i.e., you can find out whether the anomaly originated much earlier. For this, you just need to click on the graph of interest to you. This will enlarge the graph; in the enlarged mode, you can alter the graph **Timeline**, so that the performance of that measure can be analyzed over a broader time window. In this mode, you can even change the graph dimension from **3D** to **2D**, or vice-versa.

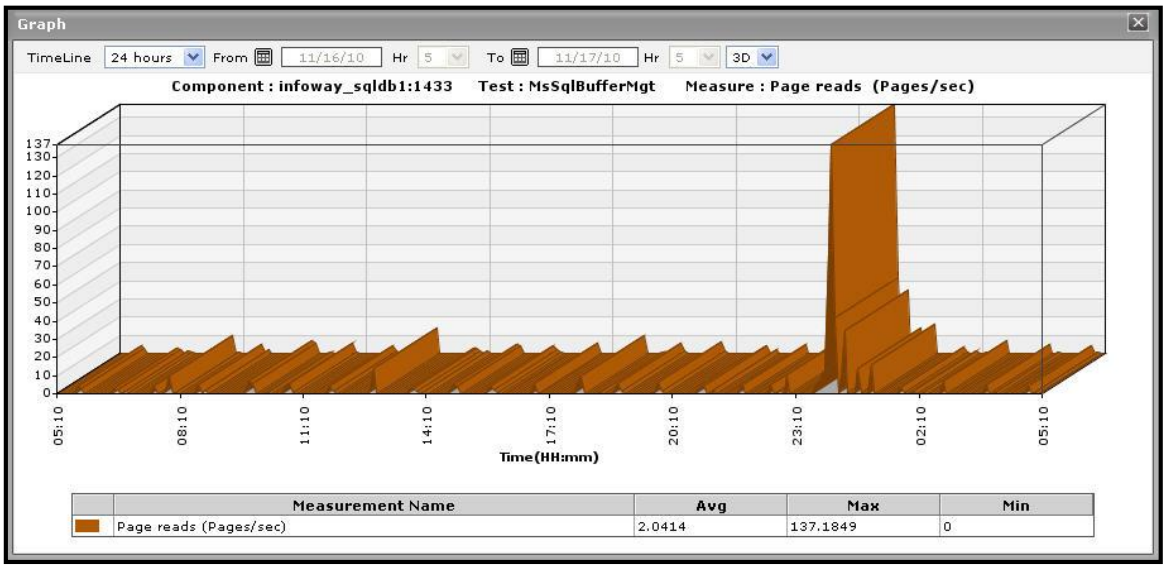


Figure 3.51: An enlarged measure graph in the History tab page of the SQLMemory dashboard

4. To view summary graphs of these memory-related measures instead of the default measure graphs, just click on the icon at the right, top corner of the **History** tab page. Figure 3.52 will then appear. The summary graphs of Figure 3.52 reveal the percentage of time during the last 24 hours (by default) the MS SQL application has been affected by memory-related issues, and the type of issues (whether critical/major/minor) the application was experiencing. These graphs help determine whether the assured service levels were delivered or not.

   The default duration (of 24 hours) of the summary graphs can be overridden by following the procedure discussed below:

   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

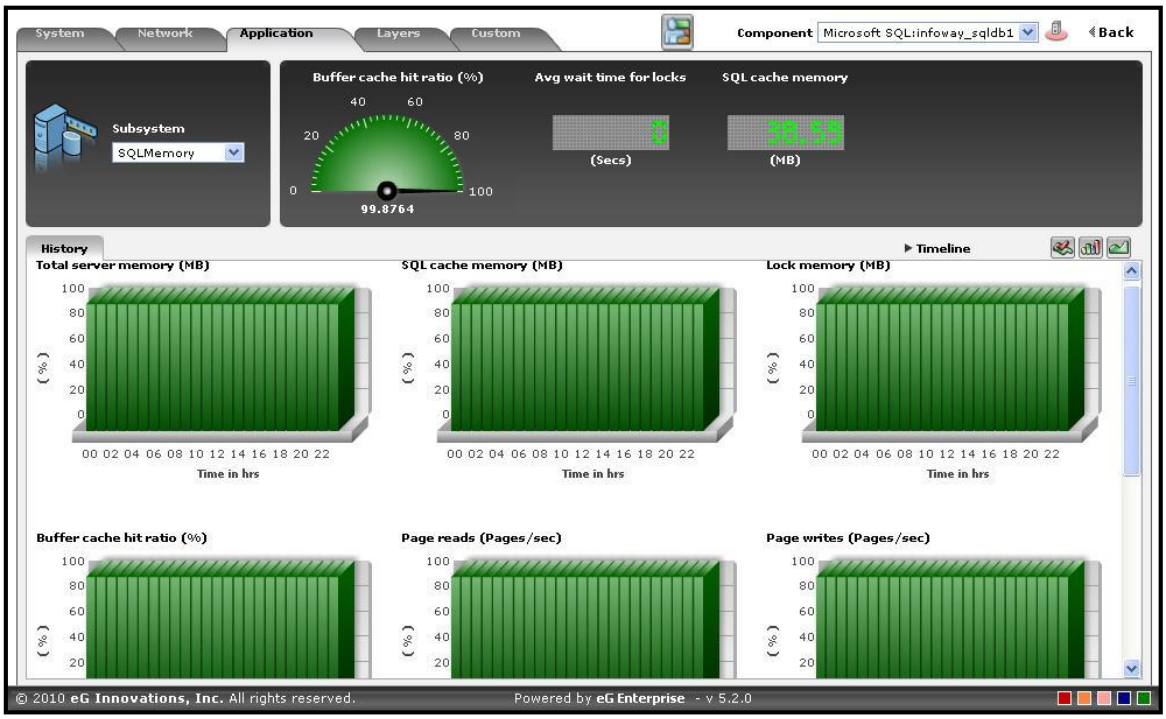


Figure 3.52: Summary graphs displayed in the History tab page of the SQLMemory Dashboard

5. Use the **Timeline** link at the right, top corner of the tab page to change the timeline of all the summary graphs at one shot. For altering the timeline of a single graph, click on it; this will enlarge the graph. In the enlarged mode, you can change the **Timeline** of the summary graph and modify the dimension (3D/2D) of the graph. Also, by default, hourly summaries are plotted in the summary graph; you can configure these graphs to plot daily/monthly summaries instead by picking the relevant option from the **Duration** list in the enlarged mode.

6. If you want to view the past trends in the memory performance, click on the icon at the right, top corner of the **History** tab page. Figure 3.53 will then appear. Using the trend graphs displayed in Figure 3.53, you can better assess the current capacity of your application and can accordingly plan its future capacity. By default, these trend graphs plot the maximum and minimum values registered by every memory-related measure during every hour of the last 24 hours. From this data, you can clearly figure out when during the last 24 hours the application performance has peaked and when it has been below-normal.

   The default duration (of 24 hours) of the trend graphs can be overridden by following the procedure discussed below:

   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Trend Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

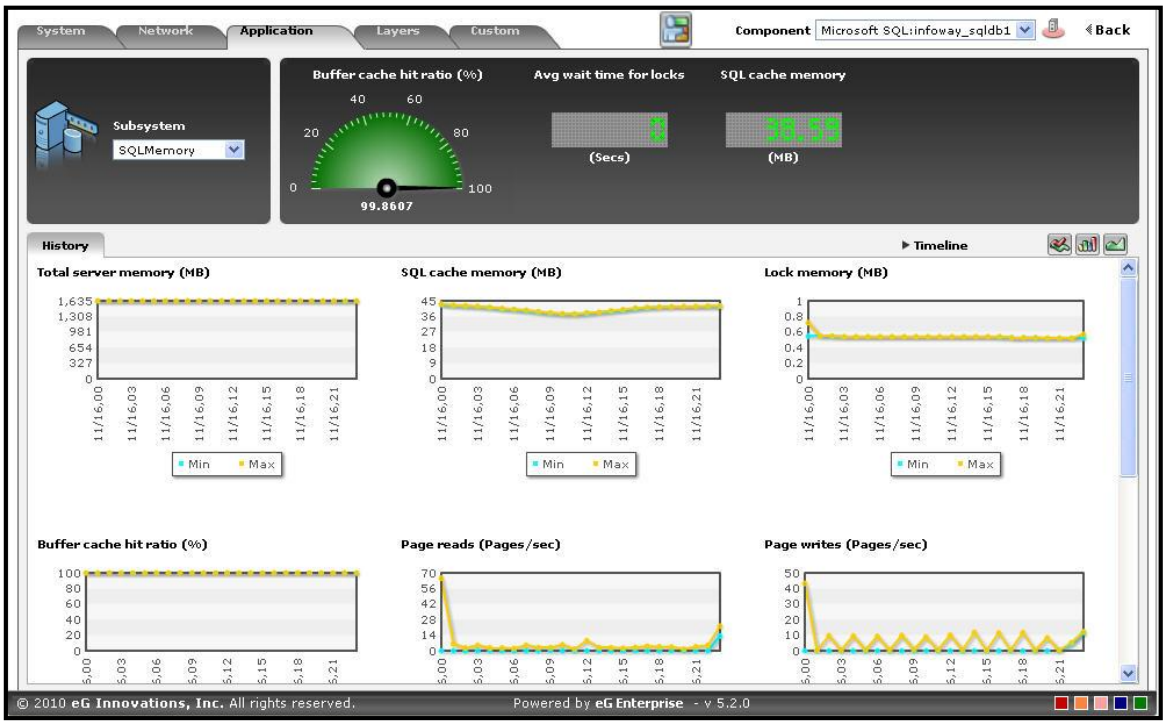


Figure 3.53: Trend graphs displayed in the History tab page of the SQLMemory Dashboard

7. Use the **Timeline** link at the right, top corner of the tab page to change the timeline of all the trend graphs at one shot. For altering the timeline of a single graph, click on it; this will enlarge the graph. In the enlarged mode, you can change the **Timeline** of the trend graph and modify the dimension (3D/2D) of the graph. Also, by default, hourly trends are plotted in the trend graph; you can configure these graphs to plot daily/monthly trend values instead by picking the relevant option from the **Duration** list in the enlarged mode. Moreover, by default, the trend graphs plot only the minimum and maximum values registered by a measure during the specified timeline - this graph will enable you to isolate those times at which performance of that measure had peaked and the times it had fared poorly. If need be, you can select the **Avg** option from the **Graph type** list in the enlarged mode to make sure that the trend graph plots the average trend values for the specified timeline. Alternatively, you can select the **Sum** option from the **Graph type** list to have the trend graph plot the sum of trends for the specified timeline.

Note:

In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.

8. At any point in time, you can switch to the measure graphs by clicking on the button.

9. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:

   - Click the button at the top of the dashboard.
   - The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.44, pick **Application**, choose **SQLMemory** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for**

list.

- The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.
- To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
- Next, select the **Measure** of interest.
- Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
- This will add a new measure, summary, and trend graph for the chosen measure to the **History** tab page.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

3.7.4 SQLProcesses

Select the **SQLProcesses** option from the **Subsystem** list to know how efficiently the class loader used by the Java application is and has been loading/unloading classes onto memory. Upon selection, Figure 3.54 will appear.

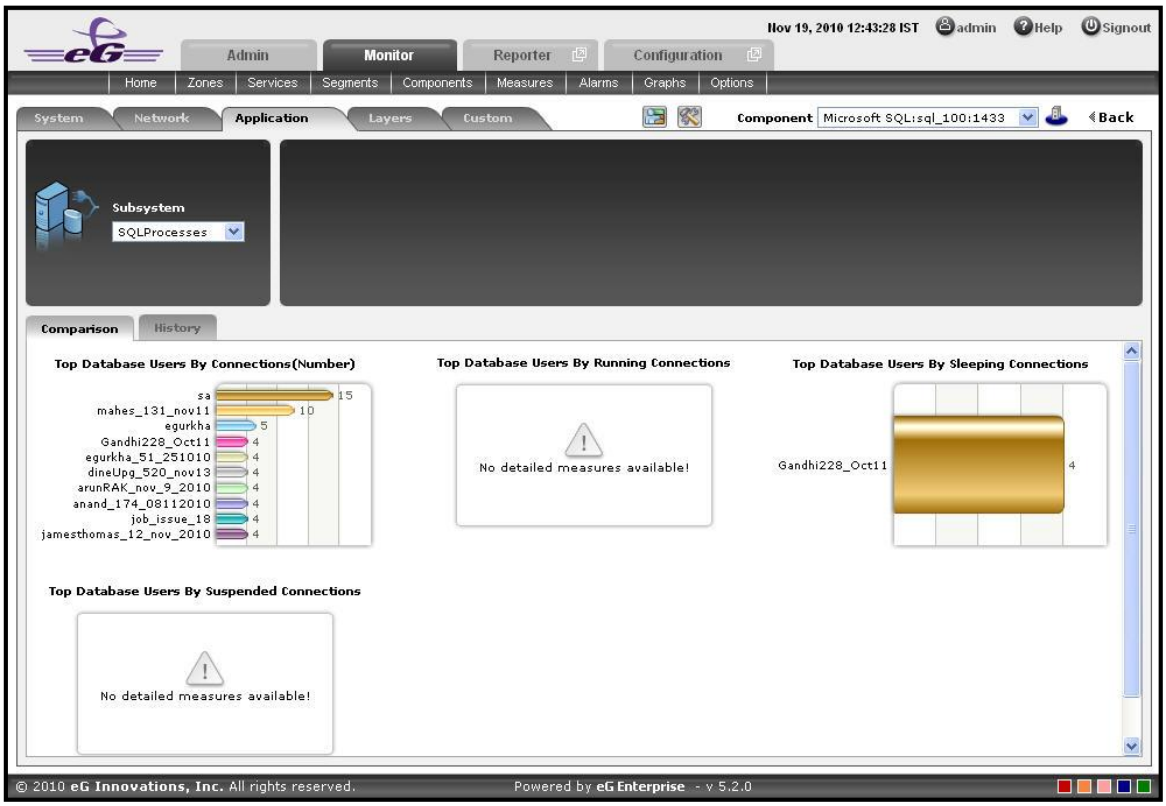


Figure 3.54: The Comparison tab page of the SQLProcesses Dashboard

The contents of this dashboard are as follows:

1. The **Comparison** tab page provides a series of top 10 charts, using which you can isolate the top Database Users for various resource-intensive processes. This default list of measures for top-n chart generation can be overridden by following the steps discussed below:

    - Click on the icon at the top of the **Application Dashboard**. In the **Dashboard Settings** window that appears, select **Application** from the **Module** list, and **SQLProcesses** from the **Sub-System** list.
    - To add new measures for which top-n graphs are to be displayed in the **Comparison** tab page, first, pick the **Comparison Graph** option from the **Add/Delete Measures for** list. Upon selection of this option, the pre-configured measures for comparison graphs will appear in the **Existing Value(s)** list.
    - Next, select the **Test** that reports the said measure, pick the measure of interest from the **Measures** list, provide a **Display** name for the measure, and click the **Add** button to add the chosen measure to the **Existing Value(s)** list.
    - If you want to delete one/more measures for which comparison graphs pre-exist in the **Comparison** tab page, then, as soon as you choose the **Comparison Graph** option from the **Add/Delete Measures for** list, pick any of the displayed measures from the **Existing Value(s)** list, and click the **Delete** button.
    - Finally, click the **Update** button to register the changes.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

2. To view the complete list of database users for each process, simply click on the corresponding graph in Figure 3.54. This enlarges the graph as depicted by Figure 3.55.

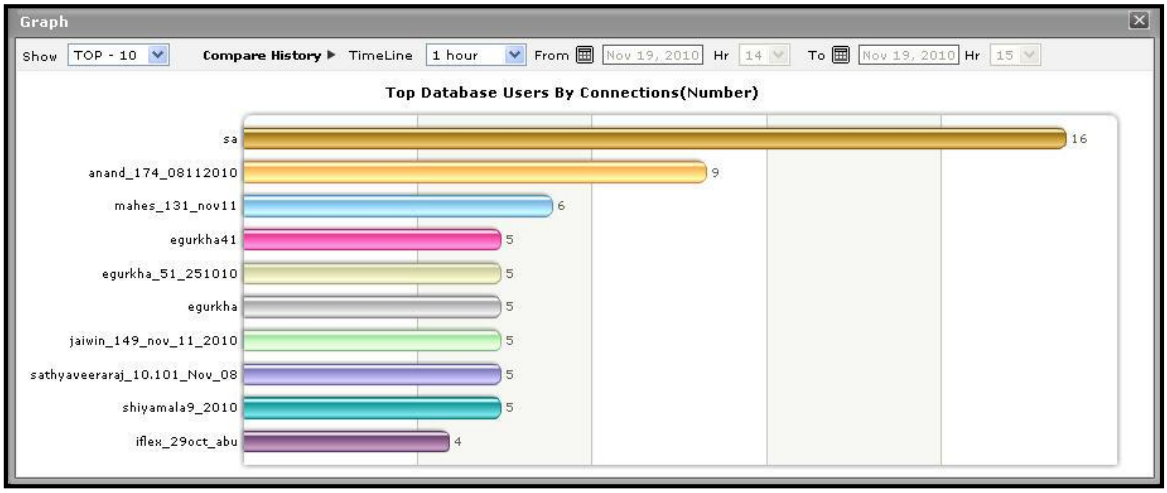


Figure 3.55: The expanded Comparison graph in the SQLProcesses dashboard

3. Though the enlarged graph lists the database users, by default, you can customize the enlarged graph to display the details of only a few of the best/worst-performing processes by picking a **TOP-N** or **LAST-N** option from the **Show** list in Figure 3.55.

4. Another default aspect of the enlarged graph is that it pertains to the current period only. Sometimes however, you might want to know what occurred during a point of time in the past; for instance, while trying to understand the reason behind a sudden slowdown in a particular process on a particular day last week, you might want to first determine which process has behaved abnormally on the same day. To figure this out, the enlarged graph allows you to compare the historical performance of the processes. For this purpose, click on the **Compare History** link in Figure 3.55 and select the **TimeLine** of your choice.

5. The **History** tab page below, by default, provides a series of measure graphs that reveal how the process has been performing over the default duration of the last 24 hours. If there is a sudden slowdown in the process, it could indicate that the server is experiencing issues with the concerned process. In such a case, a look at these measure graphs will help you figure out when exactly the bottleneck surfaced - did it happen suddenly or is it a condition that has become worse with time?

   The default duration of 24 hours can be overridden using the procedure discussed below:

   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

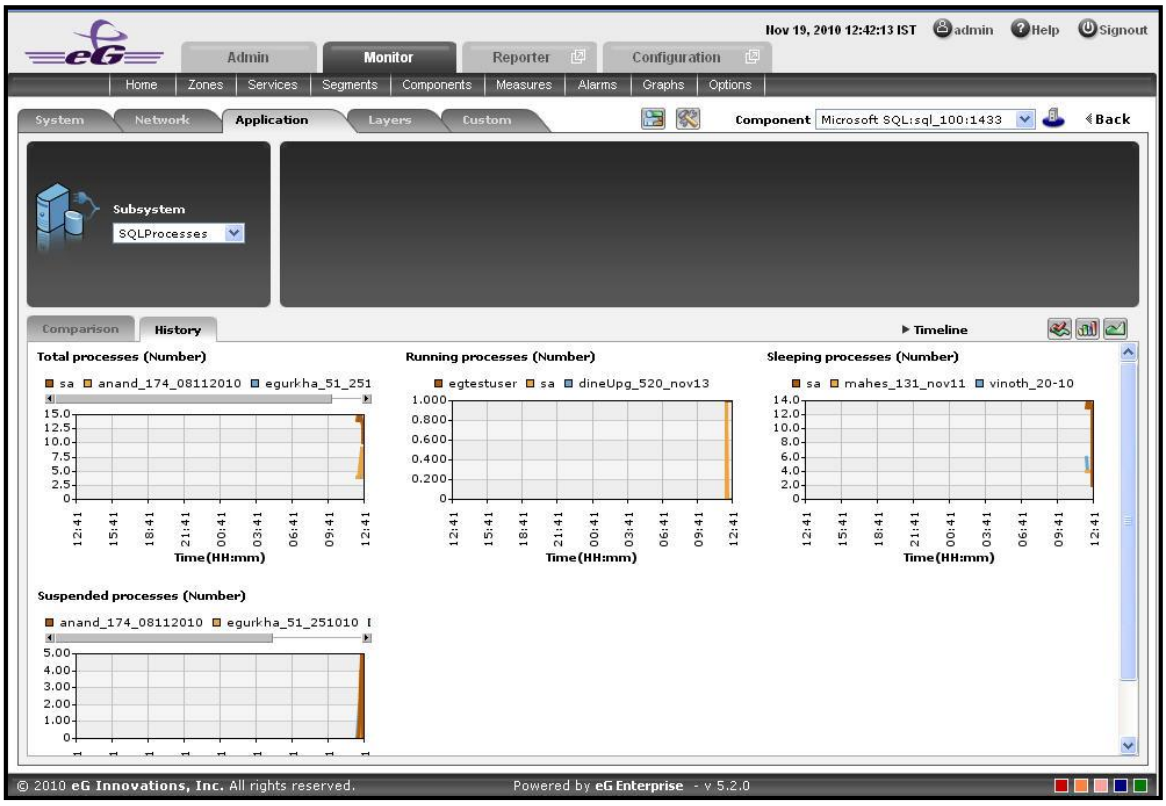


Figure 3.56: The History tab page of the SQLProcesses dashboard

6. If need be, you can even alter the timeline of all these measure graphs so that you can analyze performance across days and weeks; for this, simply click the **Timeline** link at the right, top corner of the **History** tab page and change the timeline for the graphs using the calendar that pops out. To change the timeline of a single graph alone, simply click on that graph to enlarge it, and then modify the **Timeline** of the graph in the enlarged mode. In the enlarged mode, you can even change the dimension of the measure graph (**3D** / **2D**).

7. To determine the service level achievements / slippages of the process, you need to view summary graphs of the measures and not the default measure graphs. For this, just click on the icon at the right, top corner of the **History** tab page.

8. The summary graphs reveal the percentage of time the process experienced problems in the database. Besides revealing the efficiency of your administrative staff in recognizing bottlenecks and mitigating them, these summary graphs also indicate whether the class loader has been able to maintain the assured performance levels during the default duration of 24 hours.

To override this default duration, follow the steps below:

- Click on the icon at the top of the **Application Dashboard**.
- In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
- Then, choose a **Timeline** for the graph.
- Finally, click the **Update** button.

9. In case of the summary graphs too, you can change the **Timeline** of all graphs by clicking on the **Timeline** link at the right, top corner of the **History** tab page. To alter the timeline of a single graph, here again, you will have to click on that graph, enlarge it, and modify the timeline. Also, by default, hourly summaries are plotted in the summary graph; you can configure these graphs to plot daily/monthly summaries instead by picking the relevant option from the **Duration** list in the enlarged mode.
10. To analyze past trends in the behavior of the processes, click on the icon at the right, top corner of the **History** tab page. These trend graphs, by default, plot the minimum and maximum values that every measure registered during each hour of the last 24 hours (by default). By carefully observing these past trends, you can effectively analyze the workload of the process, predict future workloads accordingly, and suggest measures to enhance the efficiency of the process. Here again, you can change the timeline of all graphs using the **Timeline** link, or just a particular graph by clicking on it and enlarging it.

    For changing the default duration (of 24 hours) of the trend graphs, do the following:

    - Click on the icon at the top of the **Application Dashboard**.
    - In the **Dashboard Settings** window that appears, select **Trend Graph** from the **Default Timeline for** list.
    - Then, choose a **Timeline** for the graph.
    - Finally, click the **Update** button.

11. In addition, when a trend graph is enlarged, it is not just the **Timeline** that you can modify. The **Duration** of the graph can also be altered. By default, trend graphs reveal only the hourly trends in performance. By picking the relevant option from the **Duration** list, you can ensure that the trend graph in question plots daily/monthly trend values instead. Also, in the enlarged mode, the **Graph type** can also be modified. Since the default **Graph type** is **Min/Max**, the trend graph, by default, reveals the minimum and maximum values registered by a measure. If need be, you can select the **Avg** or **Sum** option from the **Graph type** list to plot average trend values of a measure or sum of trends (as the case may be) in the graph.

    Note:

    In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.

12. At any point in time, you can switch to the measure graphs by clicking on the button.
13. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:

    - Click the button at the top of the dashboard.

- The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.44, pick **Application**, choose **SQLProcesses** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.
- The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.
- To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
- Next, select the **Measure** of interest.
- Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
- This will add a new measure, summary, and trend graph for the chosen measure to the **History** tab page.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

3.7.5 SQLDatabases

Select the **SQLDatabases** option from the **Subsystem** list to know how efficiently the databases are used by the MS SQL application and how well the database has been responding to the queries from other applications. Upon selection, Figure 3.57 will appear.

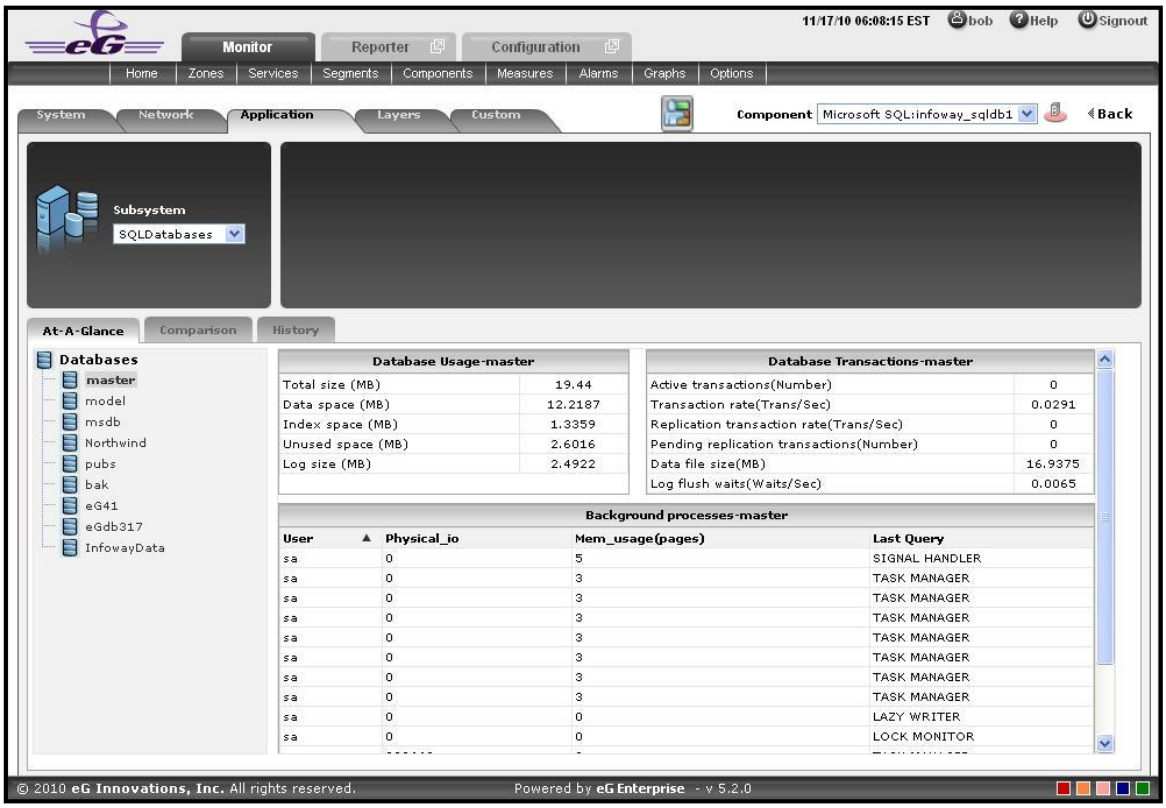


Figure 3.57: The At-A-Glance tab page of the SQLDatabases Dashboard

The contents of this dashboard are as follows:

1. The **At-A-Glance** tab page lists the databases that are available at present in the MS SQL application, in the **Databases** section. Upon selecting a database, the detailed measures corresponding to that particular database is available in a context-sensitive right panel. For instance, if master database is selected, then in the right panel, the **Database Usage** section will provide the usage details like **Total size**, etc. Also the **Database Transactions** section provides the relevant transaction measures The **Background processes** section will list out all the background process available for the users who are accessing that particular database. By default, the background process list provided by this section is sorted in the alphabetical order of the User. If need be, you can change the sort order so that the processes are arranged in, say, the descending order of values displayed in the **Physical_io** column - this column displays the physical io location of each user connected to the database. To achieve this, simply click on the column heading – **Physical_io**. Doing so tags the **Physical_io** label with a **down arrow** icon - this icon indicates that the background process list is currently sorted in the descending order of physical io location. To change the sort order to 'ascending', all you need to do is just click again on the **Physical_io** label or the **down arrow** icon. Similarly, you can sort the process list based on any column available in the **Background processes** section. Likewise the right panel may consist of **Running processes** and **Sleeping processes** sections, if those particular processes are available for execution in the selected database. Similarly **CPU cycles rate** section may also be available for the databases. This section reveals the number of CPU cycles taken by the server for each host available in the target MS SQL application. The columns available in this section can also be sorted in the same manner as that of the **Background processes** section.

2. The Comparison tab page that follows the At-A-Glance tab page provides a series of top-10 charts, using which you can isolate the databases that are leading the lot in the following fields: Size, Log Size and Active transactions. This default list of fields (i.e., measures) for top-n chart generation can be overridden by following the steps discussed below:

    - Click on the  icon at the top of the **Application Dashboard**. In the **Dashboard Settings** window that

appears, select **Application** from the **Module** list, and **SQLDatabases** from the **Sub-System** list.

- To add new measures for which top-n graphs are to be displayed in the **Comparison** tab page, first, pick the **Comparison Graph** option from the **Add/Delete Measures for** list. Upon selection of this option, the pre-configured measures for comparison graphs will appear in the **Existing Value(s)** list.
- Next, select the **Test** that reports the said measure, pick the measure of interest from the **Measures** list, provide a **Display** name for the measure, and click the **Add** button to add the chosen measure to the **Existing Value(s)** list.
- If you want to delete one/more measures for which comparison graphs pre-exist in the **Comparison** tab page, then, as soon as you choose the **Comparison Graph** option from the **Add/Delete Measures for** list, pick any of the displayed measures from the **Existing Value(s)** list, and click the **Delete** button.
- Finally, click the **Update** button to register the changes.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

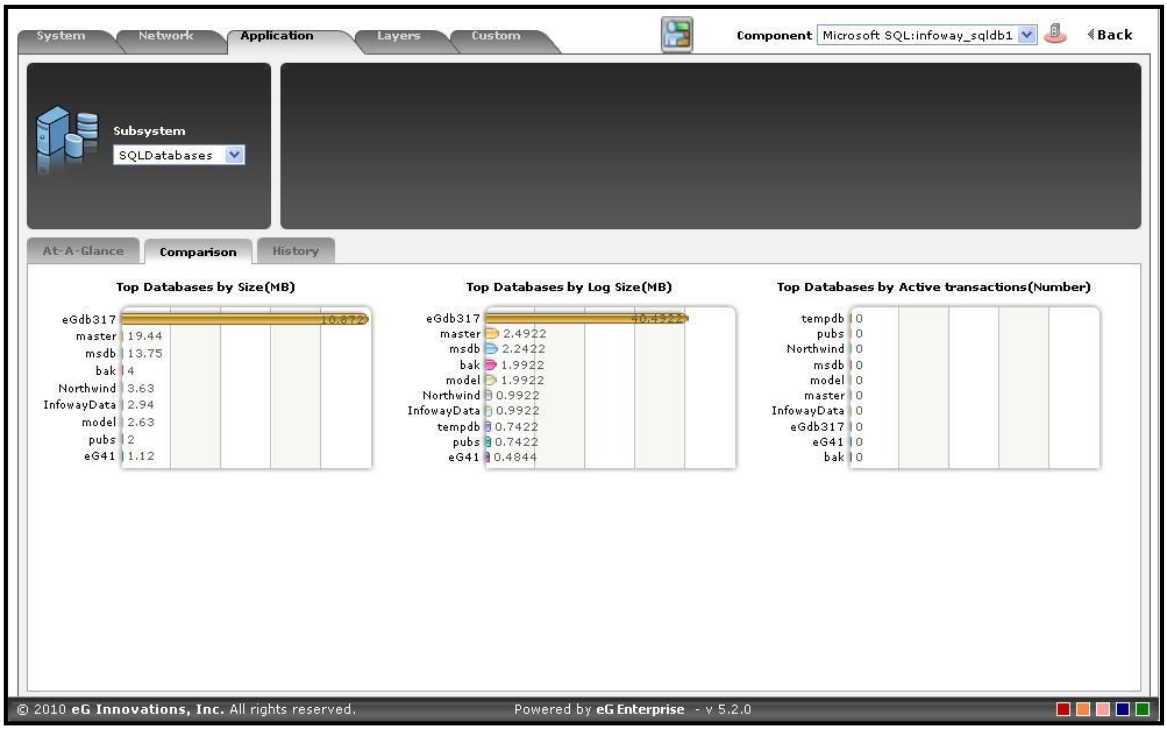


Figure 3.58: The Comparison tab page of the SQLDatabases dashboard

3. To view the complete list of databases, simply click on the corresponding graph in Figure 3.58. This enlarges the graph as depicted by Figure 3.59.

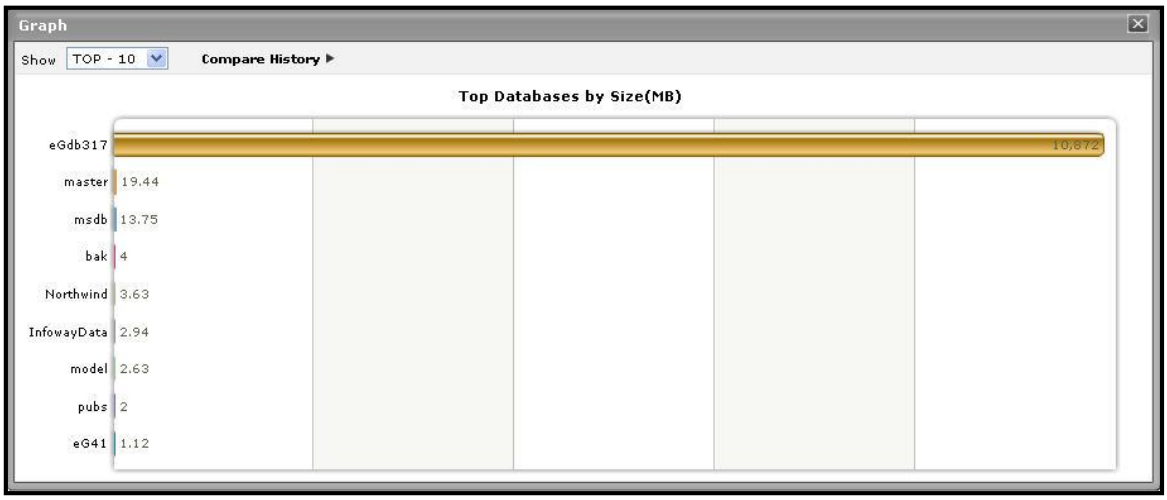


Figure 3.59: The expanded top-n graph in the Comparison tab page of the SQLDatabases Dashboard

4. Though the enlarged graph lists all the databases by default, you can customize the enlarged graph to display the details of only a few of the best/worst-performing databases by picking a **TOP-N** or **LAST-N** option from the **Show** list in Figure 3.59.

5. Another default aspect of the enlarged graph is that it pertains to the current period only. Sometimes however, you might want to know what occurred during a point of time in the past; for instance, while trying to understand the reason behind a sudden increase in the Size of the databases on a particular day last week, you might want to first determine which database has behaved abnormally on the same day. To figure this out, the enlarged graph allows you to compare the historical performance of databases. For this purpose, click on the **Compare History** link in Figure 3.59 and select the **TimeLine** of your choice.

6. The **History** tab page below, by default, provides a series of measure graphs that reveal how well the databases have been performing over the default duration of the last 24 hours. If the performance of the databases dramatically decreases, it could indicate that the databases are experiencing performance issues. In such a case, a look at these measure graphs will help you figure out when exactly the bottleneck surfaced - did it happen suddenly or is it a condition that has become worse with time?

   The default duration of 24 hours can be overridden using the procedure discussed below:

   Click on the icon at the top of the **Application Dashboard**.

   In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.

   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

Figure 3.60: The History tab page of the SQLDatabases dashboard

7. If need be, you can even alter the timeline of all these measure graphs so that you can analyze performance across days and weeks; for this, simply click the **Timeline** link at the right, top corner of the **History** tab page and change the timeline for the graphs using the calendar that pops out. To change the timeline of a single graph alone, simply click on that graph to enlarge it, and then modify the **Timeline** of the graph in the enlarged mode. Though the enlarged graph lists all the databases by default, you can customize the enlarged graph to display the details of only a few of the best/worst-performing databases by picking a **TOP-N** or **LAST-N** option from the **Show** list. In the enlarged mode, you can even change the dimension of the measure graph (**3D** / **2D**).

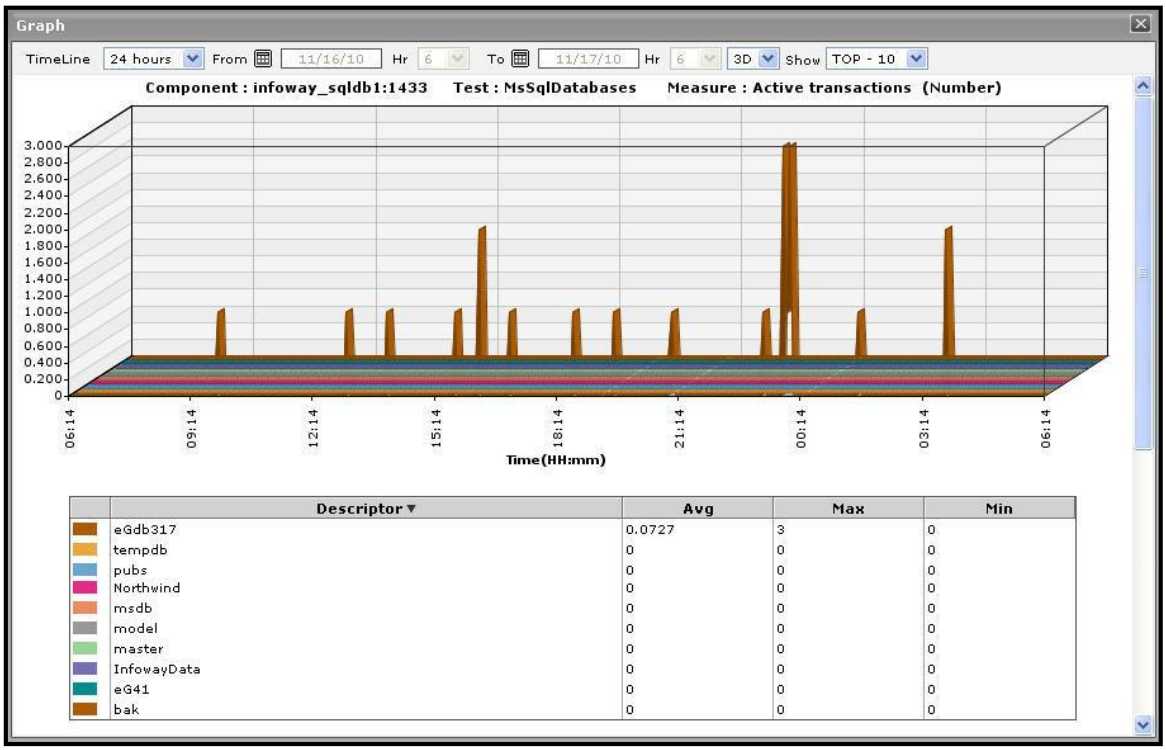


Figure 3.61: An enlarged measure graph in the History tab page of the SQLDatabases dashboard

8. To determine the service level achievements of the databases, you need to view summary graphs of the measures and not the default measure graphs. For this, just click on the icon at the right, top corner of the **History** tab page. Figure 3.62 then appears.

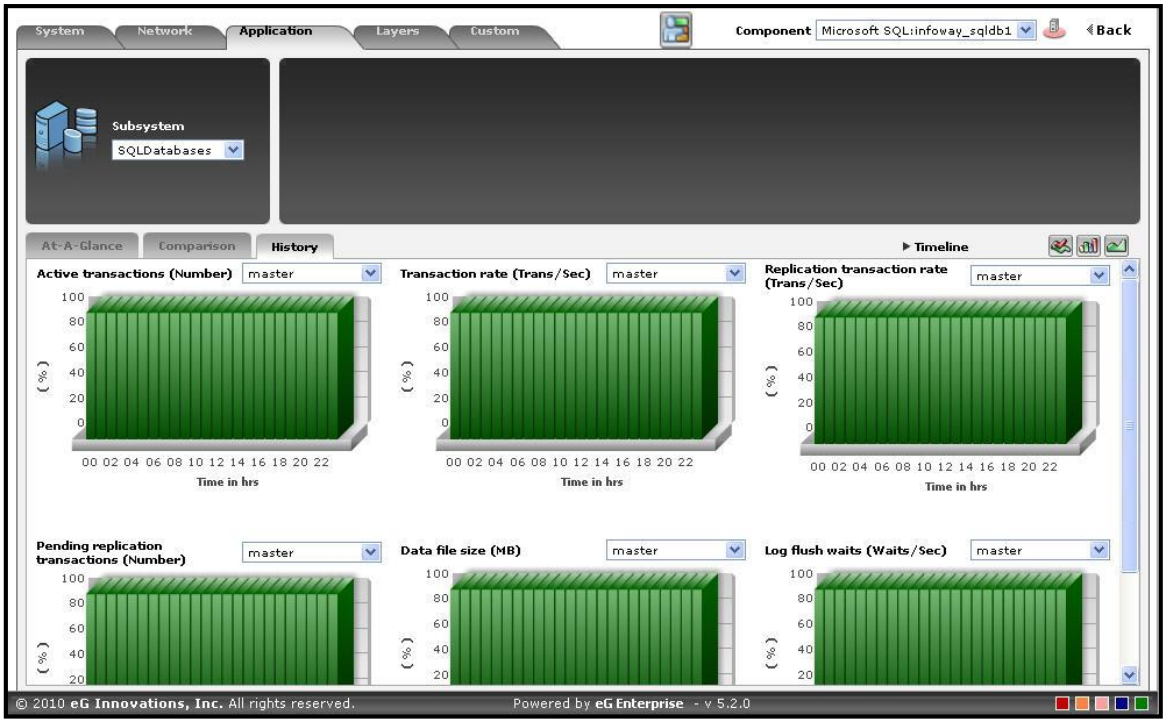


Figure 3.62: Summary graphs displayed in the SQLDatabases Dashboard

9. The summary graphs displayed in Figure 3.62 reveal the percentage of time the MS SQL application experienced problems in one of its databases. Besides revealing the efficiency of your administrative staff in recognizing bottlenecks and mitigating them, these summary graphs also indicate whether the databases has been able to maintain the assured performance levels during the default duration of 24 hours.

   To override this default duration, follow the steps below:

   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

10. In case of the summary graphs too, you can change the **Timeline** of all graphs by clicking on the **Timeline** link at the right, top corner of the **History** tab page. To alter the timeline of a single graph, here again, you will have to click on that graph, enlarge it, and modify the timeline. Also, by default, hourly summaries are plotted in the summary graph; you can configure these graphs to plot daily/monthly summaries instead by picking the relevant option from the **Duration** list in the enlarged mode.

11. To analyze past trends in the performance of the databases, click on the icon at the right, top corner of the **History** tab page. Figure 3.63 will then appear.

Figure 3.63: Trend graphs displayed in the SQLDatabases Dashboard

12. These trend graphs, by default, plot the minimum and maximum values that every measure registered during each hour of the last 24 hours (by default). Using such graphs, you can accurately point to the time windows during which there was a lull in the transaction of the selected database. Here again, you can change the timeline of all graphs using the **Timeline** link in Figure 3.63, or just a particular graph by clicking on it and enlarging it.

    For changing the default duration (of 24 hours) of the trend graphs, do the following:

- Click on the icon at the top of the **Application Dashboard**.
- In the **Dashboard Settings** window that appears, select **Trend Graph** from the **Default Timeline for** list.
- Then, choose a **Timeline** for the graph.
- Finally, click the **Update** button.

13. In addition, when a trend graph is enlarged, it is not just the **Timeline** that you can modify. The **Duration** of the graph can also be altered. By default, trend graphs reveal only the hourly trends in performance. By picking the relevant option from the **Duration** list, you can ensure that the trend graph in question plots daily/monthly trend values instead. Also, in the enlarged mode, the **Graph type** can also be modified. Since the default **Graph type** is **Min/Max**, the trend graph, by default, reveals the minimum and maximum values registered by a measure. If need be, you can select the **Avg** or **Sum** option from the **Graph type** list to plot average trend values of a measure or sum of trends (as the case may be) in the graph.

> **Note:**
>
> In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.

14. At any point in time, you can switch to the measure graphs by clicking on the button.
15. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:
    - Click the button at the top of the dashboard.
    - The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.44, pick **Application**, choose **SQLDatabases** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.
    - The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.
    - To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
    - Next, select the **Measure** of interest.
    - Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
    - This will add a new measure, summary, and trend graph for the chosen measure to the **History** tab page.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

3.7.6 SQLApplications

Select the **SQLApplications** option from the **Subsystem** list to know how well the applications are being used by the MS SQL application, Upon selecting this **Subsystem**, Figure 3.64 will appear.

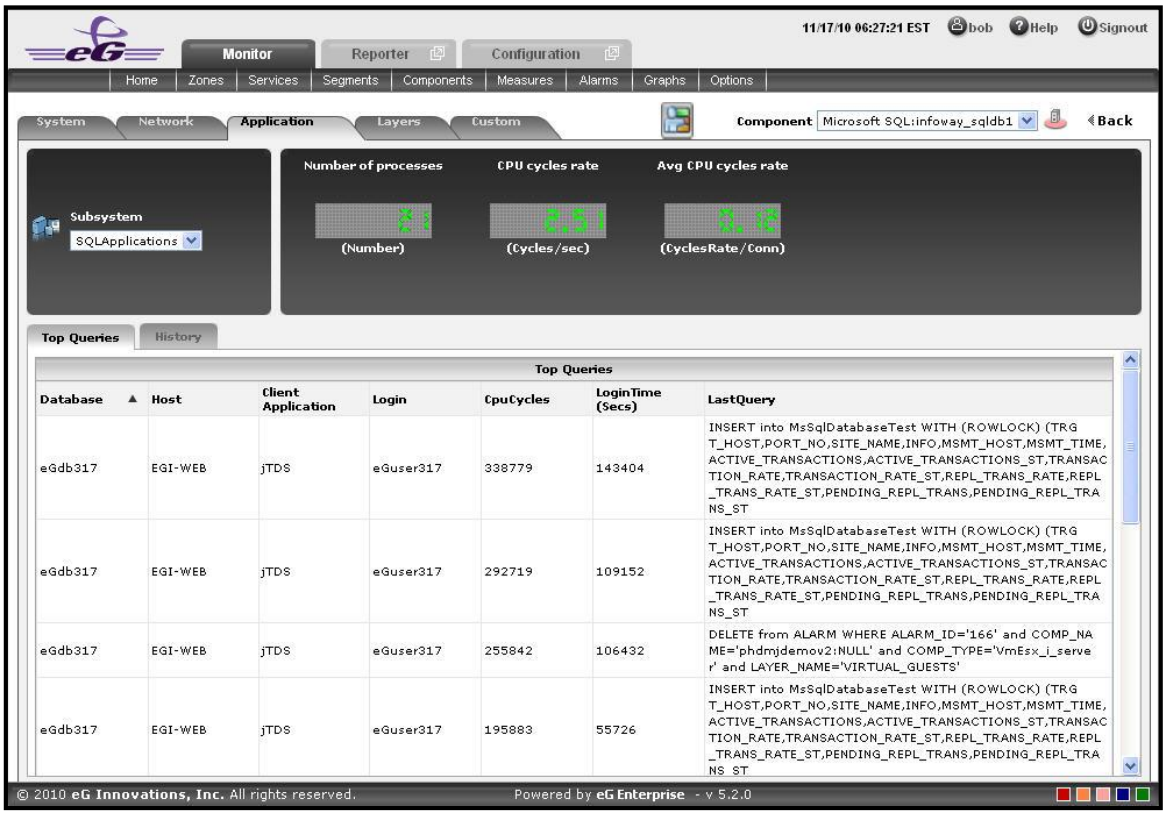


Figure 3.64: The SQLApplications Dashboard

The contents of the **SQLApplications** dashboard are as follows:

1. For an easy and single glance view of certain measures like Number of processes, CPU cycle rate and Avg CPU cycles rate, a digital graph section is included. When a digital graph is clicked, the corresponding layer-test combination which reports that particular measure will be displayed from the layer model page of the MS SQL application.

2. From the **Top Queries** tab page, you can infer the queries that have been made to the databases from the client application. By default, the queries listed in this tab page are sorted in alphabetical order of the Database. If need be, you can change the sort order so that the databases are arranged in, say, the descending order of values displayed in the **CPUCycles** column. To achieve this, simply click on the column heading – **CPUCycles**. Doing so tags the **CPUCycles** label with a **down arrow** icon - this icon indicates that this tab page is currently sorted in the descending order of the CPU cycles. To change the sort order to 'ascending', all you need to do is just click again on the **CPUCycles** label or the **down arrow** icon. Similarly, you can sort the process list based on any column available in this tab page.

3. The **History** tab page, by default, displays time-of-day graphs revealing how well the application has been performing over a default period of 24 hours. If the eG agent reports any abnormal behavior of the MS SQL application, these graphs will help determine when exactly in the last 24 hours the abnormality occurred. This default duration of 24 hours can be overridden using the following steps:

   - Click on the  icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

Figure 1.1: The history tab page of the MS SQL Applications Dashboard

4. A careful study of this graph over time periods longer than 24 hours, can reveal intermittent breaks (if any) in

the number of processes and the CPU cycle rate of the databases. To ensure that all graphs plot values for longer time periods, click on the **Timeline** link at the right, top corner of the **History** tab page, and then change the timeline using the calendar that pops out. To modify the timeline for a particular graph alone, click on the graph to enlarge it, and alter the timeline in the enlarged mode. Besides the timeline, you can even change the graph dimension (**3D** / **2D**) in the enlarged mode.

5. Sometimes, you might have to periodically determine the percentage of time for which the MS SQL application experienced problems relating to the databases. To determine such problems, summary graphs of the SQL applications measures are useful. To view summary graphs in the **History** tab page, click on the  icon at the right, top corner of the **History** tab page. These summary graphs reveal the percentage of time during the last 24 hours (by default) the MS SQL application has experienced issues related to the SQLapplications. To override this default timeline, do the following:
   - Click on the  icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

6. To perform the summary analysis over a broader time window, click on the **Timeline** link at the right, top corner of the **History** tab page and change the timeline; this will alter the timeline for all the graphs. To change the timeline of a particular graph alone, click on the graph to enlarge it, and then alter its timeline. Also, by default, hourly summaries are plotted in the summary graph; you can configure these graphs to plot daily/monthly summaries instead by picking the relevant option from the **Duration** list in the enlarged mode. Here again, the graph dimension (**3D** / **2D**) can be altered.
7. Similarly, you can analyze uptime trends by viewing trend graphs in the **History** tab page. For this, click on the  icon at the right, top corner of the tab page. These trend graphs, by default, plot the minimum and maximum values registered by every SQL application-related measure during every hour for the last 24 hours. The default duration of 24 hours can be overridden using the procedure discussed below:
   - Click on the  icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

8. To perform trend analysis over a longer time span, click on the **Timeline** link at the right, top corner of the **History** tab page and change the timeline; this will alter the timeline for all the graphs. To change the timeline of a particular graph alone, click on the graph to enlarge it, and then alter its timeline. In addition to the timeline, the graph dimension (**3D** / **2D**), the graph **Duration**, and the **Graph type** can also be changed in the enlarged mode. By default, the graph **Duration** is **Hourly**, indicating that trend graphs plot hourly trend values by default. To ensure that these graphs plot the daily/monthly trend values instead, select the relevant option from the **Duration** list. Similarly, as already mentioned, trend graphs plot only the minimum and maximum values registered by a measure during the specified timeline. Accordingly, the **Graph type** is set to **Min/Max** by default in the enlarged mode. If you want the trend graph to plot the average trend values instead, set the **Graph type** to **Avg**. On the other hand, to configure the trend graph to plot the sum of trends set the **Graph type** to **Sum**.

Note:

In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.

9. At any point in time, you can switch to the measure graphs by clicking on the button.
10. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:
    - Click the button at the top of the dashboard.
    - The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.44, pick **Application**, choose **SQLApplication** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.
    - The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.
    - To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
    - Next, select the **Measure** of interest.
    - Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
    - This will add a new measure, summary, and trend graph for the chosen measure to the **History** tab page.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

3.7.7 SQLService

The **SQLService** option is picked from the **Subsystem** list to know about the overall performance of the MS SQL application such as server health, session activity, access capacity of the server etc, in detail. Upon selecting this **Subsystem**, Figure 3.65 will appear.

Figure 3.65: The SQLService Dashboard

The contents of the **SQLService** dashboard are as follows:

1. A digital display section for some critical session activity and access capability measures is included for an easy and single glance view. When a digital display is clicked, the corresponding layer-test combination which reports that particular measure will be displayed from the layer model page of the MS SQL application.
2. The **History** tab page as shown in Figure 3.65, by default, displays time-of-day graphs revealing how well the application has been performing over a default period of 24 hours. If the eG agent reports any abnormal behavior of the MS SQL application, these graphs will help determine when exactly in the last 24 hours the abnormality occurred. This default duration of 24 hours can be overridden using the following steps:
   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

3. A careful study of this graph over time periods longer than 24 hours, can reveal intermittent breaks (if any) in the session activity and access capability measures of this MS SQL application. To ensure that all graphs plot values for longer time periods, click on the **Timeline** link at the right, top corner of the **History** tab page, and then change the timeline using the calendar that pops out. To modify the timeline for a particular graph alone, click on the graph to enlarge it (see Figure 3.66), and alter the timeline in the enlarged mode. Besides the timeline, you can even change the graph dimension (**3D** / **2D**) in the enlarged mode.

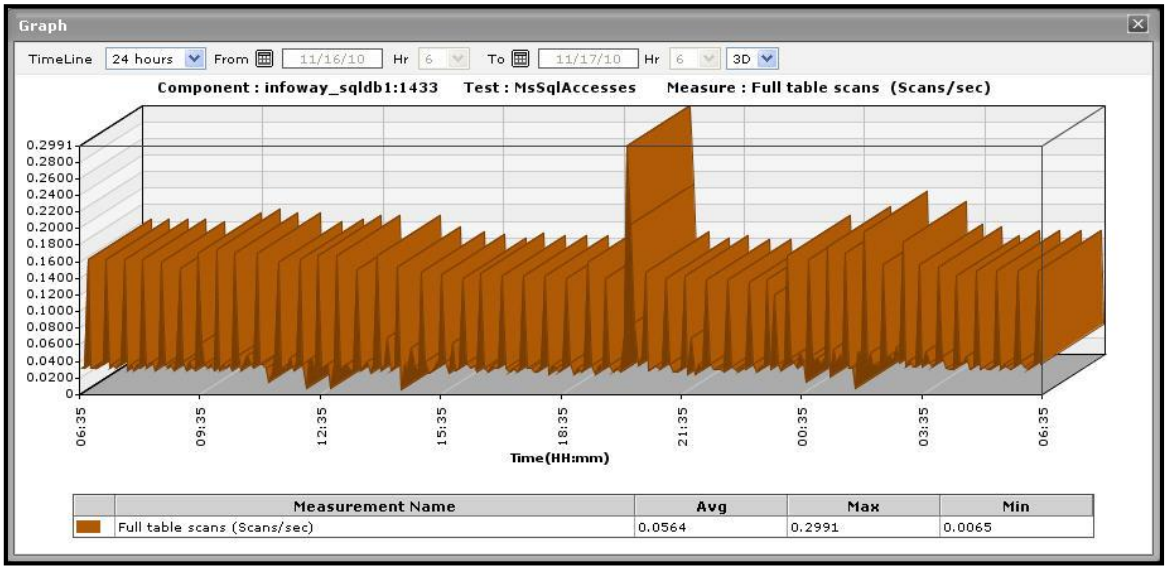


Figure 3.66: The enlarged history graph of the SQLService dashboard

4. Sometimes, you might have to periodically determine the percentage of time for which the MS SQL application experienced problems relating to the databases. To determine such problems, summary graphs are useful. To view summary graphs in the **History** tab page, click on the icon at the right, top corner of the **History** tab page. These summary graphs reveal the percentage of time during the last 24 hours (by default) the MS SQL application has experienced issues related to the SQL service. To override this default timeline, do the following:
   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

Figure 3.67: The summay graphs for the SQLService dashboard

5. To perform the summary analysis over a broader time window, click on the **Timeline** link at the right, top corner of the **History** tab page and change the timeline; this will alter the timeline for all the graphs. To change the timeline of a particular graph alone, click on the graph to enlarge it, and then alter its timeline. Also, by default, hourly summaries are plotted in the summary graph; you can configure these graphs to plot daily/monthly summaries instead by picking the relevant option from the **Duration** list in the enlarged mode. Here again, the graph dimension (**3D** / **2D**) can be altered.

6. Similarly, you can analyze uptime trends by viewing trend graphs in the **History** tab page. For this, click on the icon at the right, top corner of the tab page. These trend graphs, by default, plot the minimum and maximum values registered by every SQL application-related measure during every hour for the last 24 hours. The default duration of 24 hours can be overridden using the procedure discussed below:

   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

7. To perform trend analysis over a longer time span, click on the **Timeline** link at the right, top corner of the **History** tab page and change the timeline; this will alter the timeline for all the graphs. To change the timeline of a particular graph alone, click on the graph to enlarge it, and then alter its timeline. In addition to the timeline, the graph dimension (**3D** / **2D**), the graph **Duration**, and the **Graph type** can also be changed in the enlarged mode. By default, the graph **Duration** is **Hourly**, indicating that trend graphs plot hourly trend values by default. To ensure that these graphs plot the daily/monthly trend values instead, select the relevant option from the **Duration** list. Similarly, as already mentioned, trend graphs plot only the minimum and maximum values registered by a measure during the specified timeline. Accordingly, the **Graph type** is set to **Min/Max** by default in the enlarged mode. If you want the trend graph to plot the average trend values instead, set the **Graph type** to **Avg**. On the

other hand, to configure the trend graph to plot the sum of trends set the **Graph type** to **Sum**.

> **Note:**
>
> In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.

8. At any point in time, you can switch to the measure graphs by clicking on the  button.
9. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:
   - Click the  button at the top of the dashboard.
   - The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.44, pick **Application**, choose **SQLService** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.
   - The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.
   - To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
   - Next, select the **Measure** of interest.
   - Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
   - This will add a new measure, summary, and trend graph for the chosen measure to the **History** tab page.

> **Note:**
>
> Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

4

Monitoring DB2 UDB Servers

DB2® Universal Database (UDB) Enterprise Server Edition (ESE) is a multiuser version of DB2 that allows you to create and manage single-partitioned or partitioned database environments. Partitioned database systems can manage high volumes of data and provide benefits such as increased performance and high availability.

Before attempting to understand how to monitor a DB2 UDB server, it is essential to get acquainted with its architecture.

On the client side, either local or remote applications, or both, are linked with the DB2 Universal Database™ client library. Local clients communicate using shared memory and semaphores; remote clients use a protocol such as Named Pipes (NPIPE), TCP/IP, NetBIOS, or SNA.

On the server side, activity is controlled by engine dispatchable units (EDUs). EDUs are implemented as threads in a single process on Windows®-based platforms and as processes on UNIX®. DB2 agents are the most common type of EDUs. These agents perform most of the SQL processing on behalf of applications. Prefetchers and page cleaners are other common EDUs.

- **Prefetchers** retrieve data from disk and move it into the buffer pool before applications need the data. For example, applications needing to scan through large volumes of data would have to wait for data to be moved from disk into the buffer pool if there were no data prefetchers. Agents of the application send asynchronous read-ahead requests to a common prefetch queue. As prefetchers become available, they implement those requests by using big-block or scatter-read input operations to bring the requested pages from disk to the buffer pool. If you have multiple disks for storage of the database data, the data can be striped across the disks. Striping data lets the prefetchers use multiple disks at the same time to retrieve data.

- **Page cleaners** move data from the buffer pool back out to disk. Page cleaners are background EDUs that are independent of the application agents. They look for pages from the buffer pool that are no longer needed and write the pages to disk. Page cleaners ensure that there is room in the buffer pool for the pages being retrieved by the prefetchers.

- A set of subagents might be assigned to process the client application requests. Multiple subagents can be assigned if the machine where the server resides has multiple processors or is part of a partitioned database. For example, in a symmetric multiprocessing (SMP) environment, multiple SMP subagents can exploit the many processors.

All agents and subagents are managed using a pooling algorithm that minimizes the creation and destruction of EDUs.

Buffer pools are areas of database server memory where database pages of user table data, index data, and catalog data are temporarily moved and can be modified. Buffer pools are a key determinant of database performance because data can be accessed much faster from memory than from disk. If more of the data needed by applications is present in a buffer pool, less time is required to access the data than to find it on disk.

The configuration of the buffer pools, as well as prefetcher and page cleaner EDUs, controls how quickly data can be

accessed and how readily available it is to applications. Without the independent prefetchers and the page cleaner EDUs, the application agents would have to do all of the reading and writing of data between the buffer pool and disk storage.

All these integral components need to function like well-oiled machines in order to enable the DB2 server to quickly and efficiently process data requests. The slightest of problems in the configuration or operation of these components, if not swiftly resolved, can severely degrade database performance, and can even render the database server inaccessible to users. In order to prevent such adversities, it is imperative that the DB2 server is monitored round-the-clock, and problems brought to the attention of the administrators before irreparable damage is caused.

Moreover, a Database Partitioning Feature (DPF) is additionally available in the DB2 UDB Enterprise Server Edition (ESE). With DPF your database is scalable as you can add new machines and spread your database across them. This means more CPUs, more memory and more disks from each of the additional machines for your database.

If DB2 is installed in a DPF environment, the management challenges grow! Instead of monitoring a database on a single machine, each database now has to be monitored across all its partitons, which may be spread across multiple machines, to ascertain the health of the transactions to the database.

To address the unique monitoring requirements of the different DB2 installations, the eG Enterprise suite prescribes three exclusive monitoring models for the IBM DB2 server – one for version 8.0 (and above) of the DB2 server, one for versions 6.0 and 7.0, and one for the DB2 server that is installed in a DPF environment. These models thoroughly scrutinize every layer of a DB2 server for issues, and proactively alert administrators of probable problem conditions.

The sections to come discuss each of these models in great detail.

4.1 Monitoring IBM DB2 Server Version 8.0 (and above)

Figure 4.1 depicts the specialized *DB2 UDB* monitoring model that eG Enterprise offers for monitoring an IBM DB2 server ver. 8.0 (or above).

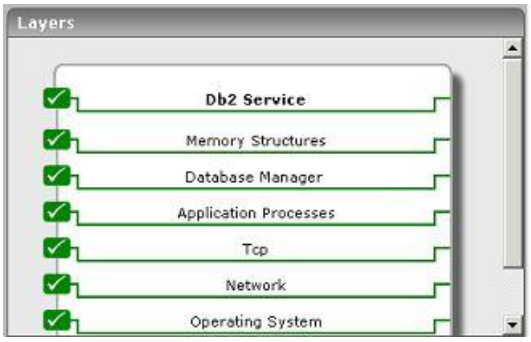


Figure 4.1: The layer model of an IBM DB2 server version 8.0 (or above)

Every layer of Figure 4.1 is mapped to a wide variety of tests that a single eG agent executes on the DB2 server; these tests extract a wealth of performance metrics from the DB2 server.

One of the key qualities of this eG agent is that it is capable of measuring the internal health of the DB2 server from anywhere in the environment! In other words, the eG agent need not be present on the DB2 server to monitor its internal operations. The eG Enterprise system supports "agentless monitoring" of a DB2 server, by means of which, the agent can be installed on any remote host in your environment, and can be easily configured to pull out statistics of interest from within the DB2 server.

The statistics so collected enable administrators to find quick and accurate answers to the following performance queries:

- Is the DB2 database server available? How quickly does it respond to user requests?

- What is the current connection load on the DB2 database manager? How many of these connections are local, and how many are remote connections?
- How quickly does the DB2 server process requests from client applications? Are there sufficient agents in the agent pool to service all the client requests?
- Are the agents in the pool utilized optimally, or are too many agents idle?
- Does the database server perform sorting efficiently? Has adequate sort heap space been allocated to the database manager to enable this?
- Are sort overflows kept at a minimum?
- Does sorting take too long?
- Are lock escalations occurring too frequently on the database?
- Are too many deadlocks been detected?
- Do applications obtain locks quickly, or do they have to wait too long for locks?
- Are the database buffer pools adequately sized?
- Are the page cleaners and prefetchers been utilized effectively?
- Are too many rollbacks happening on the database?
- Have too many SQL statements failed?

The sections to come elaborate on each layer of Figure 4.1, the tests associated with them, and the statistics they extract.

4.1.1 The Database Manager Layer

Using the tests associated with the **Database Manager** layer, the following can be monitored:

- Critical activities performed by the database manager
- Client connections to the database manager
- Usage of the agent pools on the database manager

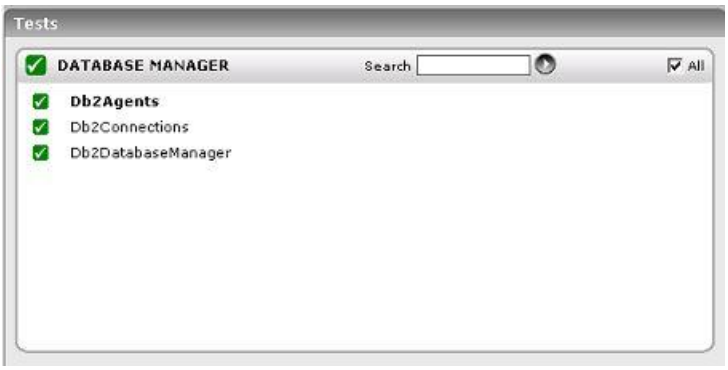


Figure 4.2: The tests associated with the Database Manager layer

4.1.1.1 Db2 Agents Test

An agent is a process or thread that carries out the requests made by a client application. Each connected application is served by exactly 1 coordinator agent and possibly, a set of subordinator agents or subagents. Subagents are used for parallel SQL processing in partitioned databases and on SMP machines.

For partitioned database environments and environments with intra-partition parallelism enabled, each partition (that is, each database server or node) has its own pool of agents from which subagents are drawn. Because of this pool, subagents do not have to be created and destroyed each time one is needed or has finished its work. The subagents can remain as associated agents in the pool and be used by the database manager for new requests from the application they are associated with.

The Db2Agents test monitors how effectively the agent pool has been utilized.

Purpose	Monitors how effectively the agent pool has been utilized		
Target of the test	A DB2 database server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test		
Outputs of the test	One set of results for every DB2 database server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Total agents: Indicates the total number of agents currently registered in the database manager instance that is being monitored (Sum of coordinator agents and subagents).	Number	This measure can be used to evaluate the setting for MAXAGENTS configuration parameter.

	Idle agents : Indicates the percentage of agents in the agent pool that is currently unassigned to an application and are, therefore, "idle".	Percent	Having idle agents available to service requests for agents can improve performance. So you can use this measure to help set the NUM_POOLAGENTS configuration parameter.
	Agents waiting on token: Indicates the percentage of agents waiting for a token so they can execute a transaction in the database manager.	Percent	You can use this element to help evaluate your setting for the MAXCAGENTS configuration parameter. Each application has a dedicated coordinator agent to process database requests within the database manager. Each agent has to get a token before it can execute a transaction. The maximum number of agents that can execute database manager transactions is limited by the configuration parameter MAXCAGENTS.
	Agents creation ratio: Indicates the ratio of number of agents assigned directly from agent pool to the total number of agents used to service requests.	Percent	A high percentage indicates the effectiveness of the agent pool. A consistent low value indicates that the number of agents in the agent pool are not adequate to service requests. You might want to consider increasing the NUM_POOLAGENTS setting in this case.
	Stolen Agents: Indicates the number of times that agents are stolen from an application. Agents are stolen when an idle agent associated with an application is reassigned to work on a different application.	Number	If this value is high, consider increasing the NUM_POOLAGENTS configuration parameter.

4.1.1.2 Db2 Connections Test

The IBMDb2Connections test reports key statistics pertaining to the local and remote connections to the DB2 database manager.

Purpose	Reports key statistics pertaining to the local and remote connections to the DB2 database manager
Target of the test	A DB2 database server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the DB2 server
3. PORT – The port number through which the DB2 server communicates. The default port is 50000.
4. USER - Specify the name of the user who has any of the following privileges to the specified DATABASE: SYSADM or SYSCTRL or SYSMAINT or SYSMON.
5. PASSWORD - Enter the password of the specified USER in the PASSWORD text box.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. DATABASE - Specify the name of the database on the monitored DB2 server to be used by this test.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every DB2 database server being monitored</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total connections:
Indicates the total number of local and remote connections that are currently present in the database manager.</td><td>Number</td><td></td></tr>
<tr><td>Local connections:
Indicates the number of local applications that are currently connected to a database within the database manager instance being monitored.</td><td>Number</td><td>This number can help you determine the level of concurrent processing occurring in the database manager. This number only includes applications that were initiated from the same instance as the database manager. The applications are connected, but may or may not be executing a unit of work in the database. When used in conjunction with the Remote connections measurement, this measure can help you adjust the setting of the MAXAGENTS configuration parameter.</td></tr>
<tr><td>Remote connections:
Indicates the percentage of agents waiting for a token so they can execute a transaction in the database manager.</td><td>Percent</td><td>This number does not include applications that were initiated from the same instance as the database manager. When used in conjunction with the Local connections measure, this measure can help you adjust the setting of the MAX_COORDAGENTS configuration parameter.</td></tr>
</table>

	Local connections in exec: Indicates the number of local applications that are currently connected to a database within the database manager instance being monitored and are currently processing a unit of work.	Number	This number can help you determine the level of concurrent processing occurring in the database manager. This number only includes applications that were initiated from the same instance as the database manager. When used in conjunction with the *Remote connections in exec* measure, this measure can help you adjust the setting of the MAXCAGENTS configuration parameter.
	Remote connections in exec: Indicates the number of remote applications that are currently connected to a database and are currently processing a unit of work within the database manager instance being monitored.	Number	This number can help you determine the level of concurrent processing occurring on the database manager. This number does not include applications that were initiated from the same instance as the database manager. When used in conjunction with the *Local connections in exec* measure, this metric can help you adjust the setting of the MAXCAGENTS configuration parameter.

4.1.1.3 Db2 Database Manager Test

The database manager includes the database engine and the facilities to access data, such as the command line processor and the application interfaces. This test reports key statistics pertaining to the health of the DB2 database manager.

Purpose	Reports key statistics pertaining to the health of the DB2 database manager
Target of the test	A DB2 database server
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test
Outputs of the test	One set of results for every DB2 database server being monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Sorts post threshold: Indicates the number of sorts per second that have requested heaps after the sort heap threshold has been exceeded.	Sorts/Sec	Under normal conditions, the database manager will allocate sort heap using the value specified by the SORTHEAP configuration parameter. If the amount of memory allocated to sort heaps exceeds the sort heap threshold (SHEAPTHRES configuration parameter), the database manager will allocate sort heap using a value less than that specified by the SORTHEAP configuration parameter. Each active sort on the system allocates memory, which may result in sorting taking up too much of the system memory available. Sorts that start after the sort heap threshold has been reached may not receive an optimum amount of memory to execute, but, as a result, the entire system may benefit. By modifying the sort heap threshold and sort heap size configuration parameters, sort operation performance and overall system performance can be improved. If this element's value is high, you can: t. Increase the sort heap threshold (SHEAPTHRES), or, u. Adjust applications to use fewer or smaller sorts via SQL query changes.
	Piped Sorts Requested: A sort is classified as piped sort if the sorted information can return directly without requiring a temporary table to store a final, sorted list of data. This measure reports the number of piped sorts that have been requested per second..	Sorts/Sec	Piped sorts may reduce disk I/O. Allowing more piped sorts therefore, can improve the performance of sort operations and possibly the performance of the overall system.

	Piped Sorts Rejected: Indicates the percentage of piped sort requests that have been rejected.	Percent	When the number of rejected piped sorts are high, you can improve sort performance by adjusting one or both of the following configuration parameters: v. SORTHEAP w. SHEAPTHRES If piped sorts are being rejected, you might consider decreasing your sort heap or increasing your sort heap threshold. You should be aware of the possible implications of either of these options. If you increase the sort heap threshold, then there is the possibility that more memory will remain allocated for sorting. This could cause the paging of memory to disk. If you decrease the sort heap, you might require an extra merge phase that could slow down the sort.
	Hash Join Post Threshold: Indicates the total number of times that a hash join heap request was limited due to concurrent use of shared or private sort heap space.	Hits/Sec	If this value is large, the sort heap threshold should be increased.

4.1.2 The Memory Structures Layer

The tests mapped to the **Memory Structures** layer (see Figure 4.3), report critical statistics that reveal:

- How efficiently the locking and sorting activities occur on the monitored DB2 database
- How well the buffer pools are managed
- The level of I/O activity on the DB2 database

Figure 4.3: The tests associated with the Memory Structures layer

4.1.2.1 Db2 Locks Test

Typically, locking activity is governed by the following factors:

- Concurrency and granularity
- Lock compatibility
- Lock conversion
- Lock escalation
- Lock waits and timeouts
- Deadlocks

In the event of an application slowdown, the measures reported by the Db2Locks test enable administrators to accurately determine whether/not any of the above-mentioned factors have adversely impacted application performance, and if so, to what extent.

Purpose	Monitors the locking activity on the DB2 database server and reports critical statistics that will enable administrators to accurately determine what has caused a significant dip in application performance		
Target of the test	A DB2 database server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test		
Outputs of the test	One set of results for every database on the DB2 database server that is currently active		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Locks Held: Indicates the total number of locks currently held by all applications in the database.	Number	

	Locks Escalated: Indicates the number of times every second that locks have been escalated from several row locks to a table lock. A lock is escalated when the total number of locks held by an application reaches the maximum amount of lock list space available to the application, or the lock list space consumed by all applications is approaching the total lock list space. The amount of lock list space available is determined by the maxlocks and locklist configuration parameters. When an application reaches the maximum number of locks allowed and there are no more locks to escalate, it will then use space in the lock list allocated for other applications. When the entire lock list is full, an error occurs. This data item includes a count of all lock escalations, including exclusive lock escalations.	Escalations/Sec	There are several possible causes for excessive lock escalations: x. The lock list size (locklist) may be too small for the number of concurrent applications y. The percent of the lock list usable by each application (maxlocks) may be too small z. One or more applications may be using an excessive number of locks. aa. To resolve these problems, you may be able to: bb. Increase the locklist configuration parameter value. cc. Increase the maxlocks configuration parameter value. Identify the applications with large numbers of locks or those that are holding too much of the lock list. These applications can also cause lock escalations in other applications by using too large a portion of the lock list. These applications may need to resort to using table locks instead of row locks, although table locks may cause an increase in lock_waits and lock_wait_time.

	Exclusive Lock Escalations: Indicates the number of times per second that locks have been escalated from several row locks to one exclusive table lock, or the number of times (per second) an exclusive lock on a row caused the table lock to become an exclusive lock.	Escalations/Sec	Other applications cannot access data held by an exclusive lock; therefore it is important to track exclusive locks since they can impact the concurrency of your data. A lock is escalated when the total number of locks held by an application reaches the maximum amount of lock list space available to the application. The amount of lock list space available is determined by the locklist and maxlocks configuration parameters. When an application reaches the maximum number of locks allowed and there are no more locks to escalate, it will then use space in the lock list allocated for other applications. When the entire lock list is full, an error occurs. See *Lock escalations* for possible causes and resolutions to excessive exclusive lock escalations. An application may be using exclusive locks when share locks are sufficient. Although share locks may not reduce the total number of lock escalations share lock escalations may be preferable to exclusive lock escalations.
	Locks Timedout: Indicates the number of times that a request to lock an object timed-out instead of being granted.	Timeouts/Sec	This measurement can help you adjust the setting for the locktimeout database configuration parameter. If the number of lock time-outs becomes excessive when compared to normal operating levels, you may have an application that is holding locks for long durations. In this case, this element may indicate that you should analyze some of the other lock and deadlock monitor elements to determine if you have an application problem. You could also have too few lock time-outs if your **locktimeout** database configuration parameter is set too high. In this case, your applications may wait excessively to obtain a lock.
	Lock Waits: Indicates the total number of times per second that applications or connections waited for locks.	Waits/Sec	If the value is consistently high, find t the applications or connections causing lock waits and fine tune the appropriate SQL queries.
	Average Lock Wait Time: Indicates the average time that all the applications were waiting for a lock.	Secs	If the average lock wait time is high, you should look for applications that hold many locks, or have lock escalations, with a focus on tuning your applications to improve concurrency, if appropriate.

	Percent of Application in Lock Wait: Indicates the percentage of applications waiting for the release of lock.	Percent	If this value is high, the applications may have concurrency problems, and you should identify applications that are holding locks or exclusive locks for long periods of time.
	Deadlocks: Indicates the total number of deadlocks that have been detected per second.	Deadlocks/Sec	This element can indicate that applications are experiencing contention problems. These problems could be caused by the following situations: dd. Lock escalations are occurring for the database ee. An application may be locking tables explicitly when system-generated row locks may be sufficient. ff. An application may be using an inappropriate isolation level when binding gg. Catalog tables are locked for repeatable read hh. Applications are getting the same locks in different orders, resulting in deadlock You may be able to resolve the problem by determining in which applications (or application processes) the deadlocks are occurring. You may then be able to modify the application to enable it to execute concurrently. Some applications, however, may not be capable of running concurrently.

4.1.2.2 Db2 Pools Test

A buffer pool is an area of memory into which database pages are read, modified, and held during processing.

Buffer pools improve database performance. If a needed page of data is already in the buffer pool, that page is accessed faster than if that page had to be read directly from disk. The database manager has agents whose tasks are to retrieve data pages from disk and place them in the buffer pool (prefetchers), and to write modified data pages from the buffer pool back to disk (page cleaners).

The reading and writing of data pages to and from disk is called disk input/output (I/O). Avoiding the wait associated with disk I/O is the primary way to improve the performance of the database. How you create the buffer pool, and configure the database manager and the agents associated with the buffer pool, controls the performance of the database. Through SQL and configuration parameters, you can control the size of the buffer pool, the number of prefetchers and page cleaners that move data pages into and out of the buffer pool, the size of the data pages, and

the number of data pages that can be moved at one time.

The statistics reported by the Db2Pools test help administrators analyze the usage of the buffer pools, and provides them with useful pointers to fine-tune the configuration of the buffer pools.

Purpose	Monitors the locking activity on the DB2 database server and reports critical statistics that will enable administrators to accurately determine what has caused a significant dip in application performance		
Target of the test	A DB2 database server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test		
Outputs of the test	One set of results for every database on the DB2 database server that is currently active		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Buffer pool hit ratio: Indicates the percentage of the requested pages that is readily available in the memory without doing disk I/O.	Percent	This measure is an overall indicator of how well the buffer pool is performing. Lower values indicate that more physical I/O is happening than logical. Since physical I/O costs more, maintaing higher buffer hit ratio is desired. Typically a hit ratio over 90% shows that the buffer pool is performing well. If this measure is consistently low, increase the size of the buffer pool by increasing BUFFPAGE configuration value.
	Buffer Pool Hit Ration (Data): Indicates the the percentage of the requested data pages that is readily available in the memory without doing disk I/O.	Percent	This measure is an indicator of how well the buffer pool is performing for the data page requests. Lower values indicate that more physical I/O is happening than logical. Since physical I/O costs more, maintaing higher buffer hit ratio is desired. Typically a hit ratio over 90% shows that the buffer pool is performing well. If this measure is consistently low, increase the size of the buffer pool by increasing BUFFPAGE configuration value.

	Buffer Pool Hit Ratio (Index): Indicates the percentage of the requested index pages that is readily available in the memory without doing disk I/O.	Percent	This measure is an overall indicator of how well the buffer pool is performing. Lower values indicate that more physical I/O is happening than logical. Since physical I/O costs more, maintaing higher buffer hit ratio is desired. Typically a hit ratio over 90% shows that the buffer pool is performing well. If this ratio is really low and the overall ratio is relatively high, then break the index(s) out into their own tablespace/buffer pool.
	Pre Fetch Ratio: The ratio of asynchronous reads to synchronous reads. The value indicates how effectively DB2 database manager is populating the buffer pools through the use of prefetchers.	Percent	High value indicates more asynchronous I/O is happening than synchronous. The value can be used to tune the **num_ioservers** configuration parameter.
	Percent Log Cleans: Indicates the percentage of times a page cleaner was invoked because the logging space used had reached a predefined criterion for the database.	Percent	If this value is high (say > 40%), this could mean that page cleaners are constantly being utilized to clean the log and aren't available for other page cleaning activities, hampering performance. On the other hand, if the value is low, (say < 10%) then the page cleaners aren't being triggered as often for this activity. this means that they would be more available for the other types of page cleaning activities, which is great for buffer pool performance.
	Percent Dirty Page Cleans: Indicates the percentage of times a page cleaner was invoked because a buffer pool had reached the dirty page threshold criterion for the database.	Percent	The threshold is set by the chngpgs_thresh configuration parameter. It is a percentage applied to the buffer pool size. When the number of dirty pages in the pool exceeds this value, the cleaners are triggered. If this value is set too low, pages might be written out too early, requiring them to be read back in. If set too high, then too many pages may accumulate, requiring users to write out pages synchronously.

	Percent Victim Cleans: Indicates the percentage of times the page cleaner(s) were triggered to oust a victim page from the buffer pool. A victim page is a clean or dirty page in the buffer pool that is removed simply because DB2 needs to make room for incoming pages. If a victim page is a dirty page then the information must be written out to disk. Any page that is removed will most likely cause more physical I/O to occur in order to retrieve it again at later time when DB2 is ready to use it.	Percent	If the ratio is higher then the above two then that is typically a good indicator that the buffer pool needs to be larger since there never seems to be enough room for new pages to be brought in. This could also be a sign that dirty pages are staying in the buffer pool too long which could mean that the changed pages threshold (CHNGPGS_THRESH) is set too high. Even the SOFTMAX parameter could be set too high and too much of the changed pages that are logged are not getting flushed out to make way for new pages. If this ratio is low, it may indicate that you have defined too many page cleaners. If your chngpgs_thresh is set too low, you may be writing out pages that you will dirty later. Aggressive cleaning defeats one purpose of the buffer pool, that is to defer writing to the last possible moment.
	Catalog Cache Hit Ratio: Indicates the percentage of time the requested information for table descriptor or authorization was readily availble in catalog cache without requiring to perform disk I/O.	Percent	The catalog cache is referenced whenever a table, view, or alias name is processed during the compilation of an SQL statement. If the ratio is greater than 80%, then the catalog cache is performing well. A smaller value indicates that the catalog cache size should be increased by tuning the parameter CATALOGCACHE_SZ in the database configuration. The value may be low immediately following the first connection to the database. The execution of Data Definition Language (DDL) SQL statements involving a table,view, or alias will evict the table descriptor information for that object from the catalog cache causing it to be re-inserted on the next reference. Therefore, the heavy use of DDLs may also increase the value of the measure.
	Package Cache Hit Ratio: The package and section information required for the execution of dynamic and static SQL statements are placed in the package cache as required. This information is required whenever a dynamic or static statement is being executed. The ratio indicates the effectiveness of package cache hit ratio.	Percent	If the hit ratio is high (more than 80%), the cache is performing well. A smaller ratio may indicate that the package cache size (pckcachesz) should be increased.

4.1.2.3 Db2 Sorting Test

Sorting represents organizing the rows in a table into the order of one or more of its columns, optionally eliminating duplicate entries. Sorting is required when:

- No index exists to satisfy a requested ordering (for example a SELECT statement that uses the ORDER BY clause).
- An index exists but sorting would be more efficient than using the index
- An index is created.
- An index is dropped, which causes index page numbers to be sorted.

Because queries often require sorted or grouped results, sorting is often required, and the proper configuration of the sort heap areas is crucial to good query performance. Using the Db2Sort test, administrators can figure out whether/not the sort heap allocations are sufficient to facilitate efficient sorting.

Purpose	Helps administrators figure out whether/not the sort heap areas are adequately configured to facilitate efficient sorting		
Target of the test	A DB2 database server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test		
Outputs of the test	One set of results for every database on the DB2 database server that is currently active		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Sort Heap Allocated:** Indicates the sum of sort heap space allocated for all sorts in all active databases in the database manager.	Pages	Normal memory estimates do not include sort heap space. If excessive sorting is occurring, the extra memory used for the sort heap should be added to the base memory requirements for running the database manager. Generally, the larger the sort heap, the more efficient the sort. Appropriate use of indexes can reduce the amount of sorting required. You may use the information returned at the database manager level to help you tune the SHEAPTHRES configuration parameter. If the element value is greater than or equal to SHEAPTHRES, it means that the sorts are not getting the full sort heap as defined by the SORTHEAP parameter.
	Average Sort Heap Space Used: Indicates the average sort heap space used by each sort.	Pages	If the SORTHEAP configuration parameter is substantially larger than the average sort heap used, you may be able to lower the value of this parameter.
	Sort Rate: Indicates the number of sort operations performed during the last measurement period.	Sorts / Sec	
	Percent Sort Overflow: Indicates the percentage of sorts that had to overflow to disk.	Percent	Sort overflows are sorts that ran out of sort heap and may have required disk space for temporary storage. When a sort overflows, additional overhead will be incurred because the sort will require a merge phase and can potentially require more I/O, if data needs to be written to disk. If this percentage is high, you may want to adjust the database configuration by increasing the value of sortheap.
	Average Sort Time: Indicates the average sort time for all sorts performed by all applications connected to a particular database.	Secs	A high value indicates the poor performance of sorting operations. Identify the statements that spend lot of time sorting. You may want to reduce the average sort time for these statements by increasing the sortheap parameter.

4.1.2.4 Db2 Direct I/O Test

This test monitors the I/O activity on the currently active DB2 database.

Purpose	Monitors the I/O activity on the currently active DB2 database
Target of the test	A DB2 database server

<table>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the DB2 server
3. PORT – The port number through which the DB2 server communicates. The default port is 50000.
4. USER - Specify the name of the user who has any of the following privileges to the specified DATABASE: SYSADM or SYSCTRL or SYSMAINT or SYSMON.
5. PASSWORD - Enter the password of the specified USER in the PASSWORD text box.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. DATABASE - Specify the name of the database on the monitored DB2 server to be used by this test</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every database on the DB2 database server that is currently active</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Direct Read Rate:
Indicates the total number of direct reads by the application per second. In other words, it is the number of read operations that do not use the buffer pool.</td><td>Reads/Sec</td><td>Direct reads are performed in units, the smallest being a 512-byte sector. They are used when:
• Reading LONG VARCHAR columns
• Reading LOB (large object) columns
• Performing a backup
A high value over a period of time may be indicative of a performance bottleneck.</td></tr>
<tr><td>Direct Write Rate:
Indicates the total number of direct writes by the application per second. In other words, it is the number of write operations that do not use the buffer pool.</td><td>Writes/Sec</td><td>Direct writes are performed in units, the smallest being a 512-byte sector. They are used when:
• Writing LONG VARCHAR columns
• Writing LOB (large object) columns
• Performing a restore
• Performing a load
A high value over a period of time may be indicative of a performance bottleneck.</td></tr>
<tr><td></td><td>Buffer Pool IO Rate:
Indicates the rate at which the buffer pool I/O operations are being done in the database.</td><td>IOs/Sec</td><td>In conjunction with the hit ratio statistics, and the characteristics of the applications executing, the I/O load may require adjustment of BUFFPAGE, or applications may require further tuning.</td></tr>
</table>

4.1.3 The Db2 Service Layer

Besides revealing the availability and responsiveness of the DB2 server, the tests associated with the **Db2 Service** layer indicate the level of SQL activity on the server, and the number and type of transactions that occur on the server. These measurements together serve as effective indicators of the processing ability of the DB2 server.

Figure 4.4: The tests associated with the Db2 Service layer

4.1.3.1 Db2 Activity Test

This test measures the level of SQL activity on the DB2 database server, and reveals how well the server processes SQL queries.

Purpose	Monitors the I/O activity on the currently active DB2 database		
Target of the test	A DB2 database server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test		
Outputs of the test	One set of results for every database on the DB2 database server that is currently active		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Dynamic SQL Rate:** Indicates the rate of dynamic SQL statements that were attempted.	Stmts/Sec	This is an indication of throughput of the system during the monitoring period. A high value of dynamic SQLs and low value of failed SQLs indicate good throughput.
	Select SQL Rate: Indicates the rate at which SELECT SQL statements were executed during the last measurement period.	Stmts/Sec	This measure can be used to determine the level of database activity.
	UID SQL Rate: Indicates the rate at which update/delete/insert statements were issued during the last measurement period.	Stmts/Sec	This measure can be used to determine the level of database activity.
	Failed SQL Rate: Indicates the rate at which SQL statements failed.	Stmts/Sec	A relatively high value indicates a problem. Failed SQL statements waste system resources. Hence, the value of this measure should be very low.
	Percent Failed SQL: Indicates the percentage of SQL statements that failed during the interval. This value includes all SQL statements that received a negative SQLCODE	Percent	
	Percent DDL SQL: Denotes the percentage of SQL statements that were DDL(Data Definition Language) during the last measurement period.	Percent	This value should normally be low.
	Percent UID SQL: Indicates the percentage of update/insert/delete statements executed during the last measurement period.	Percent	This measure can be used to determine the level of database activity.

4.1.3.2 Db2 Transaction Test

This test tracks various statistics pertaining to the transactions executing on a DB2 database.

Purpose	Tracks various statistics pertaining to the transactions executing on a DB2 database

Target of the test	A DB2 database server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test		
Outputs of the test	One set of results for every database on the DB2 database server that is currently active		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Commit Rate: Indicates the transaction throughput. This measure is the sum of the committed statements attempted and internal commits (total number of commits initiated internally by the database manager) per second.	Commits/Sec	A decrease in this measure during the monitoring period may indicate that the applications are not doing frequent commits. This may lead to problems with logging and data concurrency. The cause has to be probed in the application.
	Rollback Rate: Indicates the rate of unit of work rollbacks.	Rollbacks/Sec	A high rollback rate is an indicator of bad performance, since work performed up to the rollback point is wasted. The cause of the rollbacks has to be probed in the application.
	Transaction Rate: Indicates the rate of commits and rollbacks for the application using the DB2 Connect gateway.	Trans/Sec	A high transaction rate with high rollback rate indicates bad performance.

4.1.3.3 Db2 Service Test

This test monitors the availability and response time of an IBM DB2 server.

Purpose	Monitors the availability and response time of an IBM DB2 server

Target of the test	A DB2 database server		
Agent deploying the test	An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the target DB2 server is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the target DB2 server is listening.		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test 8. **QUERY** - The test emulates a user executing a query on the specified **DATABASE**, and thus determines the availability and responsiveness of the database server. In the **QUERY** text box, specify the select query to execute.		
Outputs of the test	One set of results for the DB2 database server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Availability: Indicates the availability of the server.	Percent	The availability is 100% when the server is responding to a request and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the database server, or if the server has not been started.
	Response Time: Indicates the time taken by the database to respond to a user query.	Secs	A sudden increase in response time is indicative of a bottleneck at the database server.

4.1.3.4 DB2 Tablespaces Test

This test auto-discovers the tablespaces on an IBM DB2 server, and monitors the space usage of each tablespace.

Purpose	Auto-discovers the tablespaces on an IBM DB2 server, and monitors the space usage of each tablespace
Target of the test	A DB2 database server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test 8. **QUERY** - The test emulates a user executing a query on the specified **DATABASE**, and thus determines the availability and responsiveness of the database server. In the **QUERY** text box, specify the select query to execute.		
Outputs of the test	One set of results for each tablespace on the DB2 database server being monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Tablespace type:** Indicates the tablespace type.		If the target tablespace is a System-managed tablespace, then this measure will report the value *System Managed Storage*. On the other hand, if the tablespace is a database managed tablespace, then the value of this measure will be *Database Managed Storage*. The SMS (System Managed Space) tablespaces allow the operating system to allocate and manage the space where the table data resides. Once the initial create has been completed, you cannot add or delete containers to an SMS tablespace. The data in the table spaces is striped by extent across all the containers in the system. An **extent** is a group of consecutive pages defined to the database. The file extension denotes the type of the data stored in the file. To distribute the data evenly across all containers in the table space, the starting extents for tables are placed in round-robin fashion across all containers. Such distribution of extents is particularly important if the database contains many small tables. In a DMS (Database Managed Space) table space, the database manager controls the storage space. The storage model consists of a limited number of devices or files whose space is managed by DB2 Database for Linux, UNIX, and Windows. The database administrator decides which devices and files to use, and DB2® manages the space on those devices and files. The table space is essentially an implementation of a special purpose file system designed to best meet the needs of the database manager. DMS table spaces are different from SMS table spaces in that space for DMS table spaces is allocated when the table space is created. For SMS table spaces, space is allocated as needed - i.e., on demand.

<table>
<tr><td></td><td></td><td></td><td>The numeric values that correspond to the tablespace types reported by this measure are as follows:<table><tr><th>Numeric Value</th><th>Tablespace Type</th></tr><tr><td>0</td><td>Database Managed Storage</td></tr><tr><td>1</td><td>System Managed Storage</td></tr></table>Note:
By default, this measure reports the Types displayed in the table above as its value. In the graph of the measure however, the types are represented using their corresponding numeric equivalents - i.e., 0 and 1.</td></tr>
<tr><td></td><td>Usable space:
Indicates the amount of space allocated to this tablespace.</td><td>MB</td><td>This measure is available only for DMS tablespaces.</td></tr>
<tr><td></td><td>Used space:
Indicates the space used in this tablespace.</td><td>MB</td><td>Ideally, the value of this measure should be low. A very high value or a consistent increase in this value could indicate a potential contention for space.</td></tr>
<tr><td></td><td>Free space:
Indicates the free space in this tablespace.</td><td>MB</td><td>This measure is available only for DMS tablespaces. This is because, for SMS tablespaces, space is allocated on demand, and deallocated when not required; this implies that SMS tablespaces will at no time have any unused space.
For DMS tablespaces, a high value is desired for this measure. A very low value or a gradual decrease in this value could be a cause for concern, as it indicates a slow, but steady space erosion.</td></tr>
<tr><td></td><td>Availability:
Indicates the percentage of free space in this tablespace.</td><td>Percent</td><td>As free space value is applicable only for DMS tablespaces, this will be available only for DMS tablespaces. The test will not report this measure for SMS tablepsaces.
For DMS tablespaces, a high value is desired for this measure. A very low value or a gradual decrease in this value could be a cause for concern, as it indicates a slow, but steady space erosion.</td></tr>
</table>

4.1.4 DB2 SQL Workload Test

Nothing can degrade the performaB server like a resource-hungry or a long-running query! When such queries execute on the server, they either hog almost all the available CPU, memory, and disk resources or keep the resources locked for long time periods, thus leaving little to no resources for carrying out other critical database operations. This can significantly slowdown the database server and adversely impact user experience with the server. To ensure peak performance of the DB2 UDB server at all times, such queries should be rapidly identified and quickly optimized to minimize resource usage. This is where the **DB2 SQL Workload** test helps. At configured intervals, this test compares the usage levels and execution times of all queries that started running on the server in the last measurement period and identifies a 'top query' in each of the following categories - CPU usage, memory usage, disk activity, and execution time. The test then reports the resource usage and execution time of the top queries and promptly alerts administrators if any query consumes more resources or takes more time to execute than it should. In such a scenario, administrators can use the detailed diagnosis of this test to view the inefficient queries and proceed to optimize them to enhance server performance.

Purpose	At configured intervals, this test compares the usage levels and execution times of all queries that started running on the server in the last measurement period and identifies a 'top query' in each of the following categories - CPU usage, memory usage, disk activity, and execution time. The test then reports the resource usage and execution time of the top queries and promptly alerts administrators if any query consumes more resources or takes more time to execute than it should. In such a scenario, administrators can use the detailed diagnosis of this test to view the inefficient queries and proceed to optimize them to enhance server performance.
Target of the test	A DB2 UDB server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the DB2 server
3. PORT – The port number through which the DB2 server communicates. The default port is 50000.
4. USER - Specify the name of the user who has any of the following privileges to the specified DATABASE: SYSADM or SYSCTRL or SYSMAINT or SYSMON.
5. PASSWORD - Enter the password of the specified USER in the PASSWORD text box.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. DATABASE - Specify the name of the database on the monitored DB2 server to be used by this test
8. QUERY - The test emulates a user executing a query on the specified DATABASE, and thus determines the availability and responsiveness of the database server. In the QUERY text box, specify the select query to execute.
9. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the server monitored.</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Top most query physical reads:
Indicates the number of physical disk reads performed by the top query per execution.</td><td>Reads/execution</td><td>If the value of this measure is abnormally high, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries generating maximum physical disk activity. From this, you can identify the top query in terms of number of physical disk reads. You may then want to optimize the query to reduce the disk reads.</td></tr>
<tr><td>Top most buffer gets:
Indicates the number of memory buffers used by the top query per execution.</td><td>Memorybuffer gets/execution</td><td>If the value of this measure is abnormally high, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries consuming memory excessively. From this, you can easily pick that query which is consuming the maximum memory. You may then want to optimize the query to minimize memory usage.</td></tr>
</table>

	Top most query CPU time: Indicates the duration for which each execution of the top query was hogging the CPU resources.	Secs/execution	If the value of this measure is over 30 seconds, you can use the detailed diagnosis of this measure to the top-5 (by default) queries hogging the CPU resources. From this, you can easily pick that query which is consuming the maximum CPU. You may then want to optimize the query to minimize CPU usage.
	Top most query elapsed time: Indicates the running time of each execution of the top query.	Secs/execution	If the value of this measure crosses 10 seconds, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries that are taking too long to execute. . From this, you can easily pick that query with the maximum execution time. You may then want to optimize the query to minimize execution time.

4.2 Monitoring DB2 Server Version 6.0/7.x

Figure 4.5 depicts the *DB2 UDB – 6/7.x* monitoring model that eG Enterprise prescribes for versions 6.0/7.x of the DB2 UDB server. This model has been deprecated, and is retained only to ensure backward compatability with older, less-used DB2 versions.

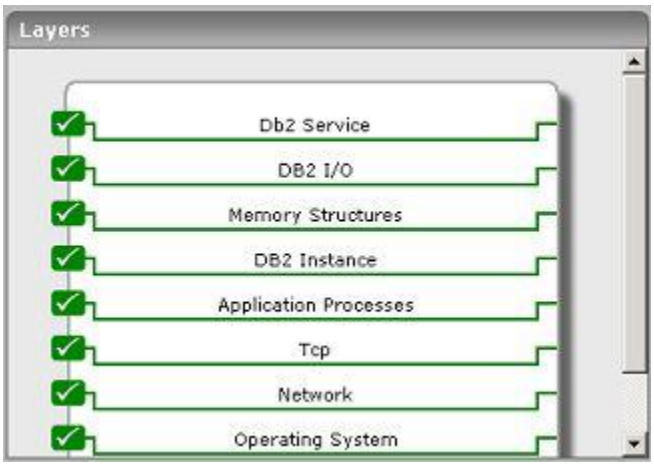


Figure 4.5: Layer model of the DB2 server version 6.0/7.x

The tests mapped to each of the layers of Figure 4.5 report critical statistics pertaining to the internal health of the DB2 server ver. 6.0/7.x. However, unlike the monitoring model discussed in Section 3.1, this model requires that an eG agent be physically installed on the DB2 server for monitoring its performance. In other words, eG Enterprise cannot monitor a DB2 server ver. 6.0/7.x in an "agentless" manner.

The sections to come discuss each of the layers of Figure 4.5, elaborately.

4.2.1 The DB2 Instance Layer

Besides monitoring the availability of DB2 UDB databases, critical instance level measurements relating to the usage and responsiveness of each instance can be obtained on an on-going basis by an eG agent. This layer tracks the

instance level measures of the DB2 UDB database server with the aid of the Db2Instance test (see Figure 4.6). In Figure 4.6, the Db2Instance test is used to monitor a specific instance called db2inst1 of the target DB2 database.

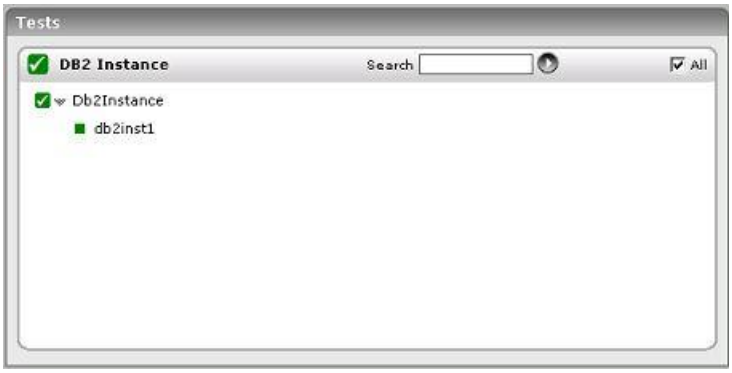


Figure 4.6: Tests mapping to the DB2 Instance layer

4.2.1.1 Db2 Instances Test

This test, executed by an internal agent, tracks various statistics at the instance level of a DB2 UDB database. The details of the test are provided below:

Purpose	To measure statistics at the instance level of a DB2 database		
Target of the test	A DB2 database server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000.		
Outputs of the test	One set of results for every database being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Waiting agents: Indicates the percentage of agents waiting to execute a unit of work.	Percent	It shows the percentage of agents that are waiting to execute (that is, that are asleep). Use this measure to adjust the *MAXCAGENTS* parameter setting for the database. If this value is high, you may consider increasing *MAXCAGENTS.*

	Piped sorts accepted : Indicates the percentage of piped sorts accepted during the interval.	Percent	Piped sorts reduce disk I/O and tend to be more efficient. A high percentage of piped sorts accepted indicate that efficient sorts are being performed. A low percentage indicates that sort heap and sort heap threshold may have to be adjusted. If this percentage is low, then increase *SHEAPTHRES* and possibly *SORTHEAP*.
	Piped sort rejected: Indicates the percentage of piped sort requests that were rejected.	Percent	Piped sorts reduce disk I/O and tend to be more efficient. A low percentage of piped sorts rejected indicate that efficient sorts are being performed. A high percentage of piped sorts rejected indicate that *sort heap* and *sort heap threshold* may have to be adjusted. If this percentage is high, then increase *SHEAPTHRES* and possibly *SORTHEAP*.
	Agents registered: Indicates the number of agents registered in the database manager instance that is being monitored (coordinator agents and subagents).	Number	This measure can be used to evaluate your setting for the *MAXAGENTS* configuration parameter.
	Agents from empty pool : Indicates how often an agent must be created because the pool is empty.	Percent	A high value may indicate that the *NUM_POOLAGENTS* configuration parameter should be increased. A low value suggests that *NUM_POOLAGENTS* is set too high, and that some of the agents in the pool are rarely used and are wasting system resources. A high percentage can also indicate that the overall workload for this node is too high. The workload can be adjusted by lowering the maximum number of coordinating agents specified by the *MAXCAGENTS* configuration parameter, or by redistributing data among the nodes.

4.2.2 The Memory Structures Layer

Next is the **Memory Structures** layer that tracks the health of the memory, lock, and buffer structures of the database server using the DB2LockAndDeadlock test and DB2BufferPool test (see Figure 4.7). Key performance metrics such as lock activity, buffer pool hit ratios, read and write rates to the database, average sorting time, rollback rates, etc. are being reported per database.

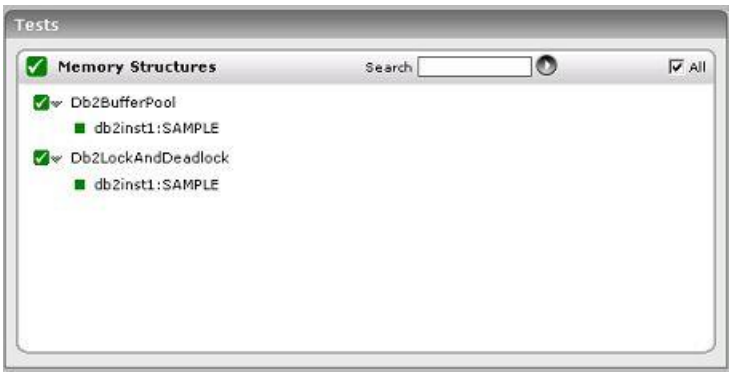


Figure 4.7: Tests mapping to the Memory Structures layer

4.2.2.1 Db2 Locks And Deadlocks Test

This test, executed by an internal agent, tracks various statistics pertaining to the locks and deadlocks in a DB2 database. The details of the test are provided below:

Purpose	To measure statistics pertaining to the locks and deadlocks in a database		
Target of the test	A DB2 database server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000.		
Outputs of the test	One set of results for every database being monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Deadlocks:** Indicates the total number of deadlocks that have been detected.	Number	A high value is indicative of contention problems. These problems could be caused by the following reasons: ➢ Lock escalations are occurring for the database ➢ An application may be locking tables explicitly when system-generated row locks may be sufficient ➢ An application may be using an inappropriate isolation level when binding ➢ Catalog tables are locked for repeatable read ➢ Applications are getting the same locks in different orders, resulting in deadlock. To resolve the problem, first determine the applications (or application processes) in which the deadlocks are occurring. Then, modify the application to enable it to execute concurrently. Some applications, however, may not be capable of running concurrently.
	Exclusive lock escalations: Indicates the number of times that locks have been escalated from several rowlocks to one exclusive table lock, or the number of times an exclusive lock on a row caused the table lock to become an exclusive lock. Other applications cannot access data held by an exclusive lock; therefore it is important to track exclusive locks since they can impact the concurrency of your data.	Number	A lock is escalated when the total number of locks held by an application reaches the maximum amount of lock list space available to the application. The amount of lock list space available is determined by the *LOCKLIST* and *MAXLOCKS* configuration parameters. A high value of this measure indicates that an application may be using exclusive locks when share locks are sufficient.

	Locks held: Indicates the total number of locks that have been currently held by all applications in/using the database.	Number	1. This measure provides summary information about locking. A high value indicates that one or more of the applications needs to be tuned to improve performance. 2. You can also compare the value of this measure with the results of the following formula to determine the number of additional locks that may be requested. This comparison can help you determine if the configuration parameters need adjusting or your applications need tuning. (LOCKLIST * 4096 / 36) - locks held = # remaining where: LOCKLIST is the configuration parameter 4096 is the number of bytes in one 4K page 36 is the number of bytes required for each lock.
	Lock timeouts: When a unit of work exceeds the maximum allowable amount of time, a lock timeout occurs and the unit of work isn't granted the lock it has been waiting for. This measure indicates the total number of lock timeouts during a specific interval.	Number	If the number of lock timeouts becomes excessive when compared to normal operating levels, an application may be holding locks for long durations. This requires an adjustment in the *LOCKTIMEOUT* configuration parameter. Committing can also free locks. If the *LOCKTIMEOUT* database configuration parameter is set too high, it may result in too few lock timeouts. In such a case, your applications may wait excessively to obtain a lock.
	Lock escalations : Denotes the total number of times that locks have been escalated from several row locks to a table lock.	Number	A high value signifies a problem. There are several possible causes for excessive lock escalations: ➢ The lock list size (*LOCKLIST*) may be too small for the number of concurrent applications ➢ The percent of the lock list usable by each application (*MAXLOCKS*) may be too small ➢ One or more applications may be using an excessive number of locks. To resolve these problems, ➢ Increase the *LOCKLIST* configuration parameter value. ➢ Increase the *MAXLOCKS* configuration parameter value.

	Percent of applications in lock wait: Indicates the percentage of applications waiting for the release of lock.	Percent	A high value indicates that the applications are experiencing concurrency problems. Hence, the applications that are holding locks or exclusive locks for long periods of time have to be identified.

4.2.2.2 Db2 Buffer Pools Test

This test, executed by an internal agent, tracks various statistics pertaining to the buffer pool in a DB2 UDB database. The details of the test are provided below:

Purpose	To measure statistics pertaining to the buffer pool in a DB2 UDB database		
Target of the test	A DB2 database server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000.		
Outputs of the test	One set of results for every database monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Buffer pool hit ratio: Indicates the effectiveness of the buffer pool. This indicates the percentage of the requested data blocks that is readily available in the memory without doing disk I/O.	Percent	The more a data block requested is found in the buffer pool, the better it is for performance since it avoids unnecessary disk input and output. The configuration of the buffer pool is the single most important tuning area, since most data manipulation for connected applications, excluding large objects and long field data, takes place here. If this measure is low (less than 90% for decision support, less than 50% for query-only and online transaction procession), increase the size of the buffer pool by increasing *BUFFPAGE* As a rule, the buffer pool should be as large as possible. Increase *BUFFPAGE* until you see an increase in swapping (indicated by an operating system monitor).

	Buffer pool hit ratio: Indicates the effectiveness of the buffer pool. This indicates the percentage of the requested index blocks that is readily available in the memory without doing disk I/O.	Percent	The more data found in the buffer pool, the better it is for performance since it avoids unnecessary input and output. The configuration of the buffer pool is the single most important tuning area, since most data manipulation for connected applications, excluding large objects and long field data, takes place here. If this measure is low (less than 90% for decision support, less than 50% for query-only and online transaction procession), increase the size of the buffer pool by increasing *BUFFPAGE*. As a rule, the buffer pool should be as large as possible. Increase *BUFFPAGE* until you see an increase in swapping (indicated by an operating system monitor.
	Catalog cache hit ratio: Indicates the percentage of catalog cache hit ratio. This indicates the percentage of the requested catalog blocks that is readily available in the memory without doing disk I/O.	Percent	This measure includes both successful and unsuccessful accesses to the catalog cache. The catalog cache is referenced whenever a table, view, or alias name is processed during the compilation of an SQL statement. If the ratio is greater than 80%, then the catalog cache is performing well. A smaller value indicates that the catalog cache size should be increased by tuning the parameter *CATALOGCACHE_SZ* in the database configuration. The value may be low immediately following the first connection to the database. The execution of Data Definition Language (DDL) SQL statements involving a table, view, or alias will evict the table descriptor information for that object from the catalog cache causing it to be re-inserted on the next reference. Therefore, the heavy use of DDLs may also increase the value of the measure.

4.2.3 The DB2 I/O Layer

The **DB2 I/O** layer monitors the input/output activity happening on the DB2 UDB server with the Db2 I/O test shown in Figure 4.8.

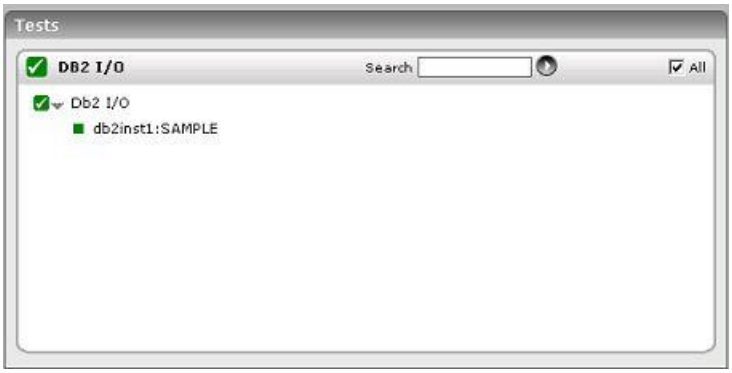


Figure 4.8: Tests mapping to the DB2 IO layer.

4.2.3.1 Db2_I/O Test

This test, executed by an internal agent, tracks various statistics pertaining to the inputs and outputs of a DB2 UDB database. The details of the test are provided below:

Purpose	To measure statistics pertaining to the inputs and outputs of a DB2 UDB database.		
Target of the test	A DB2 database server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000.		
Outputs of the test	One set of results for every database being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Direct read rate: Indicates the total number of direct reads by the application per sec. In other words, it the number of read operations that do not use the buffer pool.	Reads/Sec	Direct reads are performed in units, the smallest being a 512-byte sector. They are used when: ➢ Reading LONG VARCHAR columns ➢ Reading LOB (large object) columns ➢ Performing a backup A high value over a period of time may be indicative of a performance bottleneck.

	Direct write rate: Indicates the total number of direct writes by the application per sec. In other words, it is the number of write operations that do not use the buffer pool.	Writes/Sec	Direct writes are performed in units, the smallest being a 512-byte sector. They are used when: • Writing LONG VARCHAR columns • Writing LOB (large object) columns • Performing a restore • Performing a load. A high value over a period of time may be indicative of a performance bottleneck.
	Buffer pool I/O rate: Indicates the rate at which the buffer pool I/O operations are being done in the database.	IOs/Sec	In conjunction with the hit ratio statistics, and the characteristics of the applications executing, the I/O load may require adjustment of *BUFFPAGE*, or applications may require further tuning

4.2.4 The DB2 Service Layer

This layer tracks the overall health of the service offered by the database server to clients with the help of DB2Transaction test, DB2Sort test, and DB2SQLActivity test (see Figure 4.9). The details of all the tests mentioned above are available in the following sections.

Figure 4.9: Tests mapping to the Db2 Service layer

4.2.4.1 Db2 Transactions Test

This test, executed by an internal agent, tracks various statistics pertaining to transactions in DB2 UDB database. The details of the test are provided below:

Purpose	To measure statistics pertaining to the transactions in a DB2 UDB database
Target of the test	A DB2 database server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000.		
Outputs of the test	One set of results for every database being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Commit rate: Indicates the transaction throughput. This measure is the sum of the committed statements attempted and internal commits (total number of commits initiated internally by the database manager.) per sec.	Commits/Sec	A decrease in this measure during the monitoring period may indicate that the applications are not doing frequent commits. This may lead to problems with logging and data concurrency. The cause has to be probed in the application.
	Rollback rate: Indicates the rate of unit of work rollbacks.	Rollbacks /Sec	A high rollback rate is an indicator of bad performance, since work performed up to the rollback point is wasted. The cause of the rollbacks has to be probed in the application.
	Transaction rate: Indicates the rate of commits and rollbacks for the application using the DB2 Connect gateway.	Trans/Sec	A high transaction rate with high rollback rate indicates bad performance.

Note:

Since these measures pertain to the factors that are application dependent, no specific boundaries have been indicated for these values.

4.2.4.2 Db2 Sorts Test

This test, executed by an internal agent, tracks various statistics pertaining to the sorts in a DB2 UDB database. The details of the test are provided below:

Purpose	To measure statistics pertaining to the sorts in a DB2 UDB database
Target of the test	A DB2 database server

Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000.		
Outputs of the test	One set of results for every database being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Avg sort time: Indicates the average sort time for all sorts performed by all applications connected to a particular database.	Ms/Sort	A high value can point to a database with possible sorting problem (for example, applications are spending too much time on sorts). See the SORT screen for this database. The sort information for this database manager should also be looked at instance level. It may be necessary to increase the *SORTHEAP* configuration parameter.
	Percent Sort Overflow: Denotes the percentage of sorts that overflowed.	Percent	Sort overflows are sorts that ran out of sort heap and required disk space for temporary storage. These sorts are not efficient, and when the value of this measure is consistently high for a number of intervals, then, it may be necessary to increase the *SORTHEAP* configuration parameter.

4.2.4.3 Db2 SQL Activity Test

This test, executed by an internal agent, tracks various statistics pertaining to the SQL activities happening in a DB2 UDB database. The details of the test are provided below:

Purpose	To measure statistics pertaining to the SQL activities in a DB2 UDB database		
Target of the test	A DB2 database server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000.		
Outputs of the test	One set of results for every database being monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Dynamic SQL rate:** Indicates the number of dynamic SQL statements that were attempted.	Stmts/Sec	This is an indication of throughput of the system during the monitoring period. A high value of dynamic SQLs and low value of failed SQLs indicate good throughput.
	Select rate: Indicates the number of SELECT SQL statements that were executed.	Stmts/Sec	This measure can be used to determine the level of database activity at the application or database level. This measure is useful for analyzing the application activity and throughput.
	Failed SQL rate: Indicates the number of SQL statements that were attempted, but failed.	Stmts/Sec	This measure helps in determining reasons for poor performance, since, failed statements means resource wastage and lower throughput for the database.
	UID SQL rate: Indicates the number of SQL UPDATE, INSERT, and DELETE statements that were executed	Stmts/Sec	This information can be useful for analyzing application activity and throughput.
	SQL failures: Indicates the percentage of SQL statements that failed during the interval.	Percent	A relatively high value indicates a problem. The percentage of SQL statements that received a negative SQL code indicates a possible cause of poor performance. Failed SQL statements waste system resources. Hence, this value of this measure should be very low.
	Percent of DDL SQLs: Denotes the percentage of SQL statements that are DDL during a specific interval.	Percent	This value should normally be low.
	Percent of UID SQLs: Percentage of SQL statements that are update/insert/delete during the interval.	Percent	This measure can be used to determine the level of database activity at the application or database level.

Note:

The values for these measures are dependent on the type of application that the database is supporting. For example, in an On-Line Transaction Processing System (OLTP), we can expect a relatively high number of Updates, Inserts, and Deletes compared to a Decision Support System (DSS).

4.3 Monitoring the IBM DB2 Server in a DPF Environment

The Database Partitioning Feature (DPF) is available on DB2 UDB Enterprise Server Edition (ESE). With DPF your database is scalable as you can add new machines and spread your database across them. This means more CPUs, more memory and more disks from each of the additional machines for your database! DB2 UDB ESE with DPF is ideal to manage data warehousing, data mining and online analytical processing (OLAP) workloads. It can also work well with online transaction processing (OLTP) workloads.

When a database is partitioned, you split your database into different independent parts, each consisting of its own data, configuration files, indexes and transaction logs. Each of these parts is a database partition. You can assign multiple partitions to a single physical machine. These are called 'logical partitions' and they share the resources of the machine.

A single-partition database is a database with only one partition.

A multi-partition database (also referred to as a partitioned database), is a database with two or more partitions. Depending on your hardware environment, there can be several configurations in which you can partition your database. Figure 4.10 shows the configuration of one logical partition in a single SMP machine.

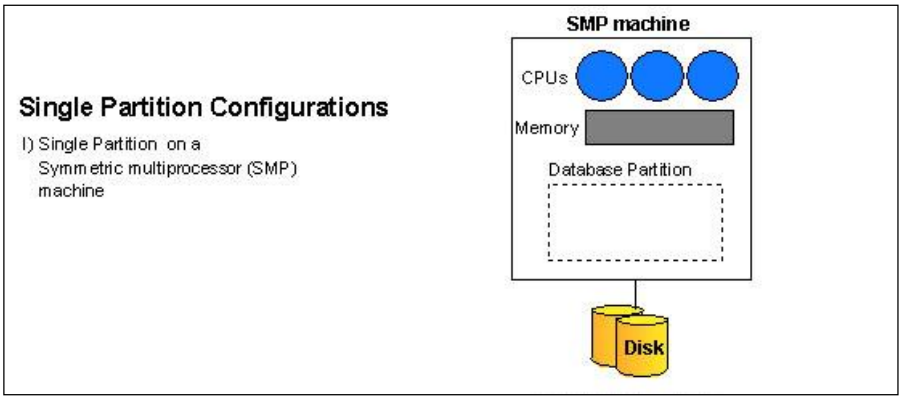


Figure 4.10: A Single-partition Configuration

Figure 4.11 shows more multi-partition configurations with several logical partitions in a machine.

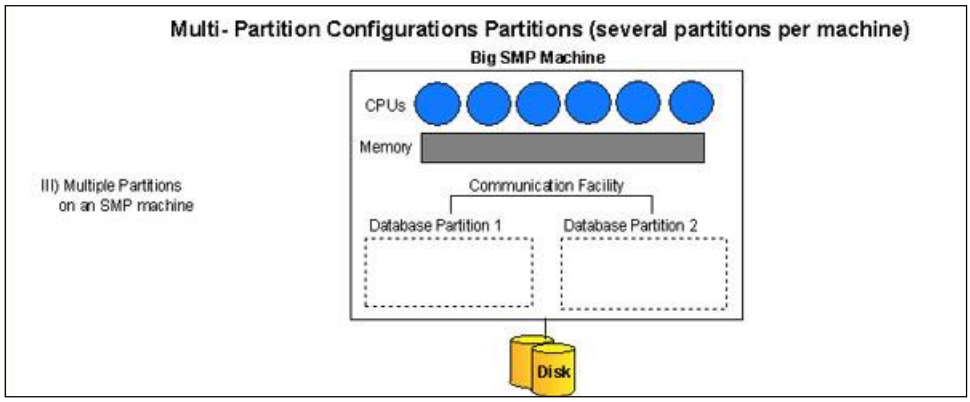


Figure 4.11: A multi-partition config

A user will connect to the database and issue queries as usual without a need to know that the database has been partitioned.

Figure 4.12 visualizes how a DB2 environment is split in a DPF system.

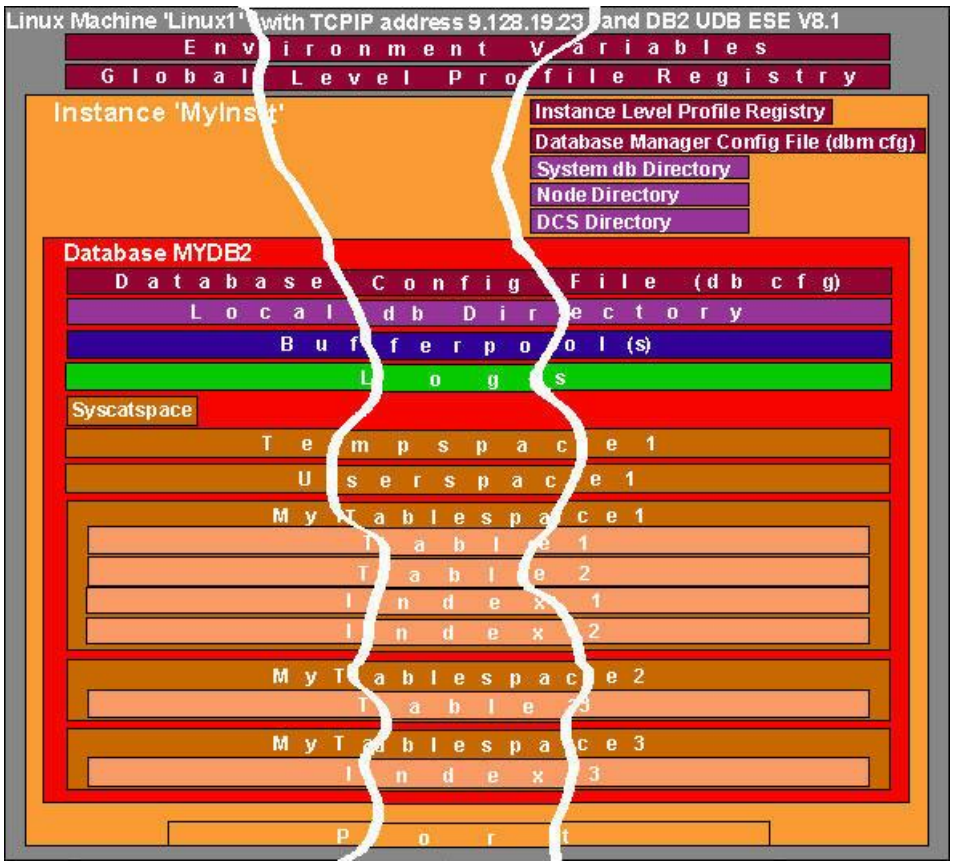


Figure 4.12: A visualization of a DPF system

eG Enterprise provides an exclusive *DB2 DPF* monitoring model that monitors the load on the DB2 server, and reveals whether the load is uniformly distributed across all the logical partitions.

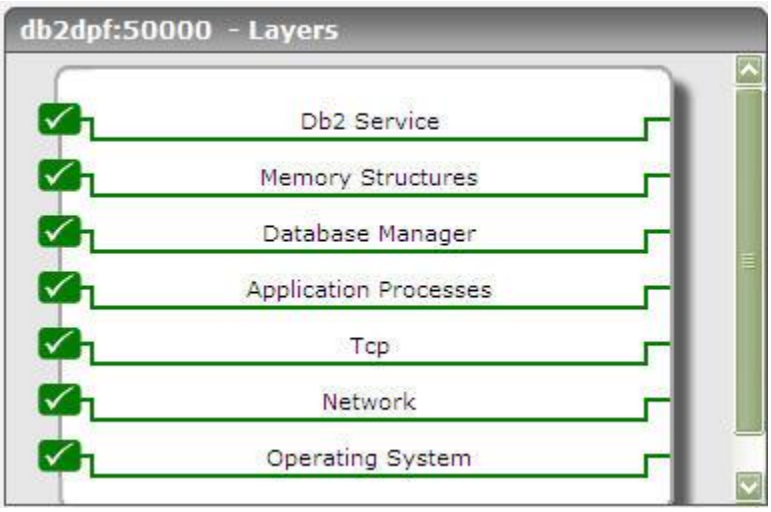


Figure 4.13: The DB2 DPF Monitoring Model

Each layer of this model is mapped to a variety of tests that report useful statistics that provide accurate answers to persistent performance queries:

- What is the current connection load on the DB2 database manager? How many of these connections are local, and how many are remote connections?
- How quickly does the DB2 server process requests from client applications? Are there sufficient agents in the agent pool to service all the client requests?
- Are the agents in the pool utilized optimally, or are too many agents idle?
- Does the database server perform sorting efficiently? Has adequate sort heap space been allocated to the database manager to enable this?
- Are sort overflows kept at a minimum?
- Does sorting take too long?
- Are lock escalations occurring too frequently on the logical partitions?
- Do applications obtain locks quickly, or do they have to wait too long for locks?
- Are the database buffer pools adequately sized for each logical partition?
- Are the page cleaners and prefetchers been utilized effectively by each logical partition?
- Are too many rollbacks happening on the logical partitions?
- Have too many SQL statements failed on any logical partition? If so, which one is it?

The sections to come elaborate on each layer of Figure 4.1, the tests associated with them, and the statistics they extract.

4.3.1 The Database Manager Layer

Using the tests associated with the **Database Manager** layer, the following can be monitored:

- Critical activities performed by the database manager
- Client connections to the database manager

- Usage of the agent pools on the database manager

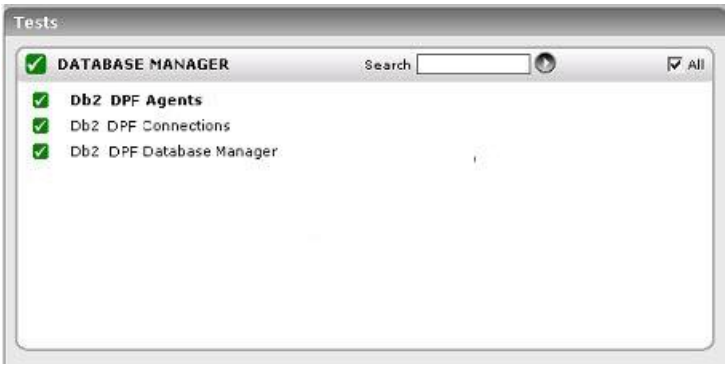


Figure 4.14: The tests associated with the Database Manager layer

4.3.1.1 Db2 DPF Agents Test

An agent is a process or thread that carries out the requests made by a client application. Each connected application is served by exactly 1 coordinator agent and possibly, a set of subordinator agents or subagents. Subagents are used for parallel SQL processing in partitioned databases and on SMP machines.

For partitioned database environments, each partition (that is, each database server or node) has its own pool of agents from which subagents are drawn. Because of this pool, subagents do not have to be created and destroyed each time one is needed or has finished its work. The subagents can remain as associated agents in the pool and be used by the database manager for new requests from the application they are associated with.

The Db2Agents test monitors how effectively the agent pool has been utilized.

Purpose	Monitors how effectively the agent pool has been utilized		
Target of the test	A DB2 database server with DPF enabled		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test		
Outputs of the test	One set of results for every DB2 database server being monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Total Agents:** Indicates the total number of agents currently registered in the database manager instance that is being monitored (Sum of coordinator agents and subagents).	Number	This measure can be used to evaluate the setting for MAXAGENTS configuration parameter.
	Idle Agents: Indicates the percentage of agents in the agent pool that is currently unassigned to an application and are, therefore, "idle".	Percent	Having idle agents available to service requests for agents can improve performance. So you can use this measure to help set the NUM_POOLAGENTS configuration parameter.
	Agents waiting on token: Indicates the percentage of agents waiting for a token so they can execute a transaction in the database manager.	Percent	You can use this element to help evaluate your setting for the MAXCAGENTS configuration parameter. Each application has a dedicated coordinator agent to process database requests within the database manager. Each agent has to get a token before it can execute a transaction. The maximum number of agents that can execute database manager transactions is limited by the configuration parameter MAXCAGENTS.
	Agents creation ratio: Indicates the ratio of number of agents assigned directly from agent pool to the total number of agents used to service requests.	Percent	A high percentage indicates the effectiveness of the agent pool. A consistent low value indicates that the number of agents in the agent pool are not adequate to service requests. You might want to consider increasing the NUM_POOLAGENTS setting in this case.
	Stolen Agents: Indicates the number of times that agents are stolen from an application. Agents are stolen when an idle agent associated with an application is reassigned to work on a different application.	Number	If this value is high, consider increasing the NUM_POOLAGENTS configuration parameter.

4.3.1.2 Db2 DPF Connections Test

The Db2 Connections test reports key statistics pertaining to the local and remote connections to the DB2 database manager.

Purpose	Reports key statistics pertaining to the local and remote connections to the DB2 database manager

<table>
<tr><td>Target of the test</td><td colspan="3">A DB2 database server with DPF enabled</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the DB2 server
3. PORT – The port number through which the DB2 server communicates. The default port is 50000.
4. USER - Specify the name of the user who has any of the following privileges to the specified DATABASE: SYSADM or SYSCTRL or SYSMAINT or SYSMON.
5. PASSWORD - Enter the password of the specified USER in the PASSWORD text box.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. DATABASE - Specify the name of the database on the monitored DB2 server to be used by this test.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every DB2 database server being monitored</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total Connections:
Indicates the total number of local and remote connections that are currently present in the database manager.</td><td>Number</td><td></td></tr>
<tr><td>Local Connections:
Indicates the number of local applications that are currently connected to a database within the database manager instance being monitored.</td><td>Number</td><td>This number can help you determine the level of concurrent processing occurring in the database manager. This number only includes applications that were initiated from the same instance as the database manager. The applications are connected, but may or may not be executing a unit of work in the database. When used in conjunction with the Remote connections measurement, this measure can help you adjust the setting of the MAXAGENTS configuration parameter.</td></tr>
<tr><td>Remote Connections:
Indicates the percentage of agents waiting for a token so they can execute a transaction in the database manager.</td><td>Percent</td><td>This number does not include applications that were initiated from the same instance as the database manager. When used in conjunction with the Local connections measure, this measure can help you adjust the setting of the MAX_COORDAGENTS configuration parameter.</td></tr>
</table>

	Local connections in exec: Indicates the number of local applications that are currently connected to a database within the database manager instance being monitored and are currently processing a unit of work.	Number	This number can help you determine the level of concurrent processing occurring in the database manager. This number only includes applications that were initiated from the same instance as the database manager. When used in conjunction with the *Remote connections in exec* measure, this measure can help you adjust the setting of the MAXCAGENTS configuration parameter.
	Remote connections in exec: Indicates the number of remote applications that are currently connected to a database and are currently processing a unit of work within the database manager instance being monitored.	Number	This number can help you determine the level of concurrent processing occurring on the database manager. This number does not include applications that were initiated from the same instance as the database manager. When used in conjunction with the *Local connections in exec* measure, this metric can help you adjust the setting of the MAXCAGENTS configuration parameter.

4.3.1.3 Db2 DPF Database Manager Test

The database manager includes the database engine and the facilities to access data, such as the command line processor and the application interfaces. This test reports key statistics pertaining to the health of the DB2 database manager.

Purpose	Reports key statistics pertaining to the health of the DB2 database manager
Target of the test	A DB2 database server with DPF enabled
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test
Outputs of the test	One set of results for every DB2 database server being monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Sorts post threshold: Indicates the number of sorts per second that have requested heaps after the sort heap threshold has been exceeded.	Sorts/Sec	Under normal conditions, the database manager will allocate sort heap using the value specified by the SORTHEAP configuration parameter. If the amount of memory allocated to sort heaps exceeds the sort heap threshold (SHEAPTHRES configuration parameter), the database manager will allocate sort heap using a value less than that specified by the SORTHEAP configuration parameter. Each active sort on the system allocates memory, which may result in sorting taking up too much of the system memory available. Sorts that start after the sort heap threshold has been reached may not receive an optimum amount of memory to execute, but, as a result, the entire system may benefit. By modifying the sort heap threshold and sort heap size configuration parameters, sort operation performance and overall system performance can be improved. If this element's value is high, you can: ii. Increase the sort heap threshold (SHEAPTHRES), or, jj. Adjust applications to use fewer or smaller sorts via SQL query changes.
	Piped Sorts Requested: A sort is classified as piped sort if the sorted information can return directly without requiring a temporary table to store a final, sorted list of data. This measure reports the number of piped sorts that have been requested per second..	Sorts/Sec	Piped sorts may reduce disk I/O. Allowing more piped sorts therefore, can improve the performance of sort operations and possibly the performance of the overall system.

	Piped Sorts Rejected: Indicates the percentage of piped sort requests that have been rejected.	Percent	When the number of rejected piped sorts are high, you can improve sort performance by adjusting one or both of the following configuration parameters: kk. SORTHEAP ll. SHEAPTHRES If piped sorts are being rejected, you might consider decreasing your sort heap or increasing your sort heap threshold. You should be aware of the possible implications of either of these options. If you increase the sort heap threshold, then there is the possibility that more memory will remain allocated for sorting. This could cause the paging of memory to disk. If you decrease the sort heap, you might require an extra merge phase that could slow down the sort.
	Hash Join Post Threshold: Indicates the total number of times that a hash join heap request was limited due to concurrent use of shared or private sort heap space.	Hits/Sec	If this value is large, the sort heap threshold should be increased.

4.3.2 The Memory Structures Layer

The tests mapped to the **Memory Structures** layer (see Figure 4.3), report critical statistics that reveal:

- How efficiently the locking and sorting activities occur on the monitored DB2 database
- How well the buffer pools are managed
- The level of I/O activity on the DB2 database

Figure 4.15: The tests associated with the Memory Structures layer

4.3.2.1 Db2 DPF Locks Test

Typically, locking activity is governed by the following factors:

- Concurrency and granularity
- Lock compatibility
- Lock conversion
- Lock escalation
- Lock waits and timeouts
- Deadlocks

In the event of an application slowdown, the measures reported by the Db2 Locks test enable administrators to accurately determine whether/not any of the above-mentioned factors have adversely impacted application performance, and if so, to what extent.

Purpose	Monitors the locking activity on the DB2 database server and reports critical statistics that will enable administrators to accurately determine what has caused a significant dip in application performance		
Target of the test	A DB2 database server with DPF enabled		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test		
Outputs of the test	One set of results for each logical partition of every database on the DB2 database server that is currently active		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Locks Held: Indicates the total number of locks currently held by all applications in this partition of this database.	Number	

	Locks Escalated: Indicates the number of times every second that locks have been escalated from several row locks to a table lock. A lock is escalated when the total number of locks held by an application reaches the maximum amount of lock list space available to the application, or the lock list space consumed by all applications is approaching the total lock list space. The amount of lock list space available is determined by the maxlocks and locklist configuration parameters. When an application reaches the maximum number of locks allowed and there are no more locks to escalate, it will then use space in the lock list allocated for other applications. When the entire lock list is full, an error occurs. This data item includes a count of all lock escalations, including exclusive lock escalations.	Escalations/Sec	There are several possible causes for excessive lock escalations: mm. The lock list size (locklist) may be too small for the number of concurrent applications nn. The percent of the lock list usable by each application (maxlocks) may be too small oo. One or more applications may be using an excessive number of locks. pp. To resolve these problems, you may be able to: qq. Increase the locklist configuration parameter value. rr. Increase the maxlocks configuration parameter value. Identify the applications with large numbers of locks or those that are holding too much of the lock list. These applications can also cause lock escalations in other applications by using too large a portion of the lock list. These applications may need to resort to using table locks instead of row locks, although table locks may cause an increase in lock_waits and lock_wait_time.

	Exclusive Lock Escalations: Indicates the number of times per second that locks have been escalated from several row locks to one exclusive table lock, or the number of times (per second) an exclusive lock on a row caused the table lock to become an exclusive lock.	Escalations/Sec	Other applications cannot access data held by an exclusive lock; therefore it is important to track exclusive locks since they can impact the concurrency of your data. A lock is escalated when the total number of locks held by an application reaches the maximum amount of lock list space available to the application. The amount of lock list space available is determined by the locklist and maxlocks configuration parameters. When an application reaches the maximum number of locks allowed and there are no more locks to escalate, it will then use space in the lock list allocated for other applications. When the entire lock list is full, an error occurs. See *Lock escalations* for possible causes and resolutions to excessive exclusive lock escalations. An application may be using exclusive locks when share locks are sufficient. Although share locks may not reduce the total number of lock escalations share lock escalations may be preferable to exclusive lock escalations.
	Locks Timedout: Indicates the number of times that a request to lock an object timed-out instead of being granted.	Timeouts/Sec	This measurement can help you adjust the setting for the locktimeout database configuration parameter. If the number of lock time-outs becomes excessive when compared to normal operating levels, you may have an application that is holding locks for long durations. In this case, this element may indicate that you should analyze some of the other lock and deadlock monitor elements to determine if you have an application problem. You could also have too few lock time-outs if your **locktimeout** database configuration parameter is set too high. In this case, your applications may wait excessively to obtain a lock.
	Lock Waits: Indicates the total number of times per second that applications or connections in this partition waited for locks.	Waits/Sec	If the value is consistently high, find t the applications or connections causing lock waits and fine tune the appropriate SQL queries.
	Average Lock Wait Time: Indicates the average time that all the applications were waiting for a lock.	Secs	If the average lock wait time is high, you should look for applications that hold many locks, or have lock escalations, with a focus on tuning your applications to improve concurrency, if appropriate.

	Percent of Application in Lock Wait: Indicates the percentage of applications in this partition waiting for the release of a lock.	Percent	If this value is high, the applications may have concurrency problems, and you should identify applications that are holding locks or exclusive locks for long periods of time.
	Deadlocks: Indicates the total number of deadlocks that have been detected per second in this partition.	Deadlocks/Sec	This element can indicate that applications are experiencing contention problems. These problems could be caused by the following situations: • Lock escalations are occurring for the database • An application may be locking tables explicitly when system-generated row locks may be sufficient. • An application may be using an inappropriate isolation level when binding • Catalog tables are locked for repeatable read • Applications are getting the same locks in different orders, resulting in deadlock You may be able to resolve the problem by determining in which applications (or application processes) the deadlocks are occurring. You may then be able to modify the application to enable it to execute concurrently. Some applications, however, may not be capable of running concurrently.

4.3.2.2 Db2 DPF Pools Test

A buffer pool is an area of memory into which database pages are read, modified, and held during processing.

Buffer pools improve database performance. If a needed page of data is already in the buffer pool, that page is accessed faster than if that page had to be read directly from disk. The database manager has agents whose tasks are to retrieve data pages from disk and place them in the buffer pool (prefetchers), and to write modified data pages from the buffer pool back to disk (page cleaners).

The reading and writing of data pages to and from disk is called disk input/output (I/O). Avoiding the wait associated with disk I/O is the primary way to improve the performance of the database. How you create the buffer pool, and configure the database manager and the agents associated with the buffer pool, controls the performance of the database. Through SQL and configuration parameters, you can control the size of the buffer pool, the number of

prefetchers and page cleaners that move data pages into and out of the buffer pool, the size of the data pages, and the number of data pages that can be moved at one time.

In Figure 3, we showed bufferpools split across the different partitions. Interpreting this figure for buffer pools, is different than for the other objects, because the data cached in the bufferpools is not partitioned as the figure may imply. What is actually happening is that buffer pools in a DPF environment can be tailored to the different partitions. Using the CREATE BUFFERPOOL statement with the DATABASE PARTITION GROUP clause, you can associate a bufferpool to a given partition group. What this means is that you have the flexibility to define the buffer pool to the specific partitions defined in the partition group. In addition, the size of the buffer pool on each partition in the partition group can be different if desired.

The statistics reported by the Db2 Pools test help administrators analyze the usage of the buffer pools, and provides them with useful pointers to fine-tune the configuration of the buffer pools.

<table>
<tr><td>Purpose</td><td colspan="3">Helps administrators analyze the usage of the buffer pools, and provides them with useful pointers to fine-tune the configuration of the buffer pools</td></tr>
<tr><td>Target of the test</td><td colspan="3">A DB2 database server with DPF enabled</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the DB2 server
3. PORT – The port number through which the DB2 server communicates. The default port is 50000.
4. USER - Specify the name of the user who has any of the following privileges to the specified DATABASE: SYSADM or SYSCTRL or SYSMAINT or SYSMON.
5. PASSWORD - Enter the password of the specified USER in the PASSWORD text box.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. DATABASE - Specify the name of the database on the monitored DB2 server to be used by this test</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every logical partition of each database on the DB2 database server that is currently active</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td></td><td>Buffer Pool Hit Ratio:
Indicates the percentage of the pages requested that is readily available in the memory of this logical partition without doing disk I/O.</td><td>Percent</td><td>This measure is an overall indicator of how well the buffer pool is performing. Lower values indicate that more physical I/O is happening than logical. Since physical I/O costs more, maintaing higher buffer hit ratio is desired. Typically a hit ratio over 90% shows that the buffer pool is performing well. If this measure is consistently low, increase the size of the buffer pool by increasing BUFFPAGE configuration value.</td></tr>
</table>

	Buffer Pool Hit Ration (Data): Indicates the percentage of the data pages requested that is readily available in the memory of this logical partition without doing disk I/O.	Percent	This measure is an indicator of how well the buffer pool is performing for the data page requests. Lower values indicate that more physical I/O is happening than logical. Since physical I/O costs more, maintaing higher buffer hit ratio is desired. Typically a hit ratio over 90% shows that the buffer pool is performing well. If this measure is consistently low, increase the size of the buffer pool by increasing BUFFPAGE configuration value.
	Buffer Pool Hit Ratio (Index): Indicates the percentage of the index pages requested that is readily available in the memory of this logical partition without doing disk I/O.	Percent	This measure is an overall indicator of how well the buffer pool is performing. Lower values indicate that more physical I/O is happening than logical. Since physical I/O costs more, maintaing higher buffer hit ratio is desired. Typically a hit ratio over 90% shows that the buffer pool is performing well. If this ratio is really low and the overall ratio is relatively high, then break the index(s) out into their own tablespace/buffer pool.
	Pre Fetch Ratio: The ratio of asynchronous reads to synchronous reads. The value indicates how effectively the DB2 database manager is populating the buffer pools through the use of prefetchers.	Percent	High value indicates more asynchronous I/O is happening than synchronous. The value can be used to tune the **num_ioservers** configuration parameter.
	Percent Log Cleans: Indicates the percentage of times a page cleaner was invoked because the logging space used had reached a predefined criterion for the database logical partition.	Percent	If this value is high (say > 40%), this could mean that page cleaners are constantly being utilized to clean the log and aren't available for other page cleaning activities, hampering performance. On the other hand, if the value is low, (say < 10%) then the page cleaners aren't being triggered as often for this activity. this means that they would be more available for the other types of page cleaning activities, which is great for buffer pool performance.
	Percent Dirty Page Cleans: Indicates the percentage of times a page cleaner was invoked because the buffer pool defined for this logical partition had reached the dirty page threshold criterion for the database.	Percent	The threshold is set by the chngpgs_thresh configuration parameter. It is a percentage applied to the buffer pool size. When the number of dirty pages in the pool exceeds this value, the cleaners are triggered. If this value is set too low, pages might be written out too early, requiring them to be read back in. If set too high, then too many pages may accumulate, requiring users to write out pages synchronously.

	Percent Victim Cleans: Indicates the percentage of times the page cleaner(s) were triggered to oust a victim page from the buffer pool. A victim page is a clean or dirty page in the buffer pool that is removed simply because DB2 needs to make room for incoming pages. If a victim page is a dirty page then the information must be written out to disk. Any page that is removed will most likely cause more physical I/O to occur in order to retrieve it again at later time when DB2 is ready to use it.	Percent	If the ratio is higher then the above two then that is typically a good indicator that the buffer pool needs to be larger since there never seems to be enough room for new pages to be brought in. This could also be a sign that dirty pages are staying in the buffer pool too long which could mean that the changed pages threshold (CHNGPGS_THRESH) is set too high. Even the SOFTMAX parameter could be set too high and too much of the changed pages that are logged are not getting flushed out to make way for new pages. If this ratio is low, it may indicate that you have defined too many page cleaners. If your chngpgs_thresh is set too low, you may be writing out pages that you will dirty later. Aggressive cleaning defeats one purpose of the buffer pool, that is to defer writing to the last possible moment.
	Catalog Cache Hit Ratio: Indicates the percentage of time the requested information for table descriptor or authorization was readily availble in catalog cache without requiring to perform disk I/O.	Percent	The catalog cache is referenced whenever a table, view, or alias name is processed during the compilation of an SQL statement. If the ratio is greater than 80%, then the catalog cache is performing well. A smaller value indicates that the catalog cache size should be increased by tuning the parameter CATALOGCACHE_SZ in the database configuration. The value may be low immediately following the first connection to the database. The execution of Data Definition Language (DDL) SQL statements involving a table,view, or alias will evict the table descriptor information for that object from the catalog cache causing it to be re-inserted on the next reference. Therefore, the heavy use of DDLs may also increase the value of the measure.
	Package cache hit ratio: The package and section information required for the execution of dynamic and static SQL statements are placed in the package cache as required. This information is required whenever a dynamic or static statement is being executed. The ratio indicates the effectiveness of package cache hit ratio.	Percent	If the hit ratio is high (more than 80%), the cache is performing well. A smaller ratio may indicate that the package cache size (pckcachesz) should be increased.

4.3.2.3 Db2 DPF Sorts Test

Sorting represents organizing the rows in a table into the order of one or more of its columns, optionally eliminating duplicate entries. Sorting is required when:

- No index exists to satisfy a requested ordering (for example a SELECT statement that uses the ORDER BY clause).
- An index exists but sorting would be more efficient than using the index
- An index is created.
- An index is dropped, which causes index page numbers to be sorted.

Because queries often require sorted or grouped results, sorting is often required, and the proper configuration of the sort heap areas is crucial to good query performance. Using the Db2Sort test, administrators can figure out whether/not the sort heap allocations are sufficient to facilitate efficient sorting.

Purpose	Helps administrators figure out whether/not the sort heap areas are adequately configured to facilitate efficient sorting		
Target of the test	A DB2 database server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test		
Outputs of the test	One set of results for every logical partition of each database on the DB2 database server that is currently active		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Sort Heap Allocated:** Indicates the sum of sort heap space allocated for all sorts in this database partition.	Pages	Normal memory estimates do not include sort heap space. If excessive sorting is occurring, the extra memory used for the sort heap should be added to the base memory requirements for running the database manager. Generally, the larger the sort heap, the more efficient the sort. Appropriate use of indexes can reduce the amount of sorting required. You may use the information returned at the database manager level to help you tune the SHEAPTHRES configuration parameter. If the element value is greater than or equal to SHEAPTHRES, it means that the sorts are not getting the full sort heap as defined by the SORTHEAP parameter.
	Average Sort Heap Space Used: Indicates the average sort heap space used by each sort.	Pages	If the SORTHEAP configuration parameter is substantially larger than the average sort heap used, you may be able to lower the value of this parameter.
	Sort Rate: Indicates the number of sort operations performed on this logical partition during the last measurement period.	Sorts / Sec	
	Percent Sort Overflow: Indicates the percentage of sorts in this logical partition that had to overflow to disk.	Percent	Sort overflows are sorts that ran out of sort heap and may have required disk space for temporary storage. When a sort overflows, additional overhead will be incurred because the sort will require a merge phase and can potentially require more I/O, if data needs to be written to disk. If this percentage is high, you may want to adjust the database configuration by increasing the value of sortheap.
	Average Sort Time: Indicates the average sort time for all sorts performed by all applications connected to this logical partition.	Secs	A high value indicates the poor performance of sorting operations. Identify the statements that spend lot of time sorting. You may want to reduce the average sort time for these statements by increasing the sortheap parameter.

4.3.2.4 Db2 DirectI/O Test

This test monitors the I/O activity on the currently active logical partitions.

Purpose	Monitors the I/O activity on the currently active logical partition

<table>
<tr><td>Target of the test</td><td colspan="3">A DB2 database server with DPF enabled</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the DB2 server
3. PORT – The port number through which the DB2 server communicates. The default port is 50000.
4. USER - Specify the name of the user who has any of the following privileges to the specified DATABASE: SYSADM or SYSCTRL or SYSMAINT or SYSMON.
5. PASSWORD - Enter the password of the specified USER in the PASSWORD text box.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. DATABASE - Specify the name of the database on the monitored DB2 server to be used by this test</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every logical partition of each database on the DB2 database server that is currently active</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Direct Read Rate:
Indicates the total number of direct reads by the application per second. In other words, it is the number of read operations that do not use the buffer pool.</td><td>Reads/Sec</td><td>Direct reads are performed in units, the smallest being a 512-byte sector. They are used when:
• Reading LONG VARCHAR columns
• Reading LOB (large object) columns
• Performing a backup
A high value over a period of time may be indicative of a performance bottleneck.</td></tr>
<tr><td>Direct Write Rate:
Indicates the total number of direct writes by the application per second. In other words, it is the number of write operations that do not use the buffer pool.</td><td>Writes/Sec</td><td>Direct writes are performed in units, the smallest being a 512-byte sector. They are used when:
• Writing LONG VARCHAR columns
• Writing LOB (large object) columns
• Performing a restore
• Performing a load
A high value over a period of time may be indicative of a performance bottleneck.</td></tr>
</table>

	Buffer Pool IO Rate: Indicates the rate at which the buffer pool I/O operations are being done in this logical partition.	IOs/Sec	In conjunction with the hit ratio statistics, and the characteristics of the applications executing, the I/O load may require adjustment of BUFFPAGE, or applications may require further tuning.

4.3.3 The Db2 Service Layer

The tests associated with the **Db2 Service** layer indicate the level of SQL activity on the server, and the number and type of transactions that occur on the server. These measurements together serve as effective indicators of the processing ability of the DB2 server.

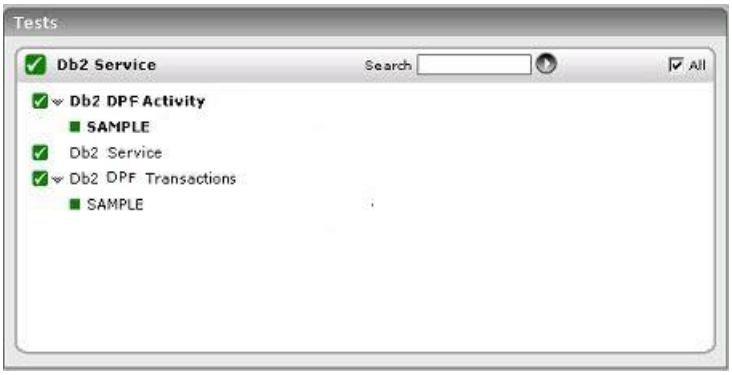


Figure 4.16: The tests associated with the Db2 Service layer

4.3.3.1 Db2 DPF Activity Test

This test measures the level of SQL activity on the logical partitions, and reveals how well the partitions process SQL queries.

Purpose	Measures the level of SQL activity on the logical partitions, and reveals how well the partitions process SQL queries
Target of the test	A DB2 database server with DPF enabled
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test		
Outputs of the test	One set of results for every logical partition of each database on the DB2 database server that is currently active		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Dynamic SQL Rate: Indicates the rate of dynamic SQL statements that were attempted on this logical partition.	Stmts/Sec	This is an indication of throughput of the system during the monitoring period. A high value of dynamic SQLs and low value of failed SQLs indicate good throughput.
	Select SQL rate: Indicates the rate at which SELECT SQL statements were executed on this logical partition during the last measurement period.	Stmts/Sec	This measure can be used to determine the level of database activity.
	UID SQL rate: Indicates the rate at which update/delete/insert statements were issued on this logical partition during the last measurement period.	Stmts/Sec	This measure can be used to determine the level of database activity.
	Failed SQL Rate: Indicates the rate at which SQL statements to this logical partition failed.	Stmts/Sec	A relatively high value indicates a problem. Failed SQL statements waste system resources. Hence, the value of this measure should be very low.
	Percent Failed SQL: Indicates the percentage of SQL statements attempted on this partion that failed during the interval. This value includes all SQL statements that received a negative SQLCODE	Percent	

	Percent DDL SQL: Denotes the percentage of SQL statements attempted on this logical partition that were DDL (Data Definition Language) during the last measurement period.	Percent	This value should normally be low.
	Percent UID SQL: Indicates the percentage of update/insert/delete statements executed on this logical partition during the last measurement period.	Percent	This measure can be used to determine the level of database activity.

4.3.3.2 Db2 DPF Transactions Test

This test tracks various statistics pertaining to the transactions executing on each logical partition of a DB2 database.

Purpose	Tracks various statistics pertaining to the transactions executing on a DB2 database		
Target of the test	A DB2 database server with DPF enabled		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST –** The IP address of the DB2 server 3. **PORT –** The port number through which the DB2 server communicates. The default port is 50000. 4. **USER** - Specify the name of the user who has any of the following privileges to the specified **DATABASE**: SYSADM or SYSCTRL or SYSMAINT or SYSMON. 5. **PASSWORD** - Enter the password of the specified **USER** in the **PASSWORD** text box. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **DATABASE** - Specify the name of the database on the monitored DB2 server to be used by this test		
Outputs of the test	One set of results for every logical partition of each database on the DB2 database server that is currently active		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Commit Rate:** Indicates the transaction throughput. This measure is the sum of the committed statements attempted on this logical partition and internal commits (total number of commits initiated internally by the database manager) per second.	Commits/Sec	A decrease in this measure during the monitoring period may indicate that the applications are not doing frequent commits. This may lead to problems with logging and data concurrency. The cause has to be probed in the application.
	Rollback Rate: Indicates the rate of unit of work rollbacks.	Rollbacks/Sec	A high rollback rate is an indicator of bad performance, since work performed up to the rollback point is wasted. The cause of the rollbacks has to be probed in the application.
	Transaction Rate: Indicates the rate of commits and rollbacks for the application using the DB2 Connect gateway.	Trans/Sec	A high transaction rate with high rollback rate indicates bad performance.

4.3.4 DB2 DPF SQL Workload Test

Nothing can degrade the performance of DB2 partition like a resource-hungry or a long-running query! When such queries execute on a logical partition, they either hog almost all the available CPU, memory, and disk resources or keep the resources locked for long time periods, thus leaving little to no resources for carrying out other critical database operations. This can significantly slowdown the partition and adversely impact user experience with the partition. To ensure peak performance of the logical partitons at all times, such queries should be rapidly identified and quickly optimized to minimize resource usage. This is where the **DB2 DPF SQL Workload** test helps. At configured intervals, this test compares the usage levels and execution times of all queries that started running on the logical partitions in the last measurement period and identifies a 'top query' in each of the following categories - CPU usage, memory usage, disk activity, and execution time. The test then reports the resource usage and execution time of the top queries and promptly alerts administrators if any query consumes more resources or takes more time to execute than it should. In such a scenario, administrators can use the detailed diagnosis of this test to view the inefficient queries and proceed to optimize them to enhance server performance.

Purpose	At configured intervals, this test compares the usage levels and execution times of all queries that started running on the logical partitions in the last measurement period and identifies a 'top query' in each of the following categories - CPU usage, memory usage, disk activity, and execution time. The test then reports the resource usage and execution time of the top queries and promptly alerts administrators if any query consumes more resources or takes more time to execute than it should. In such a scenario, administrators can use the detailed diagnosis of this test to view the inefficient queries and proceed to optimize them to enhance server performance.
Target of the test	A DB2 DPF server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the DB2 server
3. PORT – The port number through which the DB2 server communicates. The default port is 50000.
4. USER - Specify the name of the user who has any of the following privileges to the specified DATABASE: SYSADM or SYSCTRL or SYSMAINT or SYSMON.
5. PASSWORD - Enter the password of the specified USER in the PASSWORD text box.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. DATABASE - Specify the name of the database on the monitored DB2 server to be used by this test.
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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the logical partitions monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Top most query physical reads:
Indicates the number of physical disk reads performed by the top query per execution.</td><td>Reads/execution</td><td>If the value of this measure is abnormally high, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries generating maximum physical disk activity. From this, you can identify the top query in terms of number of physical disk reads. You may then want to optimize the query to reduce the disk reads.</td></tr>
<tr><td>Top most buffer gets:
Indicates the number of memory buffers used by the top query per execution.</td><td>Memorybuffer gets/execution</td><td>If the value of this measure is abnormally high, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries consuming memory excessively. From this, you can easily pick that query which is consuming the maximum memory. You may then want to optimize the query to minimize memory usage.</td></tr>
</table>

	Top most query CPU time: Indicates the duration for which each execution of the top query was hogging the CPU resources.	Secs/execution	If the value of this measure is over 30 seconds, you can use the detailed diagnosis of this measure to the top-5 (by default) queries hogging the CPU resources. From this, you can easily pick that query which is consuming the maximum CPU. You may then want to optimize the query to minimize CPU usage.
	Top most query elapsed time: Indicates the running time of each execution of the top query.	Secs/execution	If the value of this measure crosses 10 seconds, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries that are taking too long to execute. . From this, you can easily pick that query with the maximum execution time. You may then want to optimize the query to minimize execution time.

5

Monitoring the Sybase Adaptive Servers

Adaptive Server Enterprise (ASE) is a performance-optimized relational database management system that is ideally suited for online transaction processing (OLTP) and decision support systems (DSS).

Figure 5.1 depicts the components of a typical ASE.

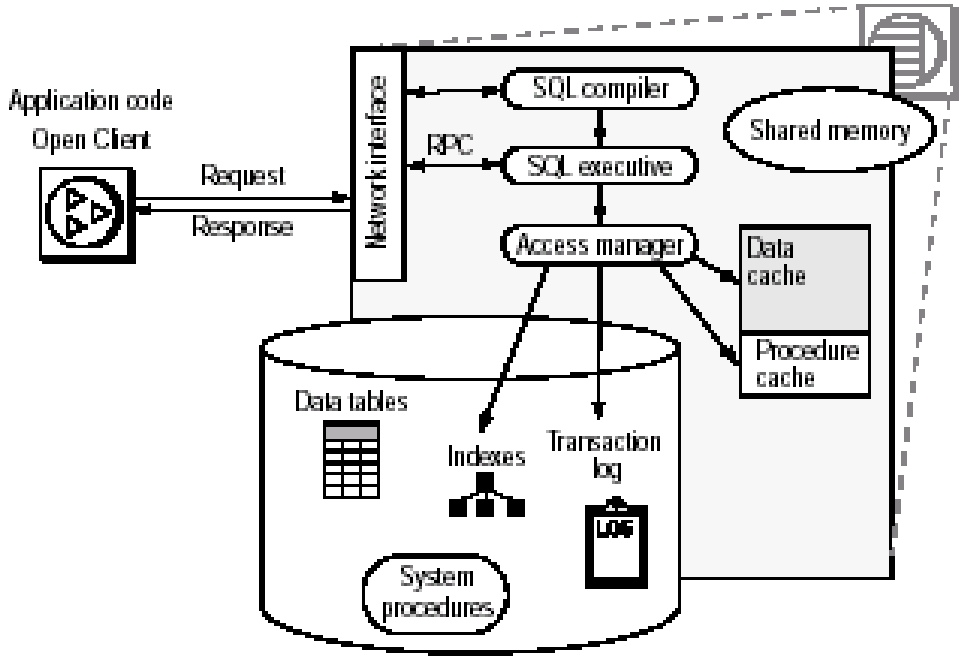


Figure 5.1: Architecture of ASE

The components include Data Tables, Transaction log, System procedures, SQL compiler, SQL Executive, Access manager, Shared Memory, Data Caches, Network Interface etc. A monitor server also comes along with an ASE. The architecture of the monitor server is shown in Figure 5.2. As the communication between the adaptive server and monitor server happens through a shared memory, the monitor server has to reside on the same machine as the adaptive server. The monitor server communicates with the adaptive server and keeps track of all performance monitoring data. It also maintains all the counters.

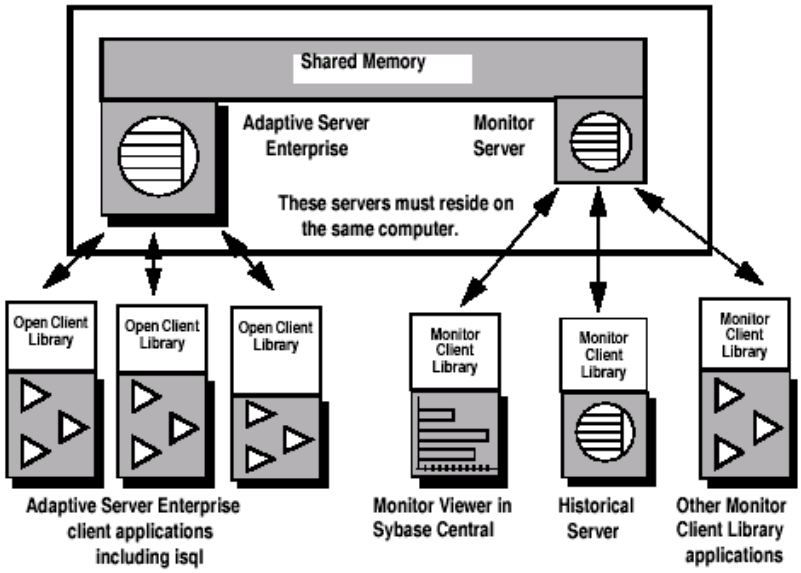


Figure 5.2: Architecture of the Monitor server

Starting in ASE 12.5.0.3, a new feature called 'MDA tables' is available to ASE users. These MDA tables provide access to low-level monitoring information in ASE; since the MDA tables can be accessed with regular SQL *select* statements, they are much easier to use than products like Monitor Server/Historical Server.

eG Enterprise supports two distinct approaches to monitoring the Sybase ASE - while one approach involves the usage of the Monitor Client Library (see Figure 5.2) for obtaining the metrics of interest, the other mandates the use of the MDA tables installed on the Sybase ASE.

The sections that follow will discuss each of these models in great detail.

5.1 Monitoring Sybase Using the Monitor Client Library

In order to extract performance data from a Sybase server using the Monitor Client Library, eG Enterprise provides the *Sybase* monitoring model (see Figure 5.3).

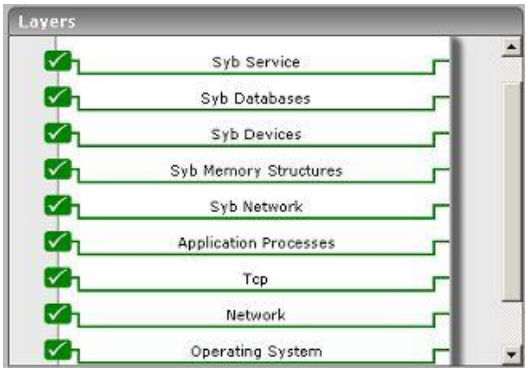


Figure 5.3: Layer model for Sybase Adaptive Server

Every layer of Figure 5.3 is associated with a series of tests, which when executed on the Sybase Adaptive server, extract a wealth of performance metrics from the server. These statistics in turn enable administrators to quickly figure out the following:

- Is the Sybase server available? If so, does it respond to requests quickly?
- Is the Sybase server overloaded?
- Is the I/O activity on the server optimal?
- How many locks are currently active on the database? Have the locks been held for an unusually long time? Do applications have to wait for too long to acquire a lock?
- Are there too many deadlocks on the database?
- Are the Sybase data and procedure caches been effectively utilized?
- Is sufficient space available for all Sybase databases? Should more space be allocated to any specific database?
- Are transactions rolled back frequently?
- Are there too many root blocker processes on the database?
- Which is the busiest program/application executing on the Sybase database in terms of number of open connections to it? Is any application consuming the CPU resources excessively?

The sections to come discuss each of the top 5 layers of Figure 5.3 elaborately, as the remaining layers have been dealt with in the *Monitoring Unix and Windows Servers* document.

5.1.1 The Syb Network Layer

This layer tracks the health of the network(s) that connect users to the adaptive server. Figure 5.4 depicts the tests mapping to this layer.

Figure 5.4: Tests mapping to the SYBASE_NET layer

5.1.1.1 Sybase Network Test

This test, executed by an internal agent, tracks various statistics pertaining to network traffic of the Sybase Adaptive Server.

Purpose	To measure the statistics pertaining to the Network in database

Target of the test	A Sybase adaptive server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – A Sybase user who has the "sa_role" and "sybase_ts_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for every database being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	I/O requests: The total number of packets received and sent per second	Pkts/Sec	
	I/O delays: The percentage of times network I/O was delayed	Percent	A non-zero value over a period of time indicates a problem.
	Receive rate: Reports the rate at which the network bytes are being received.	KBytes/Sec	This measure indicates the amount of data coming to the database server through the network. A high value of this measure indicates higher rate of incoming data to the server. If this measure is considerably high, the value of the default network packet size parameter should be increased.
	Send rate: Reports the rate at which the network bytes are being sent.	KBytes/Sec	This measure indicates the amount of data going out of the database server through the network. A high value of this measure indicates higher rate of outgoing data from the server. If this measure is considerably high, the value of the default network packet size parameter should be increased.

	Avg packet size: The average number of bytes sent per packet	Bytes	If the Adaptive server receives a command that is larger than the packet size, the server waits to begin processing until it receives the full command. Therefore, commands that require more than one packet are slower to execute and take up more I/O resources. If the average bytes per packet is near the default packet size configured for your server, you may want to configure larger packet sizes for some connections. You can configure the network packet size for all connections or allow certain connections to log in using larger packet sizes.

5.1.2 The Syb Memory Structures Layer

This layer tracks the status of the locks, deadlocks, and database caches. The tests associated with this layer are depicted in Figure 5.5.

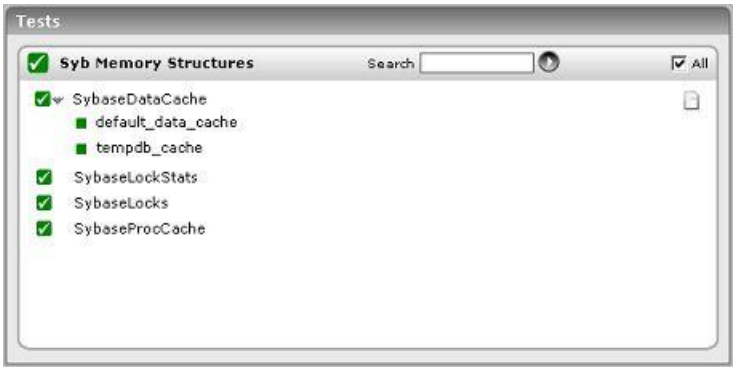


Figure 5.5: Tests mapping to the Syb Memory Structures layer

5.1.2.1 Sybase Locks Test

This test, executed by an internal agent, tracks the statistics pertaining to the lock and deadlock traffic of Sybase Adaptive Server. Locking affects the performance of the Sybase adaptive server by limiting the concurrency. An increase in the number of simultaneous users to a server may increase the lock contention thereby decreasing the performance of the server (as one process may have to wait for another process to release its lock, thereby affecting the response time and throughput). Deadlocks cause more severe damage. A deadlock causes one transaction to be aborted and transaction must be restarted by the application. If a deadlock occurs very often, it severely affects the throughput of the application.

Purpose	To measure the statistics pertaining to the locks and deadlocks in a database
Target of the test	A Sybase adaptive server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – A Sybase user who has the "sa_role".
5. PASSWORD – The password corresponding to the above user
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. 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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every database being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Lock requests:
The average number of lock requests per second</td><td>Locks/sec</td><td>A high value indicates that there is high locking activity in the system and may need close scrutiny for the type of locks being requested. The detailed diagnosis for this measure provides a listing of locks requests for each lock type.</td></tr>
<tr><td>Lock waits:
The average number of times there was lock contention</td><td>Waits/sec</td><td>A high value of waits can have an adverse impact on application performance. Possible reasons for this behavior could be:
• Inadequate number of locks available in the database
• Unusually high locking behavior of applications accessing the database
• Improper database application design, etc.
The detailed diagnosis for this measure provides a listing of waited number of locks for each lock type.</td></tr>
<tr><td></td><td>Avg lock wait time:
The average amount of wait time for each lock request that resulted in a wait</td><td>Secs</td><td>A high value may indicate that there is contention for locks in the system. When the average wait time for locks is high, users may have to wait for their transactions to complete.</td></tr>
</table>

	Lock timeouts: The number of times a task was waiting for a lock and the transaction was rolled back due to a session-level or server-level lock timeout	Timeouts/Sec	Lock timeouts can be useful for removing tasks that acquire some locks, and then wait for long periods of time blocking other users. The detailed diagnosis for this measure provides a listing of lock timeouts for each lock type.
	Deadlocks: The average number of deadlocks found in the lock requests	Number	A deadlock may arise due to various situations including bad design of queries and deficient coding practices. A deadlock is a situation where both/all the lock requestors are in a mutual or a multi-way tie. Any deadlocks are detrimental to database application performance. The detailed diagnosis for this measure provides a listing of number of dead locks for each lock type.

5.1.2.2 Sybase Lock Stats Test

A Sybase server provides data concurrency and integrity between transactions using locking mechanisms. The locking activity of a database server must be monitored carefully because an application holding a specific lock for a long time could cause a number of other transactions relying on the same lock to fail. The SybaseLockStats test monitors the locking activity on a database server instance.

Purpose	This test indicates the level of locking activity on a database in terms of the number of total locks in different modes and average block time.
Target of the test	A Sybase server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – A Sybase user who has the "sa_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.		
Outputs of the test	One set of results for every Sybase instance monitored.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Locks: Gives the total number of locks that are held.	Number	A high value may indicate one of the following: 1. Too many transactions happening 2. Locked resources not being released properly 3. Locks are being held unnecessarily.
	Avg block time due to locks: Indicates the average time for which a process is blocked	Secs	A high value may indicate one of the following: 1. Locked resources not being released properly 2. Locks are being held unnecessarily. The detailed diagnosis of this measure, if enabled, provides the complete information (i.e. duration for which the lock was held, the object that was locked, the type of lock held, etc.) pertaining to the locks that are being held currently.

5.1.2.3 Sybase Cache Test

This test reports the statistics pertaining to the default data cache and for all the named caches configured in a Sybase database server.

Purpose	Reports the statistics pertaining to the default data cache and for all the named caches configured in a Sybase database server

<table>
<tr><td>Target of the test</td><td colspan="3">A Sybase adaptive server</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – A Sybase user who has the “sa_role”.
5. PASSWORD – The password corresponding to the above user
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. 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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the default cache and every named cache being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Cache utilization:
The number of searches using this cache as a percentage of searches across all caches.</td><td>Percent</td><td>Compare this value for each cache to determine if there are caches that are over or under-utilized. If you decide that a cache is not well utilized, you can
• Change the cache bindings to balance utilization
• Resize the cache to correspond more appropriately to its utilization</td></tr>
<tr><td>Cache hit ratio:
The percentage of times that a required page was found in the data cache.</td><td>Percent</td><td>A low value indicates that the cache may be too small for the current application load.</td></tr>
</table>

	Spinlock contention: The number of times an engine encountered spinlock contention on the cache, and had to wait, as a percentage of the total spinlock requests for that cache.	Percent	If spinlock contention is more than 10%, consider using named caches or adding cache partitions.
	Buffers grabbed: The average number of times that Adaptive Server fetched a buffer from the LRU end of the cache, replacing a database page.	Buffers/Sec	The detailed diagnosis of this measure, if enabled, reports the buffers grabbed for each configured memory pool. If this value is high in some pools and low in other pools, you might want to move space from the less active pool to the more active pool, especially if it can improve the cache-hit ratio.
	Dirty buffers grabbed: The average number of times that fetching a buffer found a dirty page at the LRU end of the cache and had to wait while the buffer was written to disk.	Buffers/Sec	A nonzero value indicates that the wash area of the pool is too small for the throughput in the pool. The detailed diagnosis of this measure, if enabled, reports the dirty buffers grabbed for each configured memory pool.
	Large I/Os denied: The number of large I/O requests denied as a percentage of the total number of requests made	Percent	Large I/O requests are denied due to the following reasons: • If any page in a buffer already resides in another pool. • When there are no buffers available in the requested pool. • On the first extent of an allocation unit, since it contains the allocation page, which is always read into the 2K pool. If a high percentage of large I/Os were denied, it indicates that the use of the large pools might not be as effective as it could be.
	Large I/O effectiveness: The total number of pages that were used after being brought into cache by large I/O	Percent	A low value indicates that only few of the pages brought into cache are being accessed by queries.

5.1.2.4 Sybase Proc Cache Test

The SybaseProcCache test reports the procedure cache statistics of a Sybase database server.

Purpose	Reports the procedure cache statistics of a Sybase database server		
Target of the test	A Sybase adaptive server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – A Sybase user who has the "sa_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for the Sybase server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Procedure requests: The number of times stored procedures were executed	Trans/Sec	
	Disk reads: The number of times that stored procedures were read from disk rather than found and copied in the procedure cache	Reads/Sec	If this value is consistently high, increase the procedure cache size.
	Cache hit ratio: Percentage of procedure requests served from cache	Percent	If this value is low, consider increasing procedure cache memory or add more memory to the server.
	Procedures created: The number of procedures created during the sample interval	Procedures/Sec	

	Procedures aged out of cache: The number of times that a procedure aged out of cache	Procedures/Sec	

5.1.3 The Syb Devices Layer

This layer tracks the health of the devices in a database. Figure 5.6 shows the tests that correspond to this layer.

Figure 5.6: Tests mapping to the Syb Devices layer

5.1.3.1 Sybase Devices Test

This test, executed by an internal agent, tracks statistics pertaining to the devices of the Sybase adaptive server.

Purpose	To measure the statistics pertaining to the devices in a database		
Target of the test	A Sybase adaptive server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – A Sybase user who has the "sa_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for every database being monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**I/O requests:** The average number of IO requests for a device	IOs/Sec	
	Physical reads: The average number of reads performed on a device	Reads/Sec	
	Physical writes: The average number of writes performed on a device	Writes/Sec	
	I/O errors: The percentage of times there was an error in IO requests	Percent	
	I/O waits: The percentage of times the semaphore was busy and the IO operation had to wait for the semaphore to be released	Percent	When Adaptive Server needs to perform a disk I/O, it gives the task the semaphore for that device in order to acquire a block I/O structure. On SMP systems, multiple engines can try to post I/Os to the same device simultaneously. This creates contention for that semaphore, especially if there are hot devices or if the data is not well distributed across devices. A large value could indicate a semaphore contention issue. One solution might be to redistribute the data on the physical devices.

5.1.4 The Syb Databases Layer

Using the SybaseDatabaseSpaceUsage test associated with it, the **Syb Databases** layer monitors the space usage of ech of the databases on the Sybase Adaptive server, and reports whether any database is experiencing excessive space usage or insufficient space allocation.

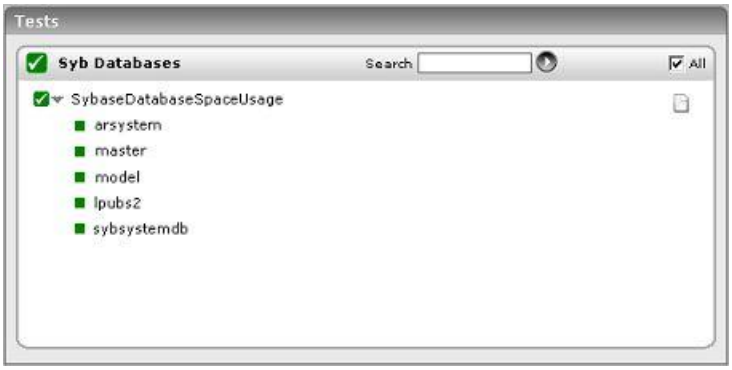


Figure 5.7: The tests associated with the Syb Databases layer

5.1.4.1 Sybase Database Space Usage Test

This test reports the space usage of all Sybase databases.

Purpose	Reports the space usage of all Sybase databases		
Target of the test	A Sybase adaptive server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – A Sybase user who has the "sa_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **INCLUDE DB** - Specify a comma-separated list of databases that you wish to monitor in the **INCLUDE DB** text box. By default, this is set to *all*. 8. **EXCLUDE DB** - Specify a comma-separated list of databases that you wish to exclude from the scope of monitoring in the **EXCLUDE DB** text box. By default, this is set to *none*.		
Outputs of the test	One set of results for every database being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Total size: The space allocated to a database	MB	
	Reserved space percent: The percentage of space reserved for tables and indexes of a database	Percent	If the value of this measure reaches 100%, it indicates that the total space in the database has been completely allocated. New tables/indexes can be added to the database, only if its total size is increased.
	Reserved space: The amount of space allocated to the tables and indexes created on a database	MB	If the value of this measure becomes equal to that of the Total_size measure, new tables/indexes can no longer be created on the database. To create new tables, you must increase the database size.
	Data space: The amount of space used by data	MB	

	Index space: The amount of space used by indexes	MB	
	Unused space: The amount of free space available in the database	MB	

5.1.5 The Syb Service Layer

This layer tracks the health of the database server by looking into how the database is responding to the applications. The tests associated with this layer are shown in Figure 5.8.

Figure 5.8: Tests mapping to the Syb Service layer

5.1.5.1 Sybase Responses Test

This test, executed by an internal agent, tracks the statistics pertaining to the availability and response time of the Sybase adaptive server.

Purpose	To measure the statistics pertaining to the availability and response time of the database server
Target of the test	A Sybase adaptive server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – A Sybase user who has the “sa_role”.
5. PASSWORD – The password corresponding to the above user
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. QUERY – By default, this parameter is set to select * from sysobjects. The test executes this executes the default query to report the availability and responsiveness of the server. If the user configured for this test does not have the right to execute the default query, then the QUERY parameter can be overridden with a query that that user has permission to execute.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every Sybase server being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Sybase server availability:
Indicates if the database instance is responding to requests or not.</td><td>Percent</td><td>A value of 100 for this measure indicates that the database is responding to a request. On the other hand, a value of 0 denotes that the database is not responding.
Availability problems may be caused by a misconfiguration/malfunctioning of the database instance, or because the instance is using an invalid user account. Besides the above, this measure will report that the server is unavailable even if a connection to the database instance is unavailable, or if a query to the database fails. In this case, you can check the values of the DB connection availability and Query processor availability measures to know what is exactly causing the database instance to not respond to requests - is it owing to a connection unavailability? or is it due to a query failure?</td></tr>
<tr><td>Total response time :
Indicates the time taken by the database server to respond to a user query. This is the sum total of the connection time and query execution time.</td><td>Secs</td><td>A sudden increase in response time is indicative of a potential performance bottleneck on the database server.</td></tr>
</table>

	DB connection availability: Indicates whether the database connection is available or not.	Percent	If this measure reports the value 100 , it indicates that the database connection is unavailable. The value 0 on the other hand indicates that the database connection is unavailable. A connection to the database may be unavailable if the database is down or if the database is listening on a port other than the one configured for it in the eG manager. If the *Availability* measure reports the value 0, then, you can check the value of this measure to determine whether/not it is due to the unavailability of a connection to the server.
	Query processor availability: Indicates whether the database query is executed successfully or not.	Percent	If this measure reports the value 100, it indicates that the query executed successfully. The value 0 on the other hand indicates that the query failed. In the event that the *Availability* measure reports the value *0*, check the value of this measure to figure out whether the failed query is the reason why that measure reported a server unavailability.
	Connection time to database server: Indicates the time taken by the database connection.	Secs	A high value could indicate a connection bottleneck. Whenever the *Response time* of the measure soars, you may want to check the value of this measure to determine whether a connection latency is causing the poor responsiveness of the server.
	Query execution time: Indicates the time taken for query execution.	Secs	A high value could indicate that one/more queries to the database are taking too long to execute. Inefficient/badly designed queries to the database often run for long periods. If the value of this measure is higher than that of the *Connection time* measure, you can be rest assured that long running queries are the ones causing the responsiveness of the server to suffer.
	Records fetched: Indicates the number of records fetched from the database.	Number	The value 0 indicates that no records are fetched from the database

5.1.5.2 Sybase System Processes Test

This test reports details about the system processes running in a Sybase database server.

Purpose	Reports details about the system processes running in a Sybase database server

<table>
<tr><td>Target of the test</td><td colspan="3">A Sybase adaptive server</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – A Sybase user who has the "sa_role".
5. PASSWORD – The password corresponding to the above user
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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every Sybase server being monitored</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Sybase processes:
The total number of Sybase processes</td><td>Number</td><td></td></tr>
<tr><td>Background processes:
The total number of background processes run by the Adaptive Server rather than by a user process</td><td>Number</td><td>The detailed diagnosis of this measure, if enabled, provides the details pertaining to the background processes currently executing.</td></tr>
<tr><td>Running processes:
The total number of running processes</td><td>Number</td><td>The detailed diagnosis of this measure, if enabled, provides details such as the ID of the running processes, the user executing each of the processes, the database on which every process is executing etc.</td></tr>
<tr><td></td><td>Sleeping processes:
The total number of sleeping processes</td><td>Number</td><td>The detailed diagnosis of this measure, if enabled, provides details such as the ID of the sleeping processes, the user executing each of the processes, the database on which every process is executing, the sleep status, sleep time etc.</td></tr>
</table>

	Infected processes: The total number of processes in which the server has detected a serious error condition	Number	The detailed diagnosis of this measure, if enabled, provides the ID of the infected processes, the user executing each of the processes, and the database on which every process is executing. This information enables the user to isolate the specific queries that are infected. Further analysis of these queries can be performed, in order to figure out the reason for the infection and take adequate measures to prevent it from recurring.
	Blocked processes: If a process attempts to access a resource that is already in use by another process, then such a process will be blocked until such time that the other process releases the resource. This measures indicates the total number of blocked processes.	Number	The detailed diagnosis of this measure, if enabled, reveals information such as the ID of the blocked processes, the user executing each of the processes, the database on which every process is executing, the waiting time of the blocked process, etc.

5.1.5.3 Sybase Tasks Test

This test reports statistics pertaining to the tasks performed by the Sybase database engine.

Purpose	Reports statistics pertaining to the tasks performed by the Sybase database engine
Target of the test	A Sybase adaptive server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – A Sybase user who has the “sa_role”.
5. PASSWORD – The password corresponding to the above user
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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every Sybase server being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Connections opened:
The rate at which connections are opened</td><td>Conns/Sec</td><td>This provides a general understanding of the Adaptive Server environment and the work load during the measure interval.</td></tr>
<tr><td>Task switches:
The rate at which the Adaptive Server switched context from one user task to another</td><td>Switches/Sec</td><td>The detailed diagnosis of this measure, if enabled, reports the percentage of times the context switch was due to each cause.</td></tr>
</table>

5.1.5.4 Sybase Transaction Log Test

This test reports statistics pertaining to the transaction logs of a Sybase database server.

Purpose	Reports statistics pertaining to the transaction logs of a Sybase database server
Target of the test	A Sybase adaptive server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – A Sybase user who has the "sa_role".
5. PASSWORD – The password corresponding to the above user
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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every Sybase server being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>ULC log writes:
The rate at which log records per transaction are written into the ULCs</td><td>Records/Sec</td><td>If this data is unusual, study the Max ULC size measure and look at your application for long-running transactions and for transactions that modify a large number of rows.</td></tr>
<tr><td>Waits for ULC log writes:
The rate at which attempts made to write to ULCs encountered semaphore contention</td><td>Waits/Sec</td><td></td></tr>
<tr><td></td><td>ULC flushes:
The rate at which user log caches (ULCs) were flushed to a transaction log</td><td>Flushes/Sec</td><td>The detailed diagnosis for this measure provides the percentage of ULC flushes due to full ULC, end transaction, change of database, system log record and other reasons.</td></tr>
</table>

	Waits for ULC flushes: The rate at which attempts made to flush ULC pages to the log encountered log semaphore contention	Waits/Sec	If this value is high, reduce the semaphore contention by: • Increasing the ULC size, if filling user log caches is a frequent cause of user log cache flushes • Reducing log activity through transaction redesign. Aim for more batching with less frequent commits. Be sure to monitor lock contention as part of the transaction redesign. • Reducing the number of multidatabase transactions, since each change of database context requires a log write. • Dividing the database into more than one database so that there are multiple logs. If you choose this solution, divide the database in such a way that multidatabase transactions are minimized.
	Max ULC size: The maximum number of bytes used in any ULCs, across all ULCs. This data can help you determine if ULC size is correctly configured.	KB	If this value is consistently less than the defined value for the **user log cache size** configuration parameter, reduce the user log cache size to *Max ULC size* value.
	Log allocations: The rate at which additional pages were allocated to the transaction log	Allocations/Sec	This data is useful for comparing to other data in this test and for tracking the rate of transaction log growth.
	Transaction log writes: The rate at which the Adaptive server wrote a transaction log page to disk	Pages/Sec	
	Avg writes per log page: The average number of times each log page was written to disk	Writes/Page	In high throughput applications, this number should be as low as possible.

5.1.5.5 Sybase Transactions Test

This test reports the transaction statistics of a Sybase database server.

Purpose	Reports the transaction statistics of a Sybase database server		
Target of the test	A Sybase adaptive server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – A Sybase user who has the "sa_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for every Sybase server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Committed transactions: The rate of transaction commits	Trans/Sec	
	Rolledback transactions: The rate of transactions rollbacks	Trans/Sec	Rollbacks are costly operations on the database, and hence, will have to be kept at the minimum.
	Inserts: The rate at which records are inserted into the database	Inserts/Sec	
	Updates: The rate at which records in the database are updated	Updates/Sec	
	Deletes: The rate at which records are deleted from the database	Deletes/Sec	

5.1.5.6 Sybase Blockers Test

One common problem encountered with databases is blocking. Suppose that process A is modifying data that process B wants to use. Process B will be blocked until process A has completed what it is doing. This is only one type of blocking situation; others exist and are common. What matters to a database administrator is identifying when blocking is a problem and how to deal with it effectively. When blocking is bad enough, users will notice slowdowns and complain about it. With a large number of users, it is common for tens or hundreds of processes to be blocked when slowdowns are noticed. Killing these processes may or may not solve the problem because 10 processes may be blocked by process B, while process B itself is blocked by process A. Issuing 10 kill statements for the processes blocked by B probably will not help, as new processes will simply become blocked by B. Killing process B may or may not help, because then the next process that was blocked by B, which is given execution time, may get blocked by process A and become the process that is blocking the other 9 remaining processes. When you have lots of blocking that is not resolving in a reasonable amount of time you need to identify the root blocker, or the process at the top of the tree of blocked processes. Imagine again that you have 10 processes blocked by process B, and process B is blocked by process A. If A is not blocked by anything, but is itself responsible for lots of blocking (B and the 10 processes waiting on B), then A would be the root blocker. (Think of it as a traffic jam.) Killing A (via kill) is likely to unblock B, and once B completes, the 10 processes waiting on B are also likely to complete successfully. The SybaseBlockers test monitors the number of root blocker processes in a database.

Purpose	Reports the number of root blocker processes in a database		
Target of the test	A Sybase adaptive server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – A Sybase user who has the "sa_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.		
Outputs of the test	One set of results for every Sybase server being monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Number of root blockers:** Indicates the number of root blocker processes.	Number	Usually, the number of root blocker processes should be low. If this value increases suddenly, this is a cause for concern. Likewise, if a root-blocker process has been blocking other processes for a long time, it is a reason for further investigation. The detailed diagnosis for this test, if enabled, provides details of the root blocker processes - their SPIDs, programs running these processes, and the queries being issued by these processes. It is usually the case that killing any root-blocker process that has been running for a long while will get the database running well again.

5.1.5.7 Sybase Applications Test

Sometimes the database performs poorly due, not to blocking, but to particularly heavy loads. Often the DBA will determine that the database simply cannot support the work that it is being asked to do and maintain adequate performance. This does not necessarily mean it is time to create more indexes or throw more hardware at the problem. One cannot always assume that periods of high utilization represent legitimate work. There could be problems in the applications that are running, or even problems caused by the user. Maybe the application has a data paging functionality, but the user has opted to receive the entire 100,000 row DataSet every time, even though she/he has applied a sort which gives her the one row she needs first with each query. Regardless, it is important to identify performance issues and eliminate them. The SybaseApps test identifies which program has more connections open to (i.e., processes running in) the Sybase database. Simply looking at the CPU cycles taken up by a process will not indicate which of these processes has been most active recently. For example, the Sybase server internal processes may have been running for days and will probably always show up as the processes that have taken the most CPU time since the database booted up. Hence, it is more helpful to find processes that have used lots of CPU for the majority of the time that they have been connected. This value represents how "expensive" a process is with respect to the Sybase database server. For each program that connects to the database server, the SybAppsTest reports the total CPU cycles for each second that the program is connected to the database. This value, represented by the *CPU cycles rate* measure, is an aggregate of all the CPU cycles consumed by every instance of the program while it is connected to the database server. The *Avg CPU cycles rate* measure represents the *CPU cycles rate* averaged across the number of processes in the database for the program under consideration. The *Avg CPU cycles rate* quantifies how bad a program is compared to the others, by dividing the *CPU cycles rate* by the number of connected instances. A high value for this value would indicate that every instance of the program was CPU-intensive. A lower value would indicate that the program may have some instances that cause performance problems, but also has instances that are mostly idle.

Purpose	Identifies which program has more connections open to (i.e., processes running in) the Sybase database
Target of the test	A Sybase adaptive server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – A Sybase user who has the "sa_role".
5. PASSWORD – The password corresponding to the above user
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. 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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every program/application executing on the Sybase server being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Number of processes:
Indicates the number of database processes associated with a specific program.</td><td>Number</td><td>A comparison of this value across programs will indicate which program is initiating most connections to the database. Comparison of this value over time can provide indications of potential changes in database activity characteristics of a program.</td></tr>
<tr><td>CPU cycles rate:
Indicates the number of CPU cycles consumed by all processes of a program, per minute of login.</td><td>Cycles/Sec</td><td>The higher the value, the more CPU resources that the program is taking in the database. The detailed diagnosis for this test provides details of the most expensive queries to the database - i.e., what host is a program running from, who is running it, and what application is running it, which database the program is accessing, etc.</td></tr>
<tr><td></td><td>Avg CPU cycles rate:
Indicates the number of CPU cycles consumed by a process of a program, per minute of login.</td><td>CyclesRate/Conn</td><td>This value is the ratio of the CPU cycles rate to the number of processes for a program.</td></tr>
</table>

5.2 Monitoring the Sybase Server Using the MDA Tables

Starting Sybase ASE 12.5.3, users have the option of installing MDA tables on the Sybase server for accessing critical performance statistics pertaining to the server. If MDA tables are installed on a Sybase server, then, you can use the *Sybase ASE 15* monitoring model (see Figure 5.9) offered by eG Enterprise to monitor that Sybase server.

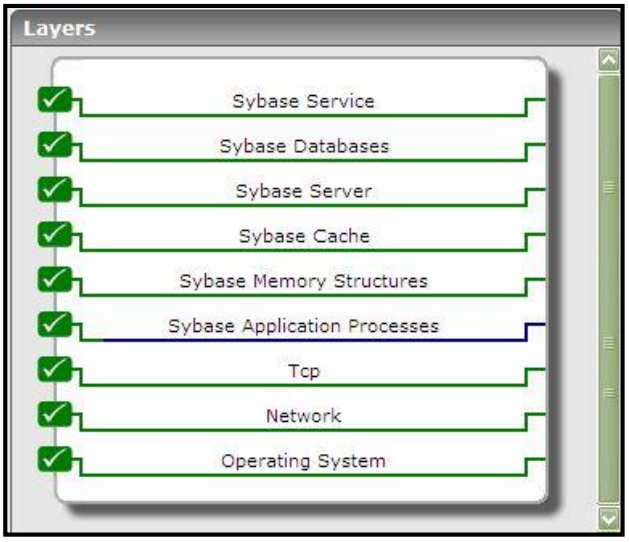


Figure 5.9: The Sybase ASE 15 monitoring model

To use the monitoring model depicted by Figure 5.9 , the following pre-requisites need to be fulfilled:

- MDA tables should be installed on the Sybase server to be monitored;
- The eG agent should be configured with the user privileges required for monitoring. These privileges may differ across tests. While most tests need to be configured with the privileges of a user who has the "mon_role", the **Sybase Database Space Usage** test and the **Sybase Database Stats** test require the privileges of a user with the "mon_role" and "sa_role", and the **Sybase System Processes** test requires the "mon_role" and "sybase_ts_role". The test-specific documentation will shed more light on this. However, for best results, its recommended that you configure all tests with the credentials of a user with all three roles - "mon_role", "sa_role", and "sybase_ts_role".
- The following configuration parameters have to be enabled on the target Sybase server:
    - *enable monitoring*
    - *sql text pipe active*
    - *sql text pipe max messages*
    - *statement pipe active*
    - *statement pipe max messages*
    - *errorlog pipe active*
    - *errorlog pipe max messages*
    - *deadlock pipe active*

- *deadlock pipe max messages*
- *wait event timing*
- *process wait events*
- *object lockwait timing*
- *SQL back capture*
- *statement statistics active*
- *per obect statistics active*
- *enable stmt cache monitoring*
- *max SQL text monitored*
- *performance monitoring option*

Once the above pre-requisites are fulfilled, then the tests mapped to each layer of Figure 5.9 will periodically run queries on the MDA tables to report a wealth of performance information pertaining to the Sybase server. Using these metrics, administrators can find quick and easy answers to the following performance queries:

- Is the Sybase server available? If so, is it responding quickly to client requests?
- Has the error log captured any errors recently? If so, what are they?
- Did too many wait events occur on the Sybase server during the last measurement period?If so, what type of events are these? How long did these events wait?
- Did any deadlocks occur?
- Are too many locks being held? Which user and process initiated these locks?
- Have locks been held for an unusually long time?
- Are the data and procedure caches being used optimally?
- Have the caches been sized correctly?
- Is the server able to process I/O requests quickly, or are any bottlenecks visible?
- Is the Sybase engine using up CPU excessively?
- Is the traffic to and from the server high?
- Are there any sleeping, blocked, or infected processes on the server? If so, what are they?
- Is any database on the server using memory and space excessively? If so, which one is it?
- Have any queries been running on the server for a very long time? If so, what are the queries?
- How is the user load on the server?
- Have any users initiated sleeping or infected processes on the server? What are these processes?

The sections that follow will discuss each of the first 6 layers of Figure 5.9, elaborately.

5.2.1 Sybase Application Processes Layer

The tests mapped to this layer proactively alerts administrators to the following:

- Non-availability of the Sybase port;
- Non-availability of the processes that are crucial to the smooth functioning of the Sybase server;
- Excessive resource usage by one/more of these critical processes;
- Errors recently encountered by the server;

Figure 5.10: The tests mapped to the Sybase Application Processes layer

Since the **TcpPort** and **Processes** tests have already been deal with in the *Monitoring Unix and Windows Servers* document, the section that will follow will handle the **Sybase ErrorLog** test only.

5.2.1.1 Sybase Errors Test

This test periodically monitors the Sybase server's error logs to promptly capture the errors that occur on the server and report the number and severity of the errors. For execution, this test requires the **enable monitoring**, **errorlog pipe max messages**, and **errorlog pipe active** configuration parameters to be enabled.

Purpose	Periodically monitors the Sybase server's error logs to promptly capture the errors that occur on the server and report the number and severity of the errors.
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role".
5. PASSWORD – The password corresponding to the above user
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. 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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every database being monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Error count severity 10-16:
The number of errors with severity codes between 10 and 16, which occurred on the Sybase server during the last measurement period.</td><td>Number</td><td>Error messages with severity levels 10-16 are generated by problems that are caused by user errors. These problems can always be corrected by the user.
To view the list of problems in this category, use the detailed diagnosis of this measure.</td></tr>
</table>

	Error count severity 17-18 : The number of errors with severity codes between 17 and 18, which occurred on the Sybase server during the last measurement period.	Number	Error messages with severity level 17 mean that the command has caused Adaptive Server to run out of resources or to exceed some limit set by the System Administrator. You can continue with the work you are doing, although you may not be able to execute a particular command. The Database Owner can correct the level 17 error messages indicating that you have run out of space. Other level 17 error messages should be corrected by the System Administrator. Error messages with severity level 18 indicate some kind of internal software bug. However, the command runs to completion, and the connection to Adaptive Server is maintained. You can continue with the work you are doing, although you may not be able to execute a particular command. An example of a situation that generates severity level 18 is Adaptive Server detecting that a decision about the access path for a particular query has been made without a valid reason. Since problems that generate such messages do not keep users from their work, users tend not to report them. Users should be instructed to inform the System Administrator every time an error message with these severity levels (or higher) occur so that the System Administrator can report them.

	Error count severity 19-26: The number of errors with severity codes between 19 and 26, which occurred on the Sybase server during the last measurement period.	Number	**Level 19: Adaptive Server fatal error in resource** Error messages with severity level 19 indicate that some non-configurable internal limit has been exceeded and that Adaptive Server cannot recover gracefully. You must reconnect to Adaptive Server. **Level 20: Adaptive Server fatal error in current process** Error messages with severity level 20 indicate that Adaptive Server has encountered a bug in a command. The problem has affected only the current process, and it is unlikely that the database itself has been damaged. Run dbcc diagnostics. You must reconnect to Adaptive Server. **Level 21: Adaptive Server fatal error in database processes** Error messages with severity level 21 indicate that Adaptive Server has encountered a bug that affects all the processes in the current database. However, it is unlikely that the database itself has been damaged. Restart Adaptive Server and run the dbcc diagnostics. You must reconnect to Adaptive Server. **Level 22: Adaptive Server fatal error: Table integrity suspect** Error messages with severity level 22 indicate that the table or index specified in the message was previously damaged by a software or hardware problem. The first step is to restart Adaptive Server and run dbcc to determine whether other objects in the database are also damaged. Whatever the report from dbcc may be, it is possible that the problem is in the cache only and not on the disk itself. If so, restarting Adaptive Server will fix the problem. If restarting does not help, then the problem is on the disk as well. Sometimes, the problem can be solved by dropping the object specified in the error message. For example, if the message tells you that Adaptive Server has found a row with length 0 in a nonclustered index, the table owner can drop the index and re-create it. Adaptive Server takes any pages or indexes offline that it finds to be suspect during recovery. Use sp_setsuspect_granularity to determine whether recovery marks an entire database or only individual pages as suspect. See

			sp_setsuspect_granularity in the *Reference Manual* for more information. You must reconnect to Adaptive Server. Level 24: Hardware error or system table corruption Error messages with severity level 24 reflect some kind of media failure or (in rare cases) the corruption of *sysusages*. The System Administrator may have to reload the database. It may be necessary to call your hardware vendor. Level 23: Fatal error: Database integrity suspect Error messages with severity level 23 indicate that the integrity of the entire database is suspect due to previous damage caused by a software or hardware problem. Restart Adaptive Server and run dbcc diagnostics. Even when a level 23 error indicates that the entire database is suspect, the damage may be confined to the cache, and the disk itself may be fine. If so, restarting Adaptive Server with startserver will fix the problem. Level 25: Adaptive Server internal error Level 25 errors are not displayed to the user; this level is only used for Adaptive Server internal errors. Level 26: Rule error Error messages with severity level 26 reflect that an internal locking or synchronization rule was broken. You must shut down and restart Adaptive Server.
	Total error count: The total number of errors that occurred on the Sybase server during the last measurement period.	Number	Ideally, this value should be 0.

5.2.2 Sybase Memory Structures Layer

This layer monitors wait events and the locking activity on the database server to identify areas where the server has been wasting resources. Using the tests mapped to this layer, administrators can quickly isolate the following:

- Wait events on which the server has spent the maximum time; who and which process initiated such an event;
- Deadlock situations;
- Locks that have been held for too long a time; who and which process initiated the lock, and which object held the lock for how long.

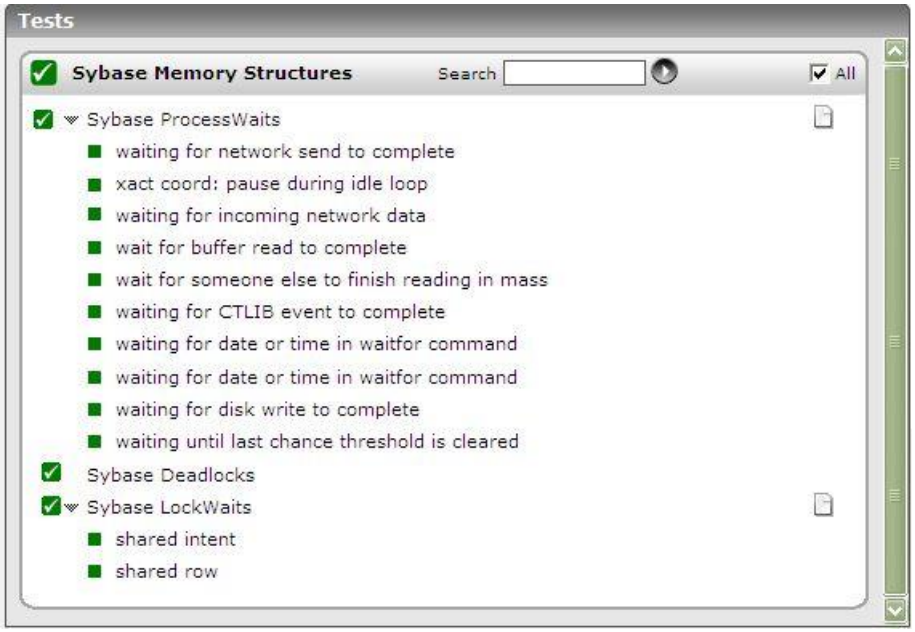


Figure 5.11: The tests associated with the Sybase Memory Structures layer

5.2.2.1 Sybase Process Waits Test

A wait event occurs when a server process suspends itself, sleeps, and waits for another event to wake it. The Sybase ASE includes unique wait event IDs for each of these wait events.

The test monitors each type of wait events on the Sybase database server and reports key performance statistics pertaining to every event type. For execution, this test requires the **enable monitoring** and **process wait events** configuration parameters to be enabled.

Purpose	Monitors each type of wait events on the Sybase database server and reports key performance statistics pertaining to every event type
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role".
5. PASSWORD – The password corresponding to the above user
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. 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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every wait event type being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Number of waits:
The number of events of this wait event type during the last measurement period.</td><td>Number</td><td>Use the detailed diagnosis of this measure to view the complete details of the wait events and to figure out who and which process initiated the event.</td></tr>
<tr><td>Avg wait time:
The average wait time of events of this type.</td><td>MillSec</td><td>If this measure value is high for a particular event type when the value of the Number of waits measure is low, it could indicate that one/more events of that type have been waiting for a long time. The DBA might have to ensure that such wait events are minimal.</td></tr>
</table>

The detailed diagnosis of the *Number of waits* measure will reveal the description of the wait event, the user and the process that initiated the wait event, the total number of waits, and the total time waited.

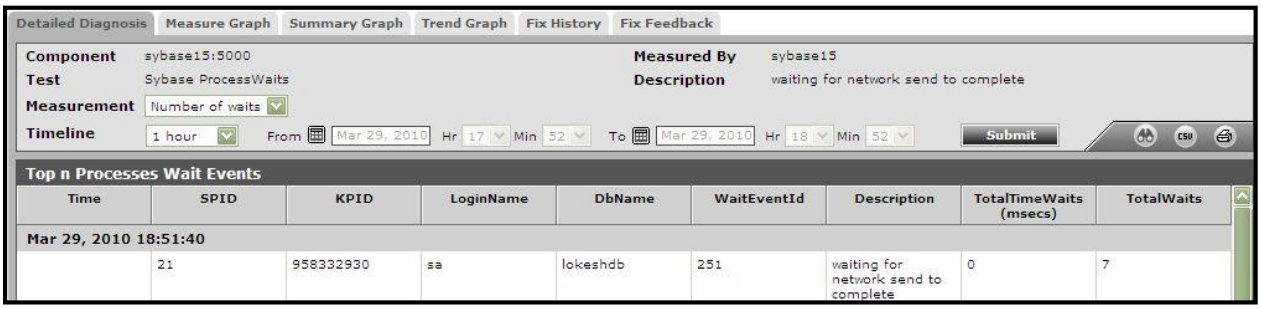


Figure 5.12: The detailed diagnosis of the Number of waits measure

5.2.2.2 Sybase Deadlocks Test

This test monitors the locking activity on the Sybase server, and reports the number of deadlocks. For this test to work, the following configuration parameters need to be enabled on the server:

- enable monitoring
- deadlock pipe max messages
- deadlock pipe active

Purpose	Monitors the locking activity on the Sybase server, and reports the number of deadlocks
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.		
Outputs of the test	One set of results for every database being monitored		
	Number of deadlocks: The number of deadlocks found in the Sybase Server.	Number	A deadlock may arise due to various situations including bad design of queries and deficient coding practices. A deadlock is a situation where both/all the lock requestors are in a mutual or a multi-way tie. Any deadlocks are detrimental to database application performance. The detailed diagnosis for this measure provides a listing of dead locks for each lock type.

5.2.2.3 Sybase Lock Waits Test

For every lock type auto-discovered, this test reports the total number of locks currently held, and the average wait time for locks. For execution, this test requires the **enable monitoring** and **wait event timing** configuration parameters to be enabled.

Purpose	For every lock type auto-discovered, this test reports the total number of locks currently held, and the average wait time for locks
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed

Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.
Outputs of the test	One set of results for every lock type being monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Number of locks: The number of locks of this type that are currently held.	Number	Ideally, both these values should be low. A large number of locks or a high wait time for locks can result in lock contention and can adversely impact server performance. You can use the detailed diagnosis of the *Number of locks* measure to view the complete lock details and to determine the process and user responsible for the lock. Furthermore, you can use the following locking guidelines to reduce lock contention and speed performance: • Use the lowest level of locking required by each application. Use isolation level 2 or 3 only when necessary. Updates by other transactions may be delayed until a transaction using isolation level 3 releases any of its shared locks at the end of the transaction. Use isolation level 3 only when nonrepeatable reads or phantoms may interfere with your desired results. If only a few queries require level 3, use the **holdlock** keyword or **at isolation serializing** clause in those queries instead of using **set transaction isolation level 3** for the entire transaction.

			If most queries in the transaction require level 3, use **set transaction isolation level 3**, but use **noholdlock** or **at isolation read committed** in the remaining queries that can execute at isolation level 1. If you need to perform mass inserts, updates, or deletes on active tables, you can reduce blocking by performing the operation inside a stored procedure using a cursor, with frequent commits. • If your application needs to return a row, provide for user interaction, and then update the row, consider using timestamps and the **tsequal** function rather than **holdlock**. • If you are using third-party software, check the locking model in applications carefully for concurrency problems. Also, other tuning efforts can help reduce lock contention. For example, if a process holds locks on a page, and must perform a physical I/O to read an additional page, it holds the lock much longer than it would have if the additional page had already been in cache. Better cache utilization or using large I/O can reduce lock contention in this case. Other tuning efforts that can pay off in reduced lock contention are improved indexing and good distribution of physical I/O across disks. Moreover, you can configure a server-wide lock-wait limit with the configuration parameter **lock wait period.** This can reduce lock wait time considerably.
	Avg wait time sec: The average wait time of locks of this type.	Secs	

The detailed diagnosis of the *Number of locks* measure provides the complete lock details including the user and the process that initiated the lock, the object that has been locked, the lock type, the lock level, the lock wait time, and the exact page number and row number that has been locked. This information enables administrators to effectively troubleshoot lock-related issues, accurately identify what caused the lock, and optimize the process code/query so that, the lock is released.

List all the LockWaits Details

Time	SPID	KPID	UserName	DbName	LockId	ObjectId	Object	LockState	LockType	LockLevel	WaitTime (secs)	PageNumber	RowNumber
Mar 29, 2010 18:43:00													
	14	1048592	_	master	28	24	sysstatistics	Granted	shared intent	TABLE	0	-	-

Figure 5.13: The detailed diagnosis of the Number of locks measure

5.2.2.4 Sybase Spinlocks Test

Spinlocks are lightweight synchronization primitives which are used to protect access to shared resources. The spinlocks can only be updated automically when the resource is accessed to perform a user task. The spinlock denies all other tasks access to the resource until the changes are made by the current user task as a result other user tasks are made to wait, this in turn causes spinlock contention. Although, the spinlocks are held for extremely brief durations, they can increase CPU resource utilization and reduce performance of the Sybase server with high transaction rates. Therefore, the spinlocks should be monitored to avoid such excess CPU utilization and performance lag of the Sybase server. The Sybase Spinlocks test helps administrators greatly in this exercise.

This test auto-discovers the named spinlocks and reports the percentage of spinlock contention on the resources, CPU cycle spinning activity, and grabs and waits of each spinlock.

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port through which the server communicates. By default, it is *1433*. 4. **USER** – To enable the eG agent to connect to the Sybase server and collect the required metrics, specify the credential of the Sybase user who has the "mon_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for each spinlock being monitored		
Measurements made by the test	**Spinlock contention:** Indicates the number of times an engine encountered this spinlock contention on the resource, and had to wait as a percentage of the total spinlock requests for that resource.	Percent	The value of this measure should not exceed 10%. For example, if the spinlock contention is more than 10% in the cache, consider using named caches or adding cache partitions. The number of cache partitions is always a power of 2 such that you can approximately reduce half of the spinlock contention when you increase the number of partitions each time. The detailed diagnosis of this measure, if enabled, provides details such as SpinlockslotID, Ownerpid, Lastownerpid, Spid, login name, Status, etc.
	Grabs rate: Indicates the number of grabs for this spinlock per second.	Grabs/Sec	

	Spins rate: Indicates the number of CPU spins that are made per second to acquire this spinlock.	Spins/Sec	Ideally, the value of this measure should be low. A signifigance increase in the value indicates high CPU utilization, which results high performance lag.
	Waits rate: Indicates the number of waits that are occured per second for this spinlock.	Waits/Sec	If the value of this measure is consistently high, identify which resource causing waits and fine tune the appropriate SQL queries.

5.2.3 Sybase Cache Layer

Use the tests associated with this layer to measure the effectiveness of your caches, check for sizing inadequacies, and take relevant optimization steps.

Figure 5.14: The tests associated with the Sybase Cache layer

5.2.3.1 Sybase Data Cache Test

Data caches in ASE hold the data, index, and log pages currently in use, as well as recently used pages in memory. These caches are critical in providing high levels of system performance as they effectively avoid expensive disk I/O. Inadequately sized data caches can increase the incidence of physical disk accesses, thereby affecting the performance of the Sybase server. Periodic monitoring of the usage of the data cache is essential to determine whether the cache needs to be resized or not.

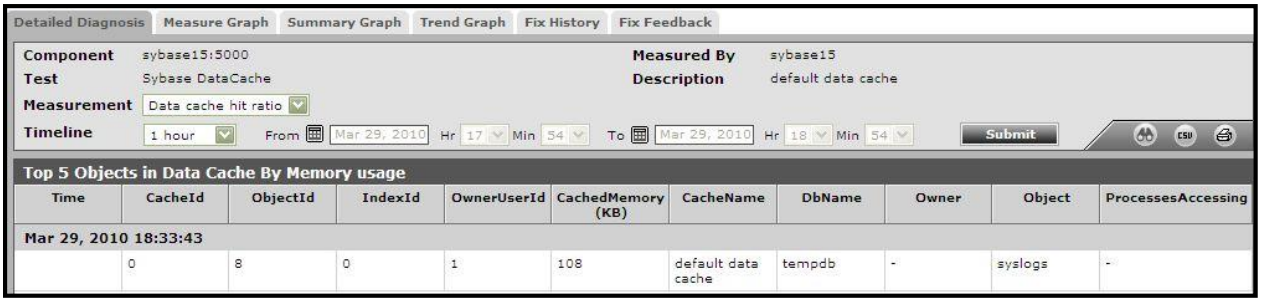
This test auto-discovers the data caches on the Sybase server, and tracks the usage of each cache. Frequently used objects and objects that occupy the memory are listed in detailed diagnosis. For this test to work, make sure that the **enable monitoring** parameter is enabled.

Purpose	To measure the statistics of each datacache available in a Sybase server
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.		
Outputs of the test	One set of results for every database being monitored		
Measurements made by the	**Measurement**	**Measurement Unit**	**Interpretation**

	Data cache hit ratio: The hit ratio of each data cache in the Sybase server.	Pct	If the value of this measure is low, it indicates that too many physical reads are occurring on the Sybase server. If the problem persists, it could result in a serious contention for disk resources, which in turn, can cause the performance of the Sybase server to significantly deteriorate. You might want to consider resizing the data cache to avoid this adversity. The detailed diagnosis for this measure provides a list of objects that occupy the memory in cache, so that we can easily identify the objects that are consuming too much memory, and accordingly initiate corrective actions.
	Number of objects: The number of objects in each cache.	Number	Use the detailed diagnosis of this measure to view the list of objects in cache; this reveals the objects that are frequently accessed by users.

The detailed diagnosis of the *Data cache hit ratio* measure reveals the objects that are in cache and the memory space occupied by each object. This way, you can quickly identify which objects are occupying the maximum space in the cache, check whether these objects are been used frequently, and if not, remove the objects from the cache, so as to free space.



5.2.3.2 Sybase Procedure Cache Test

Adaptive Server maintains an MRU/LRU (most recently used/least recently used) chain of stored procedure query plans. As users execute stored procedures, Adaptive Server looks in the procedure cache for a query plan to use. If a query plan is available, it is placed on the MRU end of the chain, and execution begins. If no plan is in memory, or if all copies are in use, the query tree for the procedure is read from the *sysprocedures* table. It is then optimized, using the parameters provided to the procedure, and put on the MRU end of the chain, and execution begins. Plans at the LRU end of the page chain that are not in use are aged out of the cache.

The memory allocated for the procedure cache holds the optimized query plans (and occasionally trees) for all batches, including any triggers. If adequate memory is not available to the cache, then cache misses will increase, thereby causing direct accesses to database tables to rise; if this trend continues, then the performance of the database server will suffer.

By periodically monitoring the usage of the procedure cache, administrators can promptly detect insufficient memory allocations to the cache; based on the findings, they can resize the cache to ensure peak performance of the database server.

This test reports whether/not the procedure cache has been utilized optimally. For execution, the test requires the **enable monitoring** configuration parameter to be enabled.

Purpose	Reports whether/not the procedure cache has been utilized optimally		
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **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 enabled/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.		
Outputs of the test	One set of results for every database being monitored		
Measurements made by the	**Measurement**	**Measurement Unit**	**Interpretation**

	Procedure hit ratio: The percentage of time a stored procedure query plan required by a user was available in the procedure cache.	Percent	Ideally, the value of this measure should be high. A low value indicates that many stored procedure query plan that users required were, more often than not, unavailable in the procedure cache. To ensure that accesses to physical database tables do not increase as a result, check the most accessed procedure and place it in named cache. A low cache hit rate could also indicate a badly sized cache. Use the detailed diagnosis of this test to view the top-5 procedures in the cache in terms of memory usage. If need be, you can resize the cache using the following formula: **max # of concurrent users) * (4 + size of largest plan) * 1.25**

5.2.4 Sybase Server Layer

The tests mapped to this layer enable administrators to do the following:

- Detect I/O processing bottlenecks on the Sybase server;
- Ascertain how CPU-efficient the Sybase engine is;
- Monitor the traffic to and from the server so that, network slowdowns can be prompty isolated;
- Monitor the type and count of processes on the Sybase server so that, resource-intensive processes can be identified.

Figure 5.15: The tests mapped to the Sybase Server layer

5.2.4.1 Sybase Device I/O Test

This test monitors the I/O activity on each device on the Sybase server, and proactively alerts administrators to contention for device I/O semaphores. To ensure that this test runs smoothly, enable the **enable monitoring** configuration parameter on the Sybase server.

Purpose	Monitors the I/O activity on each device on the Sybase server, and proactively alerts administrators to contention for device I/O semaphore		
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for every device being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Reads: The number of reads that occurred on the device during the last measurement period.	Number	

	APF reads: Indicates the number of APF reads that occurred on this device during the last measurement period.	Number	Asynchronous prefetch (APF) issues I/O requests for pages before the query needs them so that most pages are in cache by the time query processing needs to access the page. High APF activity can hence improve performance for: • Sequential scans, such as table scans, clustered index scans, and covered nonclustered index scans • Access via nonclustered indexes • Some **dbcc** checks and **update statistics** • Recovery
	Writes: Indicates the number of writes that occurred on this device during the last measurement period.	Number	
	Request: The number of requests from this device during the last measurement period.	Number	
	IO waits: The number of I/O requests to this device that were waiting during the last measurement period.	Number	
	Total io time: The time taken to read from or write to this device during the last measurement period.	Msecs	A high value of this measure indicates that the device is taking too much time to process I/O requests. This could be owing to any of the following reasons: • An I/O overload on the device; • Disk fragmentation on the device; • Table fragmentation

	Percentage io granted: The percentage of IO granted.	Percent	Ideally, this value should be high. When Adaptive Server needs to perform a disk I/O, it gives the task to the semaphore for that device in order to acquire a block I/O structure. On SMP systems, multiple engines can try to post I/Os to the same device simultaneously. This creates contention for that semaphore, especially if there are hot devices or if the data is not well distributed across devices. A low value for this measure indicates that many I/O requests were waiting for the semaphore – in other words, it indicates a contention for the semaphore. One solution might be to redistribute the data on the physical devices.

5.2.4.2 Sybase Engine Test

This test measures the efficiency of the Sybase engines by monitoring the CPU usage of each engine. You will have to enable the **enable monitoring** configuration parameter to make this test work.

Purpose	Measures the efficiency of the Sybase engine by monitoring the CPU usage by the engine		
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for every engine being monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Percentage CPU busy:** The percentage of CPU used by this engine during the last measurement period, for processing queries.	Percent	A high value for this measure indicates excessive utilization of CPU. The CPU utilization may be high because a few queries are consuming a lot of CPU, or because there are too many queries contending for a limited resource. Check the currently running queries to see the exact cause of the problem. If one/more queries appear to be resource-intensive, you might have to fine-tune them to make them resource efficient.
	Percentage system busy: The percentage of CPU used by this engine during the last measurement period, for system-level processing.	Pct	An unusually high value indicates a problem and may be due to too many system-level tasks executing simultaneously.
	Percentage user busy: The percentage of CPU used by this engine during the last measurement period, for processing user requests.	Pct	A high value for this measure indicates that one/more user transactions are consuming too much CPU. You can take the help of the detailed diagnosis information provided by the *Avg CPU time* measure reported by the **Sybase Statement** test to identify the user queries that are consuming CPU resources excessively. Once identified, you might have to fine-tune the application that initiated the query.
	Percentage idle: The percentage of time for which the CPU was idle during the last measurement period.	Pct	

5.2.4.3 Sybase Network I/O Test

This test tracks the incoming and outgoing network traffic on the Sybase server. For execution, this test requires the **enable monitoring** configuration parameter to be enabled.

Purpose	Tracks the incoming and outgoing network traffic on the Sybase server
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed

Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for every server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Total traffic: The total number of IO packets handled by the server per second.	Packets/sec	
	Packets send rate: The number of packets sent from the Sybase Server per second.	Packets/sec	
	Packets received rate: The number of packets received by the Sybase server per second.	Packets/sec	
	Data sent: The number of bytes sent by the Sybase server per second.	Bytes/Sec	
	Data received: The number of bytes received by the Sybase server per second.	Bytes/Sec	

5.2.4.4 Sybase System Processes Test

This test reports details about the system processes running in a Sybase database server.

Purpose	Reports details about the system processes running in a Sybase database server		
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role" and "sybase_ts_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role". 5. **PASSWORD** – The password corresponding to the above user 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 enabled/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.		
Outputs of the test	One set of results for every Sybase server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Sybase processes: The total number of Sybase processes	Number	
	Background processes: The total number of background processes run by the Adaptive Server rather than by a user process	Number	The detailed diagnosis of this measure, if enabled, provides the details pertaining to the background processes currently executing.

	Running processes: The total number of running processes	Number	The detailed diagnosis of this measure, if enabled, provides details such as the ID of the running processes, the user executing each of the processes, the database on which every process is executing etc.
	Sleeping processes: The total number of sleeping processes	Number	The detailed diagnosis of this measure, if enabled, provides details such as the ID of the sleeping processes, the user executing each of the processes, the database on which every process is executing, the sleep status, sleep time etc.
	Infected processes: The total number of processes in which the server has detected a serious error condition	Number	The detailed diagnosis of this measure, if enabled, provides the ID of the infected processes, the user executing each of the processes, and the database on which every process is executing. This information enables the user to isolate the specific queries that are infected. Further analysis of these queries can be performed, in order to figure out the reason for the infection and take adequate measures to prevent it from recurring.
	Blocked processes: If a process attempts to access a resource that is already in use by another process, then such a process will be blocked until such time that the other process releases the resource. This measures indicates the total number of blocked processes.	Number	The detailed diagnosis of this measure, if enabled, reveals information such as the ID of the blocked processes, the user executing each of the processes, the database on which every process is executing, the waiting time of the blocked process, the command that initiated the block, etc.
	Remote processes: Indicates the number of processes that are processing I/O with a remote server.	Number	The detailed diagnosis of this measure, if enabled, reveals information such as the ID of the remote processes, the user executing each of the processes, the database on which every process is executing, resource usage per process, etc.
	Stopped processes: Indicates the number of processes that have stopped executing.	Number	The detailed diagnosis of this measure, if enabled, reveals information such as the ID of the stopped processes, the user executing each of the processes, the database on which every process is executing, resource usage per process, etc.

The detailed diagnosis of the *Blocked processes* measure not only reveals the IP of the process that has been blocked, but also provides useful information indicating the resource usage of each blocked process, along with the last query that was executed by the blocked process.

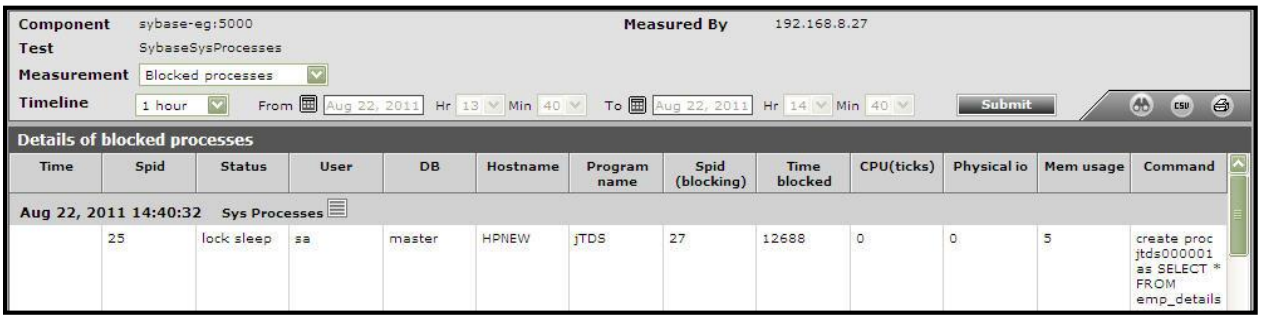


Figure 5.16: Detailed diagnosis of the Blocked processes reported by the SybaseSysProcesses test

For performing additional diagnosistics on the blocked process, click on the **Sys Processes** icon ( ) in Figure 5.16. Figure 5.17 will then appear, listing the **Process ID** of each blocked process. By default, the **Process ID** will be sorted in the ascending order of the ID itself; to change the sort order, pick a different option from the **Sort By** list. The **Process ID**s will then be listed in the chosen sort order. Selecting a **Process ID** from the list will reveal the following in the right panel of Figure 5.17: the current status of the chosen process, the database on which the process is executing, the duration for which it was blocked, and the CPU, I/O, and memory resources that were consumed by the process. In addition, the **LastCommand** executed by the process will also be displayed.

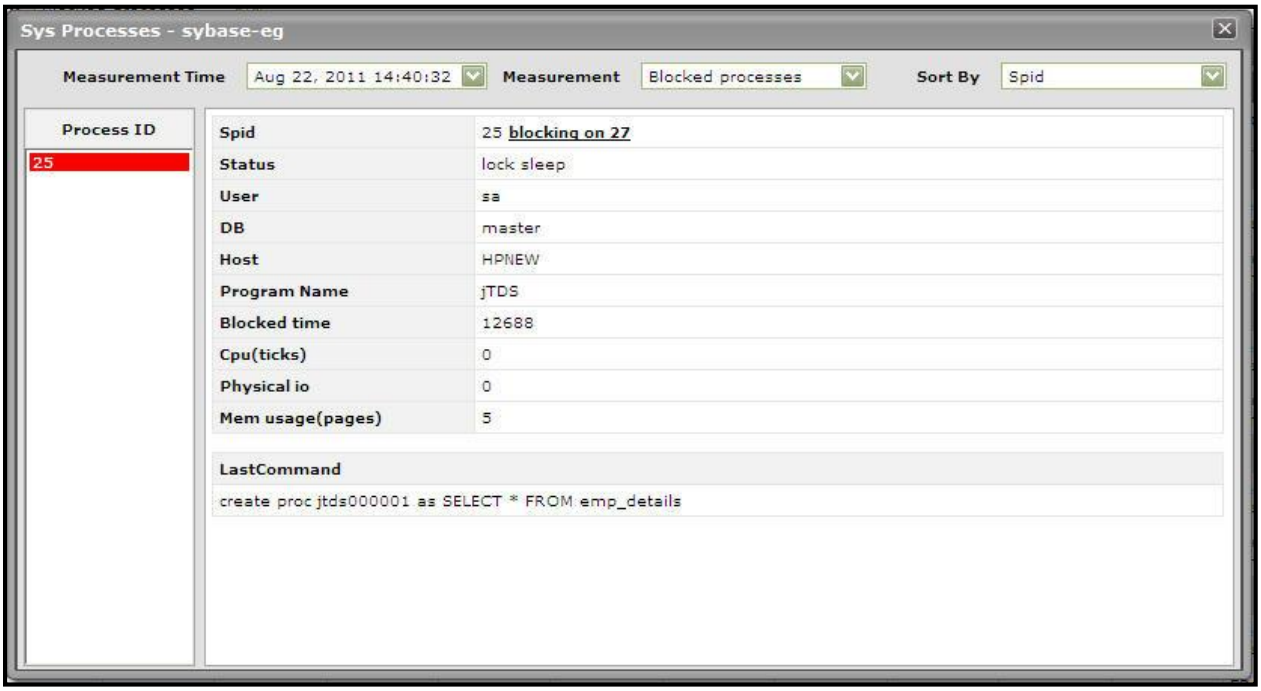


Figure 5.17: Analyzing blocked processes

Moreover, the **Spid** of the blocked process, as displayed in Figure 5.17, will be accompanied by a link to the ID of the process that is blocking it. Clicking on this link will reveal the complete details of the blocking process (see Figure 5.18), including the command executed by that process.

Figure 5.18: Details of the blocking process

By analyzing the query of the blocking process, you can determine whether deficiencies in query formulation caused this process to block the other process; if so, you can fine-tune the query to remove the block.

View the complete details of running processes by using the detailed diagnosis of the *Running processes* measure. Besides the process ID and the user who initiated the process, the detailed diagnosis will also reveal the resource usage (CPU, IO, Physical memory) of each process, thus bringing resource-intensive processes to your attention. The last command executed by the process will also be revealed. For further diagnosis, use the **Sys Processes** icon ( ) in Figure 5.20.

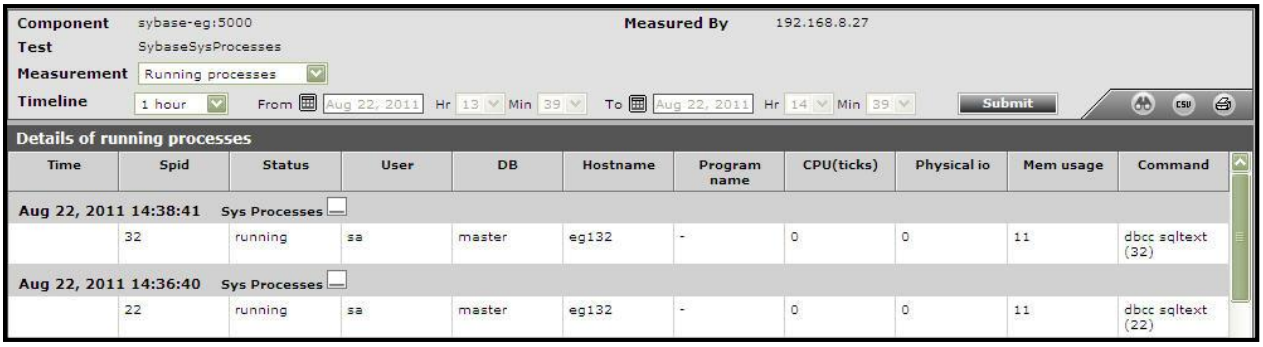


Figure 5.19: The detailed diagnosis of the Running processes measure reported by the SybaseSysProcesses test

To view the complete details of sleeping processes, use the detailed diagnosis of the *Sleeping processes* measure. In addition to process ID and user name, the detailed diagnosis also reveals the last command that was executed by the sleeping process, thereby pointing you to queries that may require optimization. For further diagnosis, use the **Sys Processes** icon ( ) in Figure 5.20.

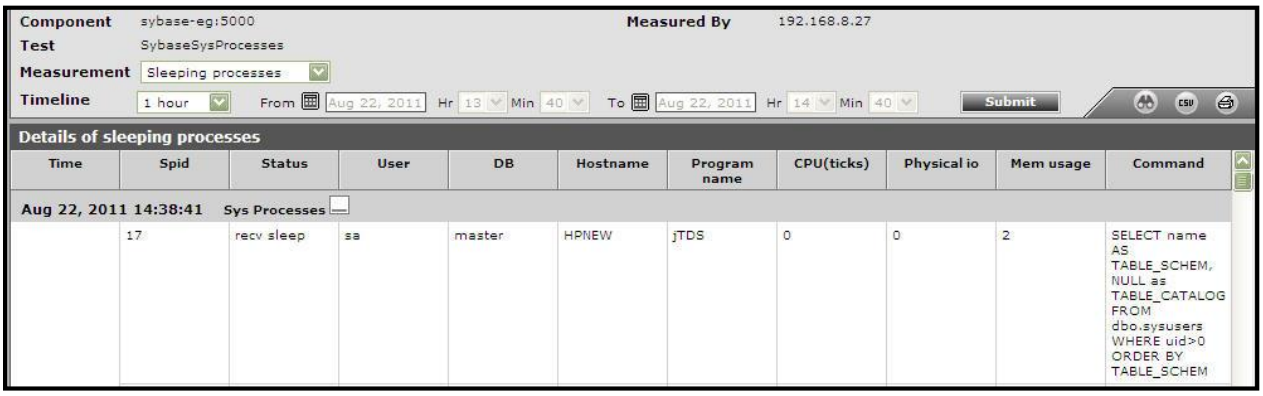


Figure 5.20: The detailed diagnosis of the Sleeping processes measure reported by the SybaseSysProcesses test

For viewing detailed metrics related to remote processes, use the detailed diagnosis of the *Remote processes* measure. In addition to process ID and user name, the detailed diagnosis also reveals the last command that was executed by the remote process, thus revealing the nature of interactions beween Sybase and the remote server. For further diagnosis, click on the icon in Figure 5.21.

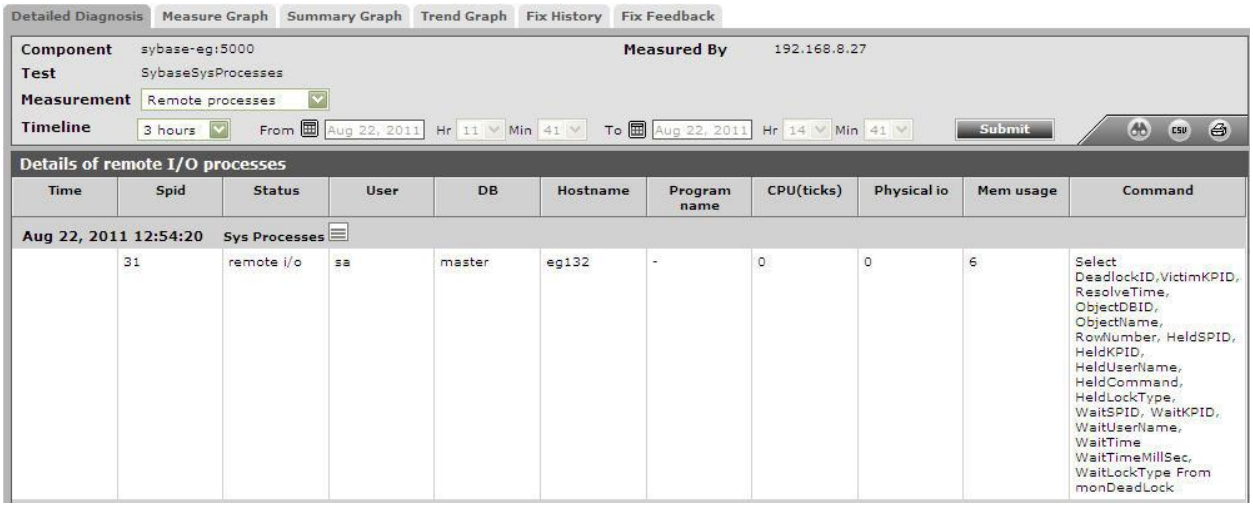


Figure 5.21: The detailed diagnosis of the Remote processes measure reported by the SybaseSysProcesses test

5.2.5 Sybase Databases Layer

This layer monitors each of the databases on the Sybase server. Using the tests mapped to this layer, you can promptly identify the following:

- Which database is consuming memory excessively?
- The transaction log on which database has run out of space?
- Which database is left with very little free space?

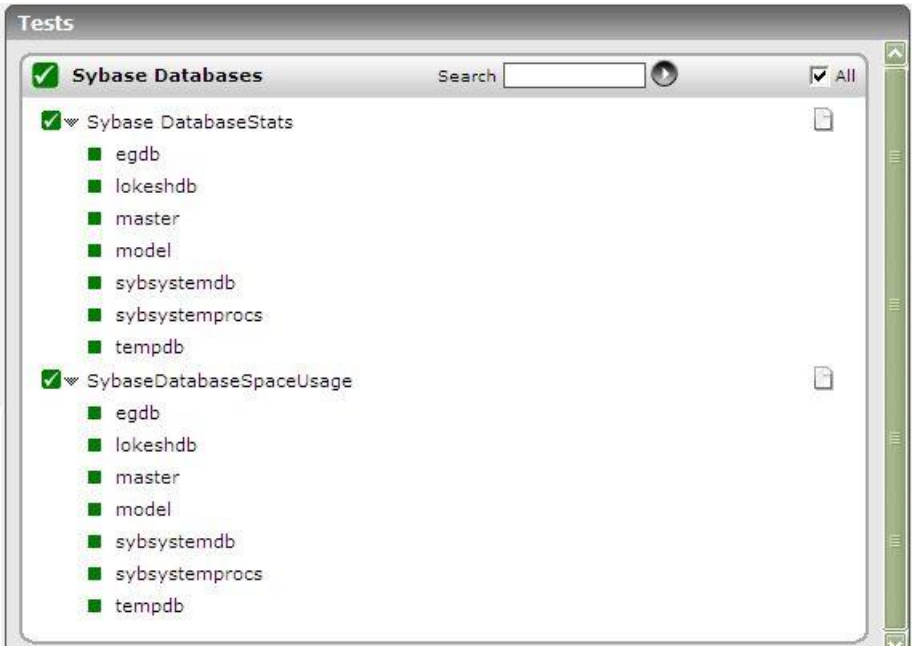


Figure 5.22: The tests mapped to the Sybase Databases layer

5.2.5.1 Sybase DatabaseStats Test

This test is used to track the memory usage, backup state, transaction log space requests, and wait statistics of each database. For this test to work, make sure that the **enable monitoring** parameter is enabled.

Purpose	To measure the statistics of each database
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role" and "sa_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role".
5. PASSWORD – The password corresponding to the above user
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. 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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every database being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Used memory:
The memory currently used by each database</td><td>KB</td><td>Ideally, the value of this measure should be low.
In the event of a sudden slowdown in the performance of the Sybase server, you might want to compare the memory usage of all databases on the server to identify which database is excessively consuming memory resources.</td></tr>
<tr><td>Last backup state:
Whether the last backup succeeded or failed.</td><td>Number</td><td>If the value of this measurement is '1', then it indicates that the last backup you attempted or scheduled failed. You would then have to investigate the reasons for the failure and rectify them.
If the last backup was successful, then this measure will return the value '0'.</td></tr>
<tr><td></td><td>Transaction log state:
The current status of transaction log gets.</td><td>Number</td><td>If this measurement is '1', it implies that the transaction log of that database has run out of space. When this occurs, you would have to allocate more space for that transaction log file; if not, database performance will suffer.</td></tr>
</table>

	Append log request: The number of times the Sybase server placed a request for log space for that database.	Number	
	Append log waits: The current number of log requests waiting to be processed.	Number	If most of the **Append log requests** are in waiting – i.e., if the value of this measure is dangerously close to that of the **Append log requests** measure- it could indicate a processing bottleneck on the server; this is typically caused by insufficient space in the transaction logs. In such a situation, it would be best to allocate more space to the transaction log, so that the request queue length reduces.

5.2.5.2 Sybase Database Space Usage Test

This test reports the space usage of all Sybase databases, and sheds light on those databases that are running short of space.

Purpose	Reports the space usage of all Sybase databases
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role" and "sa_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role".
5. PASSWORD – The password corresponding to the above user
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. INCLUDE DB - Specify a comma-separated list of databases that you wish to monitor in the INCLUDE DB text box. By default, this is set to all.
8. EXCLUDE DB - Specify a comma-separated list of databases that you wish to exclude from the scope of monitoring in the EXCLUDE DB text box. By default, this is set to none.
9. USE STORED PROCEDURE – By default, this flag is set to Yes, indicating that the test uses a stored procedure for collecting space usage metrics. To use this stored procedure, the eG agent requires sa_role privileges. In high security environments, where administrators may not prefer to expose the credential of a user with sa_role privilegs to the eG agent, you can configure the eG agent to run a query instead for pulling the required metrics. To enable the eG agent to use this query, set the USE STORED PROCEDURE flag to No.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every database being monitored</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total size:
The space allocated to a database</td><td>MB</td><td></td></tr>
<tr><td>Reserved space percent:
The percentage of space reserved for tables and indexes of a database</td><td>Percent</td><td>If the value of this measure reaches 100%, it indicates that the total space in the database has been completely allocated. New tables/indexes can be added to the database, only if its total size is increased.</td></tr>
<tr><td>Reserved space:
The amount of space allocated to the tables and indexes created on a database</td><td>MB</td><td>If the value of this measure becomes equal to that of the Total_size measure, new tables/indexes can no longer be created on the database. To create new tables, you must increase the database size.</td></tr>
<tr><td></td><td>Data space:
The amount of space used by data</td><td>MB</td><td></td></tr>
</table>

	Index space: The amount of space used by indexes	MB	
	Unused space: The amount of free space available in the database	MB	

5.2.5.3 Sybase Segments Test

A segment is a label that points to one or more database devices. Segment names are used in *create table and create index* commands to place tables or indexes on specific database devices. Using segments can improve Adaptive Server performance and give the System Administrator or Database Owner increased control over the placement, size, and space usage of database objects.

If a segment runs out of space, it may no longer be able to accommodate critical database objects such as tables/indexes, and will consequently, hamper the usage of the corresponding databases and adversely impact the performance of the underlying the Syvase server. If this is to be avoided, the space usage of the individual segments should be closely tracked, the segments experiencing a space contention should be rapidly isolated, and more space should be promptly allocated to such segments. This is where the **Sybase Segment Space Usage** test helps! This test monitors the space usage of each segment of each database on the Sybase server, and accurately points to those segments that may be running out of space.

Purpose	Reports the space usage of all Sybase databases
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Sybase server 3. **PORT** – The port on which the server is listening 4. **USER** – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role" and "sa_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role". 5. **PASSWORD** – The password corresponding to the above user 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **INCLUDE SEGMENTS** - Specify a comma-separated list of *database:segment* pairs that you want the test to monitor. By default, this parameter is set to *master:default*, indicating that the *default* segment on the *master* database is monitored.
Outputs of the test	One set of results for every *database:segment* configured for monitoring in the **INCLUDE SEGMENTS** text box

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Total size: Indicates the maximum size upto which this segment can grow.	GB	
	Used size: Indicates the space used by this segment.	GB	A low value is desired for this measure.
	Reserved space: Indicates the amount of space allocated to this segment.	GB	
	Free space: Indicates the amount of space that is not used by this segment	GB	A high value is desired for this measure.
	Percentage of used space: Indicates the percentage of space used by this segment.	Percent	This measure is the ratio of *Used space* over the *Total size* of a segment. A value close to 100% indicates excessive space usage by the segment. Compare the value of this measure across segments to identify that segment which has very little or no space left. You may want to allocate more space to that segment to prevent performance degradations.
	Percentage of free space: Indicates the percentage of free space for this segment.	Percent	This measure is the ratio of *Free space* over the *Total size* of a segment. A very low value for this measure indicates excessive space usage by the segment. Compare the value of this measure across segments to identify that segment which has very little or no space left. You may want to allocate more space to that segment to prevent performance degradations.

5.2.6 Sybase Service Layer

The tests mapped to this layer proactively alert administrators to the following:

- Non-availability of the Sybase server;
- Poor responsiveness of the Sybase server;
- Queries that have been running too long a time;
- Resource-intensive transactions to a database on the Sybase server;

- Infected processes and sleeping processes initiated by users to the Sybase server.

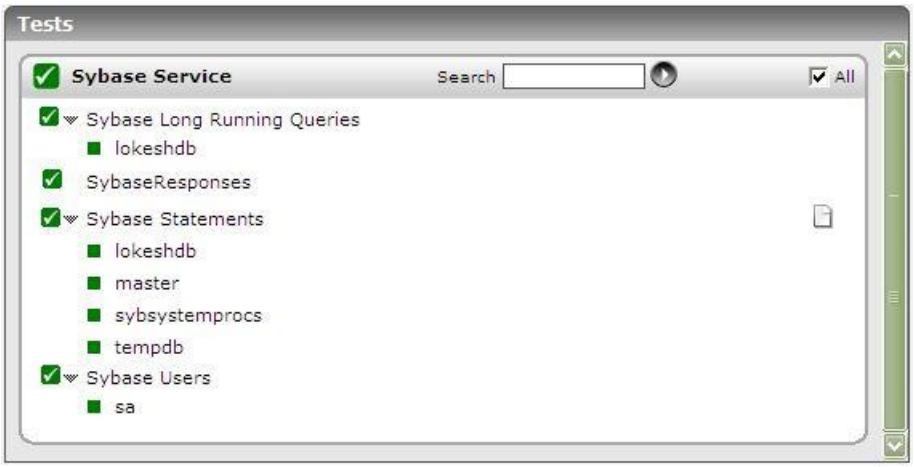


Figure 5.23: The tests mapped to the Sybase Service Layer

5.2.6.1 Sybase Long Running Queries Test

This test tracks the currently executing queries on the Sybase database server and determines the number of queries that have been running for a long time. For execution, this test requires the **enable monitoring**, **statement statistics active**, and **per object statistics active** configuration parameters to be enabled.

Purpose	To measure the long running queries on the Sybase server.
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed
Agent deploying the test	An internal agent

Configurable parameters for the test

1. **TEST PERIOD** – How often should the test be executed
2. **HOST** – The IP address of the Sybase server
3. **PORT** – The port on which the server is listening
4. **USER** – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role".
5. **PASSWORD** – The password corresponding to the above user
6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
7. **ELAPSED TIME** - In the **ELAPSED TIME** text box, specify the duration (in seconds) for which a query should have executed for it to be regarded as a long running query. The default value is 5.
8. **DDROWCOUNT** – Specify the number of long running queries for which details will be available in the detailed diagnosis page. By default, this parameter is set to 5. This indicates that even if the total number of long running queries is, say 10, the detailed diagnosis of this test will provide information pertaining to only 5 queries by default. For information related to more number of queries, you should increase the **DDROWCOUNT**.
9. **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 enabled/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.

Outputs of the test

One set of results for the Sybase server being monitored

Measurements made by the test

Measurement	Measurement Unit	Interpretation
Number of queries: Indicates the number of queries currently executing on the database server that have been running for more time than the configured **ELAPSED TIME**.	Number	The detailed diagnosis for this measure indicates the exact queries and which user is executing the queries. This information can be very useful in identifying queries that may be candidates for optimization.

	Avg elapsed time: The average time taken by the long running queries to execute.	MSecs

5.2.6.2 Sybase Statements Test

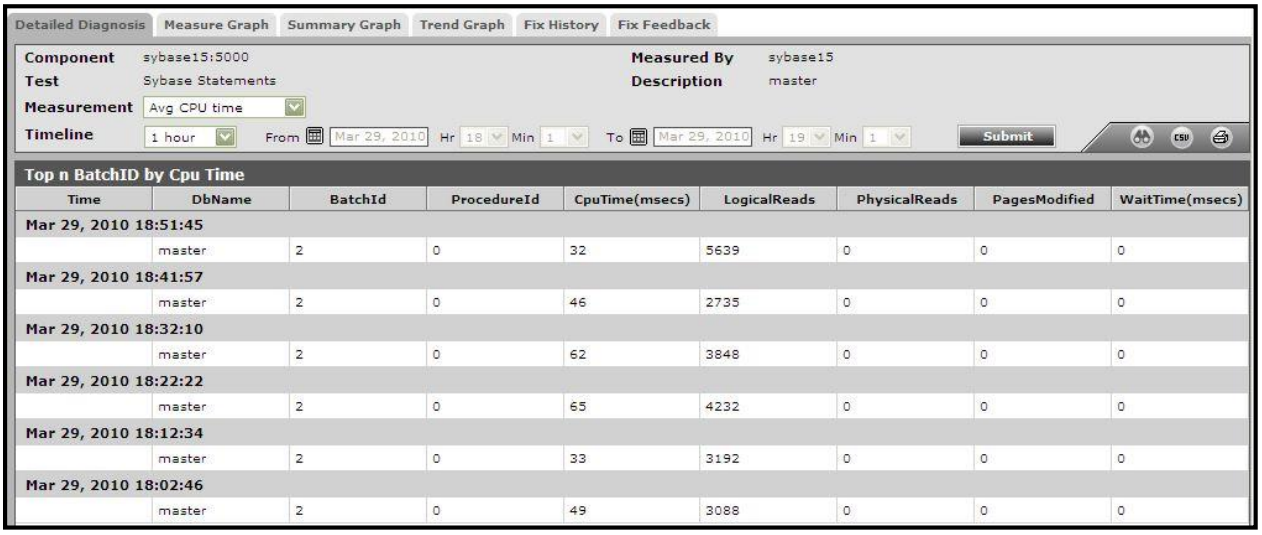
This test reports critical resource usage and general performance statistics related to transactions that are executed on the Sybase server. For execution, this test requires the **enable monitoring**, **statement statistics active**, **per object statistics active**, **statement pipe max messages**, and **statement pipe active** configuration parameters to be enabled.

Purpose	Reports critical resource usage and general performance statistics related to transactions that are executed on the Sybase server
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role".
5. PASSWORD – The password corresponding to the above user
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. DDROWCOUNT – Specify the number of transactions for which details will be available in the detailed diagnosis page. Such limit has been set to By default, this parameter is set to 5. This indicates that by default, the detailed diagnosis of this test will only report information related to the top-5 transactions to the database. To view details pertaining to more number of top transactions, increase the DDROWCOUNT.
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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every database being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Avg CPU time:
The average cpu time consumed by the transactions to this database.</td><td>MillSec</td><td>Ideally, this value should be low. If the value is high, then use the detailed diagnosis of this measure to view the top transactions in terms of CPU consumption. Such transactions can then be fine-tuned, so that they are more resource-efficient.</td></tr>
<tr><td>Max CPU time:
The maximum CPU time of transactions to this database.</td><td>MillSec</td><td></td></tr>
</table>

	Avg logical reads: The average number of logical reads performed by the transactions to this database.	Number	Use the detailed diagnosis of this measure to view the complete details of transactions causing logical reads to occur on the database.
	Max logical reads: The maximum number of logical reads performed by transactions to this databases	Number	
	Avg physical reads: The average number of physical reads performed by transactions to this database.	Number	Physical database reads can cause processing overheads to escalate, thereby affecting database performance. These transactions therefore have to be minimized. If the value of this measure is very high, then use the detailed diagnosis of this measure to view the details of the transactions that are causing physical reads to occur.
	Max physical reads: The maximum number of physical reads performed by transactions to this database.	Number	
	Avg page modified: The average number of pages modified by transactions to this database.	Number	Use the detailed diagnosis of this measure to view the details of transactions that are modifying pages.
	Max Page modified: The maximum number pages modified by transactions to this database.	Number	
	Avg long waited: The average execution time of transactions that have taken too long to execute.	Millsec	If the value of this measure is high, it could indicate that one/more inefficient queries are executing on this database. You might then have to use the detailed diagnosis of the measure to view the top transactions in terms of execution time, and identify that transaction that has taken the longest to execute. Fine-tune the transaction to prevent database slowdowns.”
	Max long waited: The maximum number of times transactions waited in a database.	MillSec	

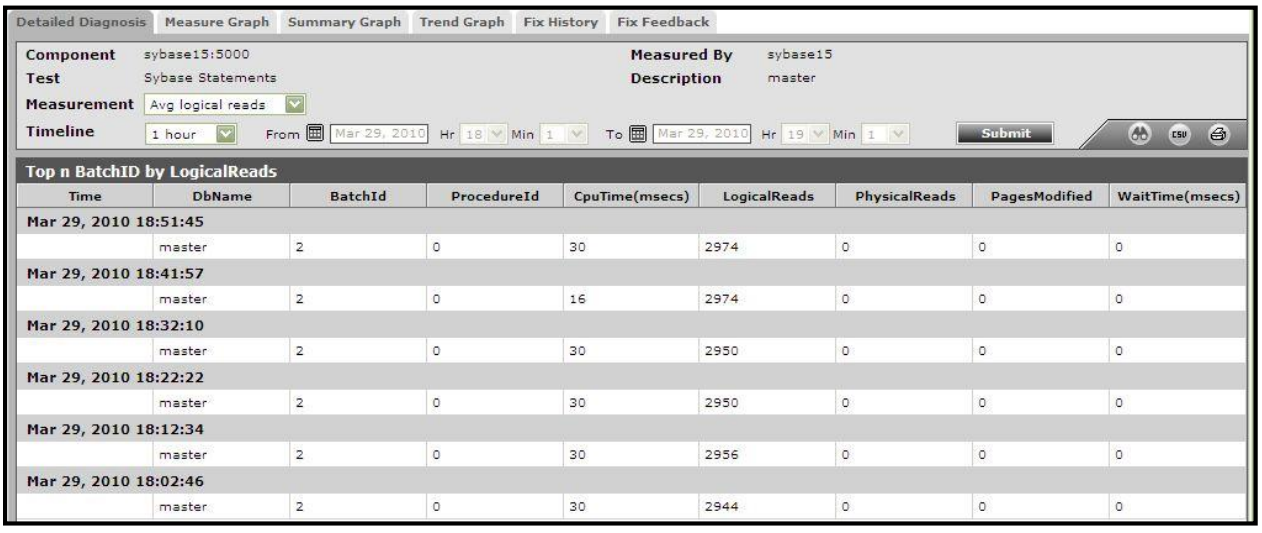
If the transactions to a database appear to be consuming too much CPU, then, you may want to identify the exact transaction that is utilizing the CPU resources excessively. For this purpose, you can use the detailed diagnosis of the *Avg CPU time* measure. This reveals the top 10 transactions in terms of their CPU usage, and thus enables administrators to isolate the most CPU-intensive transaction.



Time	DbName	BatchId	ProcedureId	CpuTime(msecs)	LogicalReads	PhysicalReads	PagesModified	WaitTime(msecs)
Mar 29, 2010 18:51:45								
	master	2	0	32	5639	0	0	0
Mar 29, 2010 18:41:57								
	master	2	0	46	2735	0	0	0
Mar 29, 2010 18:32:10								
	master	2	0	62	3848	0	0	0
Mar 29, 2010 18:22:22								
	master	2	0	65	4232	0	0	0
Mar 29, 2010 18:12:34								
	master	2	0	33	3192	0	0	0
Mar 29, 2010 18:02:46								
	master	2	0	49	3088	0	0	0

Figure 5.24: The detailed diagnosis of the Avg CPU time measure

In the event of a sudden spike in the number of logical reads to a database, check the detailed diagnosis of the *Avg logical reads* measure to identify the transaction that is causing reads to rise.



Time	DbName	BatchId	ProcedureId	CpuTime(msecs)	LogicalReads	PhysicalReads	PagesModified	WaitTime(msecs)
Mar 29, 2010 18:51:45								
	master	2	0	30	2974	0	0	0
Mar 29, 2010 18:41:57								
	master	2	0	16	2974	0	0	0
Mar 29, 2010 18:32:10								
	master	2	0	30	2950	0	0	0
Mar 29, 2010 18:22:22								
	master	2	0	30	2950	0	0	0
Mar 29, 2010 18:12:34								
	master	2	0	30	2956	0	0	0
Mar 29, 2010 18:02:46								
	master	2	0	30	2944	0	0	0

Figure 5. 25: The detailed diagnosis of the Avg logical reads measure

5.2.6.3 Sybase Users Test

This test monitors user sessions to the database server, and reports the number and status of processes executed by each user on the server.

<table>
<tr><td>Purpose</td><td colspan="3">Monitors user sessions to the database server, and reports the number and status of processes executed by each user on the server</td></tr>
<tr><td>Target of the test</td><td colspan="3">A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role".
5. PASSWORD – The password corresponding to the above user
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. EXCLUDEUSER - Specify a comma-separated list of user names to be exclude from the monitoring scope of this test in the EXCLUDEUSER text box.
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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every user being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total processes:
The number of processes initiated by this user on the database server. each username.</td><td>Number</td><td></td></tr>
<tr><td>Running processes:
The number of processes currently running for this user in the database.</td><td>Number</td><td>Use the detailed diagnosis of this measure to view the details of the running processes.</td></tr>
</table>

	Sleeping processes: The number of processes initiated by this user that are currently sleeping.	Number	If the value of this measure is very high, it could indicate a memory leak in the application. The administrator should take care to avoid sleeping processes. Use the detailed diagnosis of this measure to view the details of sleeping processes.
	Infected processes: The number of processes initiated by this user that are currently infected.	Number	If the value of this measure is very close to that of the *Total processes* measure, then further investigation would be required to control the infected processes. Use the detailed diagnosis of this measure to identify these processes and take the necessary corrective action.

Using the detailed diagnosis of the *Running processes* measure, you can find out which processes are currently running for a particular user.

Figure 5.26: The detailed diagnosis of the Running processes measure reported by the Sybase Users test

Using the detailed diagnosis of the *Sleeping processes* measure, you can find out which processes initiated by a particular user are currently sleeping.

Figure 5.27: The detailed diagnosis of the Sleeping processes measure reported by the Sybase Users test

5.2.6.4 Sybase Responses Test

This test, executed by an internal agent, tracks the statistics pertaining to the availability and response time of the Sybase adaptive server.

Purpose	To measure the statistics pertaining to the availability and response time of the database server
Target of the test	A Sybase adaptive server (ver. 12.5 and above) on which MDA tables have been installed
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Sybase server
3. PORT – The port on which the server is listening
4. USER – To enable this test to connect to the Sybase server and collect the required metrics, it is enough if you configure the test with the name of a Sybase user who has the "mon_role". However, for best results, it is recommended that you configure all Sybase tests with the credentials of a Sybase user who has the "mon_role", "sa_role', and "sybase_ts_role".
5. PASSWORD – The password corresponding to the above user
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. QUERY – By default, this parameter is set to select * from sysobjects. The test executes this executes the default query to report the availability and responsiveness of the server. If the user configured for this test does not have the right to execute the default query, then the QUERY parameter can be overridden with a query that that user has permission to execute.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every Sybase server being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Sybase server availability:
Indicates if the database instance is responding to requests or not.</td><td>Percent</td><td>A value of 100 for this measure indicates that the database is responding to a request. On the other hand, a value of 0 denotes that the database is not responding.
Availability problems may be caused by a misconfiguration/malfunctioning of the database instance, or because the instance is using an invalid user account.
Besides the above, this measure will report that the server is unavailable even if a connection to the database instance is unavailable, or if a query to the database fails. In this case, you can check the values of the DB connection availability and Query processor availability measures to know what is exactly causing the database instance to not respond to requests - is it owing to a connection unavailability? or is it due to a query failure?</td></tr>
<tr><td>Total response time:
Indicates the time taken by the database server to respond to a user query. This is the sum of connection time and query execution time.</td><td>Secs</td><td>A sudden increase in response time is indicative of a potential performance bottleneck on the database server.</td></tr>
</table>

	DB connection availability: Indicates whether the database connection is available or not.	Percent	If this measure reports the value 100 , it indicates that the database connection is available. The value 0 on the other hand indicates that the database connection is unavailable. A connection to the database may be unavailable if the database is down or if the database is listening on a port other than the one configured for it in the eG manager or owing to a poor network link. If the *Sybase server availability* measure reports the value 0, then, you can check the value of this measure to determine whether/not it is due to the unavailability of a connection to the server.
	Query processor availability: Indicates whether the database query is executed successfully or not.	Percent	If this measure reports the value 100, it indicates that the query executed successfully. The value 0 on the other hand indicates that the query failed. In the event that the *Sybase server availability* measure reports the value *0*, check the value of this measure to figure out whether the failed query is the reason why that measure reported a server unavailability.
	Query execution time: Indicates the time taken for query execution.	Secs	A high value could indicate that one/more queries to the database are taking too long to execute. Inefficient/badly designed queries to the database often run for long periods. If the value of this measure is higher than that of the *Connection time* measure, you can be rest assured that long running queries are the ones causing the responsiveness of the server to suffer.
	Records fetched: Indicates the number of records fetched from the database.	Number	The value 0 indicates that no records are fetched from the database

6

Monitoring MySQL Servers

MySQL is a multithreaded, multi-user SQL database management system, and is one of the most popular databases in the market today. Owing to a rise in its popularity in the past decade, the need to ensure the continuous availability and optimal performance of the MySQL database server has also attained significance.

eG Enterprise provides an exclusive *MySQL* monitoring model that runs quick health checks on the MySQL database server at configured intervals, and proactively alerts administrators to potential bottlenecks to the performance of the server.

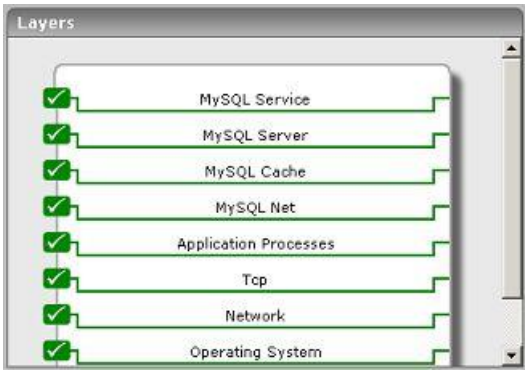


Figure 6.1: The layer model of the MySQL server

Using the model depicted by Figure 6.1, administrators can determine the following:

- Is the database server available? If so, how quickly does it respond to user requests?
- Is the server overloaded?
- Are clients able to connect to the server, or are there too many connection failures?
- Are connections been closed properly? Are there an unusual number of open connections to the server?
- Is the query cache been utilized optimally?
- Has adequate memory been allotted to the cache?
- Is the key buffer cache utilized well?
- Is query execution efficient, or do queries need to be optimized for better performance?
- Are rollbacks kept at a minimum?
- Should the sort_buffer be increased?

- How is the overall locking activity on the server? Are too many requests waiting to acquire locks?

This section will deal with the first four layers of the layer model only, as the rest of the layers have been discussed elaborately in the *Monitoring Unix and Windows Servers* document.

6.1 Pre-requisites for Monitoring the MySQL Server

An eG agent can monitor a MySQL database server and return performance metrics only when a MySQL connector is available in the **<EG_INSTALL_DIR>/lib** directory (in Unix, this would be **/opt/egurkha/lib** directory). To achieve this, refer to the procedure discussed in the *Configuring and Monitoring Database Servers* document.

The eG agent should be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges. For the related procedure, refer to Section 6.1.1 of this document.

The MySQL server should be added/managed in the eG administrative interface using its IP address only, and not its host name.

Note:

The MySQL server can be monitored in an agent-based or an agentless manner.

6.1.1 Configuring the eG Agent with Access Privileges

The MySQL tests should be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges on the target MySQL server.

If such a user does not pre-exist, then, in the *user* table of the *mysql* database of the target MySQL server, you need to manually create a user account with the aforesaid privileges.

To achieve this, follow the procedure discussed below:

1. To create a new user account, you must connect to the MySQL server as the MySQL **root** user. For that, first login to the MySQL host, and at the command prompt, issue the following command:

   mysql -u root

   If you have assigned a password to the **root** account, you will also need to supply a—**password** or **-p** option, as shown below:

   mysql -u root -pegurkha

2. After successfully logging into the MySQL server, issue the following statement to access the *mysql* database, which holds the *user* table:

   use mysql

3. Then, at the MySQL prompt, issue the following command to create a user:

   CREATE USER '<username>'@'<IP_address_of_eG_agent>' IDENTIFIED BY '<password>';

   GRANT PROCESS,SELECT ON . TO '<username>'@'<IP_address_of_eG_agent>';

   For instance, to ensure that user *john* (with password *john*) is able to connect to the MySQL server (being monitored) from the eG agent host, *192.168.8.91*, the following command is to be issued:

   CREATE USER 'john'@'192.168.8.91' IDENTIFIED BY 'john';

   GRANT PROCESS,SELECT ON . TO 'john'@'192.168.8.91';

Note:

a. The *CREATE* and *GRANT* commands are case-sensitive; therefore, take care while specifying the user name, password, and privileges.

b. Only the IP address of the eG agent's host can be provided as part of the *CREATE* command's syntax; the host name of the eG agent cannot be provided instead.

4. To ensure that the external agent is able to execute the **MySQL Network** test, make sure that you create a user with the same credentials (i.e., name and password) and privileges as above and map that user to the IP address of the external agent. For instance, in the example above, to enable the external agent at IP address *192.168.8.92* to run the **MySQL Network** test, your command should be:

   CREATE USER 'john'@'192.168.8.92' IDENTIFIED BY 'john';

   GRANT PROCESS,SELECT ON . TO 'john'@'192.168.8.92';

5. Once the above-mentioned commands execute successfully, the *user* table will be updated with two records for the user account that was newly created - one mapped to the internal/remote agent's IP address and another mapped to the external agent's IP address.

Then, proceed to configure the tests. While doing so, remember to configure the **USER** name and **PASSWORD** parameters with the name and password (respectively) that corresponds to the eG agent's IP address in the *user* table.

6.2 The MySQL Net Layer

This layer measures the network connectivity of the MySQL server by indicating the availability of the server over a network and its responsiveness to requests. In addition, the layer also tracks the data traffic to and from the server, and the overall health of the client connections to the server.

Figure 6.2: The tests associated with the MySQL Net layer

6.2.1 MySQL Test

This test monitors the availability and responsiveness of the MySQL database server by emulating a client connecting and executing queries on the MySQL server.

<table>
<tr><td>Purpose</td><td colspan="3">Monitors the availability and responsiveness of the MySQL database server by emulating a client connecting and executing queries on the MySQL server</td></tr>
<tr><td>Target of the test</td><td colspan="3">A MySQL server</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An internal/remote agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the MySQL server
3. PORT – The port on which the server is listening
4. DB – the name of a database on the server
5. USERNAME and PASSWORD - The eG agent has to be configured with the credentials of a user who has server-wide PROCESS and SELECT privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the database being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>MySQL server availability:
Indicates the availability of the server to execute a user query.</td><td>Percent</td><td>The availability is 100% when the server does respond to a request and 0% when it is not. Availability problems may be caused by a misconfiguration / malfunctioning of the database server, or if the server has not been started.
Besides the above, this measure will report that the server is unavailable even if a connection to the database instance is unavailable, or if a query to the database fails. In this case, you can check the values of the DB connection availability and Query processor availability measures to know what is exactly causing the database instance to not respond to requests - is it owing to a connection unavailability? or is it due to a query failure?</td></tr>
<tr><td>Connection time to database server:
Indicates the time taken to connect to the database server.</td><td>Secs</td><td>A dramatic increase in this value may be because the server has reached its connection handling capacity.</td></tr>
</table>

	Query execution time: Indicates the time taken to execute a database query.	Secs	A dramatic increase in this value could indicate a processing bottleneck with the database server.
	Total response time: Indicates the time taken by the database server to respond to a user query. This is the sum of the connection and query execution times.	Secs	A sudden increase in response time is indicative of a bottleneck with the database server.
	DB connection availability: Indicates whether the database connection is available or not.	Percent	If this measure reports the value 100 , it indicates that the database connection is available. The value 0 on the other hand indicates that the database connection is unavailable. A connection to the database may be unavailable if the database is down or if the database is listening on a port other than the one configured for it in the eG manager or owing to a poor network link. If the *MySQL server availability* measure reports the value 0, then, you can check the value of this measure to determine whether/not it is due to the unavailability of a connection to the server.
	Query processor availability: Indicates whether the database query is executed successfully or not.	Percent	If this measure reports the value 100, it indicates that the query executed successfully. The value 0 on the other hand indicates that the query failed. In the event that the *MySQL server availability* measure reports the value *0*, check the value of this measure to figure out whether the failed query is the reason why that measure reported a server unavailability.
	Records fetched: Indicates the number of records fetched from the database.	Number	The value 0 indicates that no records are fetched from the database

6.2.2 MySQL Network Test

This test monitors the data transmission between the MySQL server and its clients.

Purpose	Monitors the data transmission between the MySQL server and its clients
Target of the test	A MySQL server
Agent deploying the test	An external agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the MySQL server 3. **PORT** – The port on which the server is listening 4. **DB** – the name of a database on the server 5. **USERNAME** and **PASSWORD** - The eG agent has to be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for the database being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Data transmit rate: The rate of data being transmitted by the server in response to client requests during the last measurement period	KB/Sec	The data transmission rate reflects the workload on the server.
	Data received rate: The rate of data received by the server from clients during the last measurement period	KB/Sec	This measure also characterizes the workload on the server.

6.2.3 MySQL Connection Test

This test gives information about client connections to a MySQL server.

Purpose	Monitors the client connections to a MySQL server
Target of the test	A MySQL server
Agent deploying the test	An internal/remote agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the MySQL server
3. PORT – The port on which the server is listening
4. DB – the name of a database on the server
5. USERNAME and PASSWORD - The eG agent has to be configured with the credentials of a user who has server-wide PROCESS and SELECT privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the database being monitored</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Connection create rate:
The rate at which applications attempted to connect to the MySQL server during the last measurement period</td><td>Conns/Sec</td><td>The data transmission rate reflects the workload on the server.</td></tr>
<tr><td>Active connections:
The number of currently open connections to the MySQL server</td><td>Number</td><td>A high value may be indicative of connections not being closed properly by applications, or a bottleneck in the server</td></tr>
<tr><td>Connection aborts:
The number of connections that were aborted because the client terminated without closing the connection properly during the last measurement period</td><td>Number</td><td></td></tr>
<tr><td></td><td>Failed connections:
The number of connection attempts to the MySQL server that failed during the last measurement period</td><td>Number</td><td>A high value can indicate:
➢ a configuration problem with the server/clients;
➢ some malicious attack on the server;</td></tr>
</table>

	Max connections: The maximum number of connections that were in use simultaneously in use during the last measurement period	Number	

6.3 The MySQL Cache Layer

The tests associated with this layer monitor the health of the query cache and the buffer cache on the MySQL server.

Figure 6.3: The tests associated with the MySQL Cache layer

6.3.1 MySQL Queue Cache Test

This test monitors the health of the query cache in the MySQL server.

Purpose	Monitors the health of the query cache in the MySQL server
Target of the test	A MySQL server
Agent deploying the test	An internal/remote agent
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the MySQL server 3. **PORT** – The port on which the server is listening 4. **DB** – the name of a database on the server 5. **USERNAME** and **PASSWORD** - The eG agent has to be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
Outputs of the test	One set of results for the database being monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Queries registered in cache: Indicates the number of queries that are registered with the cache.	Number	
	Queries added to cache: Indicates the number of queries added to the cache during the last measurement period.	Number	
	Cache hits: Indicates the number of times the cache was accessed during the last measurement period.	Number	
	Queries deleted from cache: Indicates the number of queries that were deleted from the cache during the last measurement period.	Number	A high value could indicate low memory allocation for the query cache. Since reading from the cache is less expensive than reading directly from the database, a higher memory allocation to the cache is advisable.
	Non-cached queries: Indicates the number of queries that were not cached (not cacheable due to their QUERY_CACHE_TYPE) during the last measurement period.	Number	
	Free memory in cache: Indicates the amount of free memory in the query cache.	MB	A low value of this measure could cause subsequent queries to be rejected by the cache, owing to low memory availability. It would be good practice to tune the cache memory to handle more load.
	Free blocks in cache: Indicates the number of free memory blocks in the query cache currently.	Number	

	Blocks in cache: Indicates the total number of blocks in the query cache currently.	Number	
	Queries in cache: This is the sum of cache inserts, cache_hits, and cache_not_cached queries during the last measurement period.	Number	

6.3.2 MySQL Cache Test

This test reports statistics pertaining to the key buffer cache in the MySql server.

Purpose	Reports statistics pertaining to the key buffer cache in the MySql server		
Target of the test	A MySQL server		
Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the MySQL server 3. **PORT** – The port on which the server is listening 4. **DB** – the name of a database on the server 5. **USERNAME** and **PASSWORD** - The eG agent has to be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for the database being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Key reads: Indicates the number of physical reads of a key block from the disk during the last measurement period.	Number	A high value of this measure indicates that your key_buffer_size variable is probably too small.

	Key read requests: Indicates the number of key read requests received by the cache during the last measurement period.	Number	
	Not flushed key blocks: Indicates the key blocks in the key cache that have been modified, but have not been flushed to the disk during the last measurement period.	Number	

6.4 The MySQL Server Layer

The tests associated with this layer indicate the level of activity on the MySQL server by monitoring the threads, queries, and SQL statements executed on it.

Figure 6.4: The tests associated with the MySQL Server layer

6.4.1 MySQL Threads Test

This test tracks the MySQL server threads and reports various performance statistics pertaining to them.

Purpose	Reports statistics pertaining to the server threads
Target of the test	A MySQL server
Agent deploying the test	An internal/remote agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the MySQL server
3. PORT – The port on which the server is listening
4. DB – the name of a database on the server
5. USERNAME and PASSWORD - The eG agent has to be configured with the credentials of a user who has server-wide PROCESS and SELECT privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the database being monitored</td></tr>
<tr><td rowspan="5">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Threads cached:
Indicates the number of threads in the thread cache currently.</td><td>Number</td><td></td></tr>
<tr><td>Threads spawned:
Indicates the number of threads newly created to handle new connections during the last measurement period.</td><td>Number</td><td>Where possible ensure that the number of total threads is not set to be much larger than the maximum number of simultaneous requests expected for the server.</td></tr>
<tr><td>Threads active:
Indicates the number of threads that are currently active.</td><td>Number</td><td>A high value for this measure is indicative of a high load on database server.</td></tr>
<tr><td>Slow launch threads:
Indicates the number of threads that have taken more than slow_launch_time for creation, during the last measurement period.</td><td>Number</td><td>A high value for this measure could cause excessive delay in processing new connection requests to the database server.</td></tr>
<tr><td></td><td>Delated insert threads:
Indicates the number of delayed insert handler threads in use.</td><td>Number</td><td></td></tr>
</table>

6.4.2 MySQL Resources Test

This test measures the effect of query execution on the disk and memory resources of the MySQL server.

Purpose	Measures the effect of query execution on the disk and memory resources of the MySQL server		
Target of the test	A MySQL server		
Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the MySQL server 3. **PORT** – The port on which the server is listening 4. **DB** – the name of a database on the server 5. **USERNAME** and **PASSWORD** - The eG agent has to be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for the database being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Opened tables: Indicates the number of tables that were opened during the last measurement period.	Number	If this value is high, then it indicates that the "table_cache" value is probably too small.
	Created tmp disk tables: Indicates the number of implicit temporary tables that were created on the disk, while executing statements during the last measurement period.	Number	If the value for this measure is high, consider increasing the 'tmp_table_size' configuration setting for the server.
	Created tmp tables: Indicates the number of implicit temporary tables that were created in the memory created while executing statements during the last measurement period.	Number	

6.4.3 MySql Query Test

This test reports the performance statistics pertaining to the queries executed on the MySQL server.

Purpose	Reports the performance statistics pertaining to the queries executed on the MySQL server		
Target of the test	A MySQL server		
Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the MySQL server 3. **PORT** – The port on which the server is listening 4. **DB** – the name of a database on the server 5. **USERNAME** and **PASSWORD** - The eG agent has to be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for the database being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Query rate: Indicates the rate at which queries are sent to the server.	Queries/Sec	This is an indicator of server workload.
	Slow queries: Indicates the number of queries that have taken more than the 'long_query_time' for execution, during the last measurement period.	Number	This value should ideally be 0. If it remains consistently high, the administrator should look to identify and optimize the slow queries.
	Handler read first: Indicates the number of times the first entry was read from an index during the last measurement period.	Number	If the value of this measure is high, it suggests that the server is doing a lot of full index scans.

	Handler read key: Indicates the number of requests that were received in the last measurement period, to read a row based on a key.	Number	If the value of this measure is high, it indicates that your queries and tables are properly indexed.
	Handler read next: Indicates the number of requests received in the last measurement period, to read the next row in the key order.	Number	This will be incremented if you are querying an index column with a range constraint. This will also be incremented if you are doing an index scan.
	Handler read prev: Indicates the number of requests received in the last measurement period, to read the previous row in the key order.	Number	This is mainly used to optimize ORDER BY... DESC.
	Handler read rnd: Indicates the number of requests received in the last measurement period, to read a row based on a fixed position.	Number	This will be high if you are executing a lot of queries that require sorting of the result. If the value of this measure is high, then you probably have a lot of queries that require MySQL to scan whole tables or you have joins that do not use keys properly.
	Handler read rnd next: Indicates the number of requests received in the last measurement period, to read the next row in the datafile.	Number	This will be high if you are performing a lot of table scans. Generally, this suggests that your tables are not properly indexed or that your queries are not written to use the indexes properly.

6.4.4 MySql Activity Test

This test tracks the writes, inserts, deletes, and flushes happening on a MySQL server database.

Purpose	Tracks the writes, inserts, deletes, and flushes happening on a MySQL server database
Target of the test	A MySQL server
Agent deploying the test	An internal/remote agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the MySQL server
3. PORT – The port on which the server is listening
4. DB – the name of a database on the server
5. USERNAME and PASSWORD - The eG agent has to be configured with the credentials of a user who has server-wide PROCESS and SELECT privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the database being monitored</td></tr>
<tr><td rowspan="6">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Delayed writes:
Indicates the rate at which rows are written with INSERT DELAYED.</td><td>Writes/Sec</td><td></td></tr>
<tr><td>Delayed errors:
Indicates the rate at errors occurred in the rows written with INSERT DELAYED.</td><td>Errors/Sec</td><td></td></tr>
<tr><td>Updates:
Indicates the rate at which requests to update a row in a table were received.</td><td>Updates/Sec</td><td></td></tr>
<tr><td>Insertes:
Indicates the rate at which requests to insert a row in a table were received.</td><td>Inserts/Sec</td><td></td></tr>
<tr><td>Deletions:
Indicates the rate at which requests to delete a row in a table were received.</td><td>Deletions/Sec</td><td></td></tr>
<tr><td></td><td>Flushes:
Indicates the rate at which FLUSH commands were executed.</td><td>Flushes/Sec</td><td></td></tr>
</table>

6.5 The MySQL Service Layer

The tests associated with this layer monitor the locking activity, sorting activity, and transactions executing on the MySQL server.

Figure 6.5: The tests associated with the MySQL Service layer

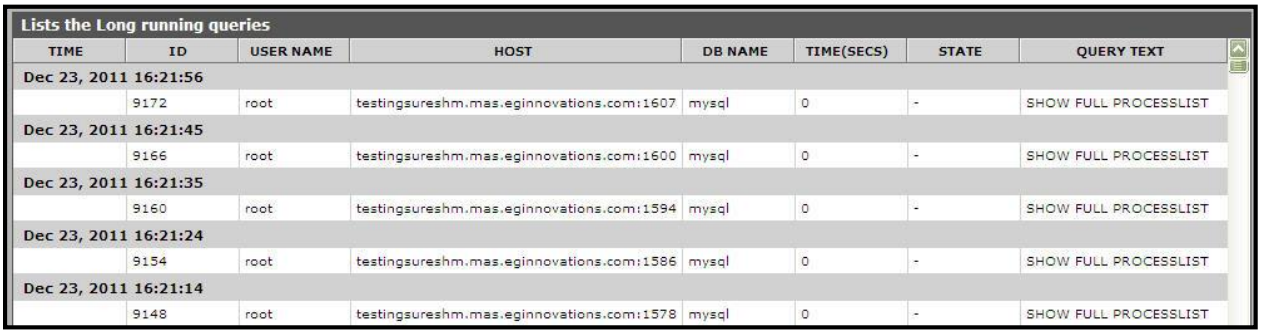
6.5.1 MySQL Long Running Queries Test

This test tracks the currently executing queries on a MySQL server and determines the number of queries that have been running for a long time. You can also use the detailed diagnosis of this test to drill down to the exact queries that have been running for an unreasonably long time, and thus isolate the resource-intensive queries to the database.

Purpose	Tracks the writes, inserts, deletes, and flushes happening on a MySQL server database
Target of the test	A MySQL server
Agent deploying the test	An internal/remote agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the MySQL server
3. PORT – The port on which the server is listening
4. DB – the name of a database on the server
5. USERNAME and PASSWORD - The eG agent has to be configured with the credentials of a user who has server-wide PROCESS and SELECT privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. ELAPSED TIME - In the ELAPSED TIME text box, specify the duration (in seconds) for which a query should have executed for it to be regarded as a long running query. The default value is 10.
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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the database being monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Long running queries:
Indicates the number of queries currently executing on the database server that have been running for more time than the configured ELAPSED TIME.</td><td>Number</td><td>The detailed diagnosis for this measure indicates the exact queries and which user is executing the queries. This information can be very useful in identifying queries that may be candidates for optimization.</td></tr>
</table>

The detailed diagnosis for the *Long running queries* measure indicates the exact queries and which user is executing the queries. This information can be very useful in identifying queries that may be candidates for optimization.



Lists the Long running queries

TIME	ID	USER NAME	HOST	DB NAME	TIME(SECS)	STATE	QUERY TEXT
Dec 23, 2011 16:21:56							
	9172	root	testingsureshm.mas.eginnovations.com:1607	mysql	0	-	SHOW FULL PROCESSLIST
Dec 23, 2011 16:21:45							
	9166	root	testingsureshm.mas.eginnovations.com:1600	mysql	0	-	SHOW FULL PROCESSLIST
Dec 23, 2011 16:21:35							
	9160	root	testingsureshm.mas.eginnovations.com:1594	mysql	0	-	SHOW FULL PROCESSLIST
Dec 23, 2011 16:21:24							
	9154	root	testingsureshm.mas.eginnovations.com:1586	mysql	0	-	SHOW FULL PROCESSLIST
Dec 23, 2011 16:21:14							
	9148	root	testingsureshm.mas.eginnovations.com:1578	mysql	0	-	SHOW FULL PROCESSLIST

Figure 6.6: The detailed diagnosis of the Long running queries measure

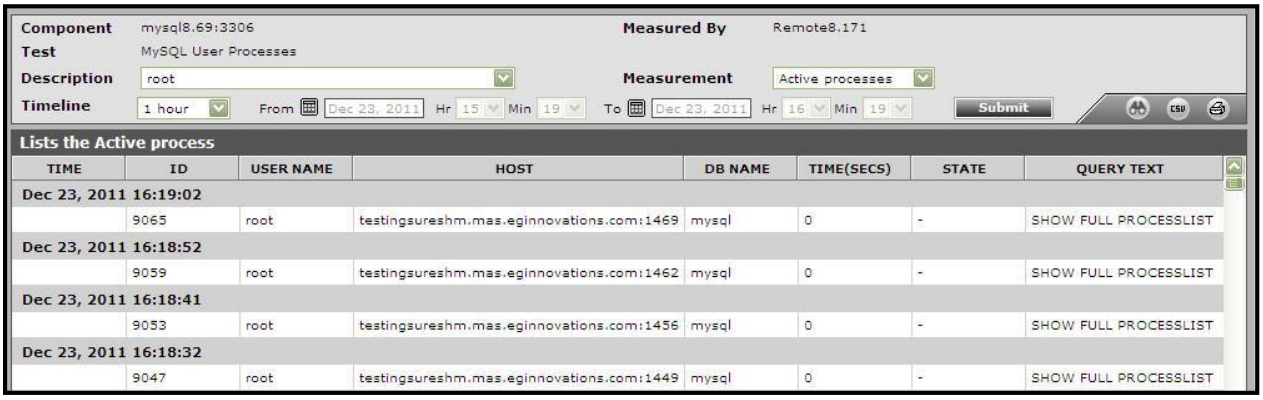
6.5.2 MySQL User Processes Test

This test reports the number and state of the processes of each user who is currently connected to the MySQL server. Using the metrics reported by this test, administrators can promptly isolate idle processes, which are a drain on a server's resources.

Purpose	Reports the number and state of sessions of each user who is currently connected to the MS SQL server
Target of the test	A MySQL server
Agent deploying the test	An internal/remote agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the MySQL server
3. PORT – The port on which the server is listening
4. DB – the name of a database on the server
5. USERNAME and PASSWORD - The eG agent has to be configured with the credentials of a user who has server-wide PROCESS and SELECT privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document.
6. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
7. IDLE TIME - Specify the time duration (in seconds) above which the processes that are waiting in the database will be regarded as idle.
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.
9. The option to selectively enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each user currently connected to the MySQL server monitored</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total processes:
Indicates the total number of processes for this user.</td><td>Number</td><td>The detailed diagnosis of this measure if enabled, lists out all the processes for this user.</td></tr>
<tr><td>Active processes:
Indicates the number of processes of this user that are currently active.</td><td>Number</td><td>The detailed diagnosis of this measure indicates the exact active processes of this user and the time for which the processes are actually active.</td></tr>
<tr><td>Inactive processes:
Indicates the processes that were inactive for this user in this database.</td><td>Number</td><td>The detailed diagnosis of this measure if enabled, indicates the inactive processes of this user and the time for which the processes were inactive.</td></tr>
<tr><td></td><td>Idle processes:
Indicates the processes that are idle for this user in this database.</td><td>Number</td><td>The detailed diagnosis of this measure if enabled, indicates the idle processes of this user and the time for which the processes were idle.</td></tr>
</table>

The detailed diagnosis of the *Active processes* measure, if enabled, will indicatethe exact active processes of this user and the time for which the processes were actually active.



Component: mysql8.69:3306 | Measured By: Remote8.171
Test: MySQL User Processes
Description: root | Measurement: Active processes
Timeline: 1 hour | From: Dec 23, 2011 Hr 15 Min 19 | To: Dec 23, 2011 Hr 16 Min 19 | Submit

Lists the Active process

TIME	ID	USER NAME	HOST	DB NAME	TIME(SECS)	STATE	QUERY TEXT
Dec 23, 2011 16:19:02							
	9065	root	testingsureshm.mas.eginnovations.com:1469	mysql	0	-	SHOW FULL PROCESSLIST
Dec 23, 2011 16:18:52							
	9059	root	testingsureshm.mas.eginnovations.com:1462	mysql	0	-	SHOW FULL PROCESSLIST
Dec 23, 2011 16:18:41							
	9053	root	testingsureshm.mas.eginnovations.com:1456	mysql	0	-	SHOW FULL PROCESSLIST
Dec 23, 2011 16:18:32							
	9047	root	testingsureshm.mas.eginnovations.com:1449	mysql	0	-	SHOW FULL PROCESSLIST

Figure 6.7: The detailed diagnosis of the Active processes measure

The detailed diagnosis of this measure if enabled, indicates the idle processes of this user and the time for which the processes were idle. Using this information, you can understand how each of the idle connections were made - i.e., using which program - and from where - i.e., from which host.

Lists the Idle process

TIME	ID	USER NAME	HOST	DB NAME	TIME(SECS)	STATE	QUERY TEXT
Dec 23, 2011 15:05:40							
	6415	root	testingsureshm.mas.eginnovations.com:2056	mysql	0	-	-
Dec 23, 2011 14:50:24							
	5865	root	testingsureshm.mas.eginnovations.com:1315	mysql	0	-	-
Dec 23, 2011 12:51:12							
	5334	root	localhost:2974	mysql	241	-	-

Figure 6.8: The detailed diagnosis of the Idle processes meaure

6.5.3 MySQL Database Size Test

This test reports the size of each MySQL database.

Purpose	Reports the number and state of sessions of each user who is currently connected to the MS SQL server
Target of the test	A MySQL server
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the MySQL server 3. **PORT** – The port on which the server is listening 4. **DB** – the name of a database on the server 5. **USERNAME** and **PASSWORD** - The eG agent has to be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for each database on the target MySQL server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Database size: Indicates the current size of this database.	MB	

6.5.4 MySQL Transactions Test

Rollbacks are costly operations on the database. This test monitors the percentage of rollbacks happening for user transactions in a database instance.

Purpose	Tracks the writes, inserts, deletes, and flushes happening on a MySQL server database		
Target of the test	A MySQL server		
Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the MySQL server 3. **PORT** – The port on which the server is listening 4. **DB** – the name of a database on the server 5. **USERNAME** and **PASSWORD** - The eG agent has to be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for the database being monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**User commits:** Indicates the number of internal COMMIT commands issued during the last measurement period.	Number	
	User rollbacks: Indicates the number of internal ROLLBACK commands issued during the last measurement period.	Number	Ideally, there should be few user rollbacks happening.
	Rollbacks: Indicates the number of internal rollbacks expressed as a percentage of the total transactions with the database.	Percent	Ideally, there should be few user rollbacks happening.

6.5.5 MySQL Sorts Test

The test monitors the sort operation performed on the MySQL database server.

Purpose	Monitors the sort operation performed on the MySQL database server		
Target of the test	A MySQL server		
Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the MySQL server 3. **PORT** – The port on which the server is listening 4. **DB** – the name of a database on the server 5. **USERNAME** and **PASSWORD** - The eG agent has to be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for the database being monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Range sort rate:** Indicates the rate at which ranges were sorted.	Sorts/Sec	
	Row sort rate: Indicates the rate at which rows were sorted	Sorts/Sec	
	Scan sort rate: Indicates the rate at which sorting was performed by scanning the table	Sorts/Sec	
	Merge passed sort rate: Indicates the rate at which the sort algorithm performed merge passes	Sorts/Sec	If this value is large you should consider increasing the sort_buffer.

6.5.6 MySQL Locks Test

The locking activity of a database server must be monitored carefully because an application holding a specific lock for a long time could cause a number of other transactions relying on the same lock to fail. The MySQLLocks test monitors the locking activity on a database server instance.

Purpose	Monitors the locking activity on a database server instance		
Target of the test	A MySQL server		
Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the MySQL server 3. **PORT** – The port on which the server is listening 4. **DB** – the name of a database on the server 5. **USERNAME** and **PASSWORD** - The eG agent has to be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.		
Outputs of the test	One set of results for the database being monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Lock waits:** Indicates the percentage of lock requests that could not be satisfied immediately and hence, required the caller to wait before being granted the lock.	Percent	A high value of waits can have an adverse impact on application performance. Possible reasons for this behavior could be: ➢ inadequate number of locks available in the database, ➢ unusually high locking behavior of applications accessing the database, ➢ improper database application design, etc.
	Table lock waits: Indicates the number of times in the last measurement period a table lock could not be acquired immediately and a wait was needed.	Number	If the number of waits is high, application performance could suffer. You should first optimize your queries, and then either split your table(s) or use replication.

6.5.7 MySQL Top Tables By Size Test

To make sure that a database does not grow uncontrollably, administrators may want to periodically check the size of the database tables, isolate the tables with the maximum size, and see if the table size can somehow be reduced. The **MySQL Table Size** test helps administrators greatly in this exercise. This test automatically identifies to top 5 tables in a given database in terms of size, and reports the current size of each table.

To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *MySQL* as the **Component type**, *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Automatically identifies to top 5 tables in a given database in terms of size, and reports the current size of each table
Target of the test	A MySQL server
Agent deploying the test	An internal/remote agent
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the MySQL server 3. **PORT** – The port on which the server is listening 4. **DB** – the name of a database on the server 5. **USERNAME** and **PASSWORD** - The eG agent has to be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

Outputs of the test	One set of results for each of the top-5 tables on the database being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Table size: Indicates the current size of this table.	MB	Compare the values of this measure across the top-5 tables to know which table is of the maximum size.

6.5.8 MySQL Top Tables By Records Test

In an effort to track the growth of individual databases, administrators may want to time and again check how many rows of data each table in a database contains. In the process, they will be able to identify those tables with more than a permissible number of records. Based on this finding, they can then decide whether deletion of a few or more unnecessary records from these 'large tables' can keep database growth under check. The **MySQL Table Record Count** test helps administrators in this exercise. This test automatically discovers those tables in a given database that contain more than a configured number of records, and reports the number of records fetched from each such table.

To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *MySQL* as the **Component type**, *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Automatically discovers those tables in a given database that contain more than a configured number of records, and reports the number of records fetched from each such table		
Target of the test	A MySQL server		
Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the MySQL server 3. **PORT** – The port on which the server is listening 4. **DB** – the name of a database on the server 5. **USERNAME** and **PASSWORD** - The eG agent has to be configured with the credentials of a user who has server-wide **PROCESS** and **SELECT** privileges on the monitored MySQL server. To know how to create such a user, refer to Section 6.1.1 of this document. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 7. **TABLE ROWS** - Specify the number of records beyond which this test will consider a table as a large table.		
Outputs of the test	One set of results for each of the tables with a record count that exceeds the **TABLE ROWS** configuration		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Total records:** Indicates the number of records fetched from this table.	Number	Compare the values of this measure across the ables to know which table has the maximum number of records.

7

Monitoring Informix Dynamic Servers

The Informix Dynamic Server is a database server that manages traditional relational, object-relational, and web-based databases. It supports alphanumeric and rich data, such as graphics, multimedia, geospatial, HTML, and user-defined types. It is typically used on UNIX, Linux, or Windows with online transaction processing (OLTP), data marts, data warehouses, and e-business applications. Any operational inefficiency or non-availability of the Informix server can therefore adversely impact the performance of the e-business application it supports, causing the business itself to suffer. To avoid such adversities, it is imperative that the Informix server is monitored, and performance problems instantly brought to the attention of the administrator.

eG Enterprise presents an exclusive *Informix* monitoring model (see Figure 7.1) that consists of a set of hierarchical layers, each of which is associated with a wide variety of tests.

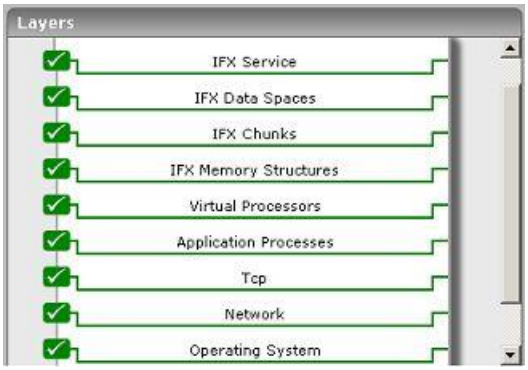


Figure 7.1: The layer model of an Informix database server

These tests, which are configured to execute on the Informix server periodically, extract a wide range of performance statistics from the server. These statistics reveal the following:

- Is the Informix database server available? If so, how quickly does it respond to user requests?
- Is any virtual processor class consuming CPU resources excessively?
- Are buffers being effectively used while reading/writing data, or are direct disk reads and writes high?
- How well does the database manage locks? Are too many requests waiting for locks? Are there a large number of deadlocks?
- Do the logical logs have adequate free pages?
- Are any chunks on the Informix server in an offline or inconsistent state? If so, which are tehy?
- Is any chunk experiencing a space crunch?
- Is any dbspace running out of free space?

- Are users able to access data quickly? Are sorts on disk and memory performed frequently to ensure quick and easy access?
- Are there too many open sessions on the server? Who initiated the sessions, and how long have they been open?
- Are transaction rollbacks kept at a minimum?
- Have too many transactions been running for a long time?

7.1 The Virtual Processors Layer

Database server processes are called virtual processors because the way they function is similar to the way that a CPU functions in a computer. Just as a CPU runs multiple operating-system processes to service multiple users, a database server virtual processor runs multiple threads to service multiple SQL client applications. Virtual processors are divided into classes depending upon the type of processing that they do. Each class of virtual processor is dedicated to processing certain types of threads.

This layer monitors every virtual processor class to report the number of processors associated with it and their collective CPU usage, so that CPU-intensive classes can be quickly determined.

Figure 7.2: The tests associated with the Virtual Processors layer

7.1.1 Informix VP Test

The IfxVPTest reports the CPU utilization and number of processors available for each virtual processor class.

Purpose	Reports the CPU utilization and number of processors available for each virtual processor class
Target of the test	An Informix Dynamic server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Informix Dynamic server
3. PORT – The port on which the server is listening
4. DB – the name of a database on the server
5. INSTANCE - the Informix server instance being monitored
6. USER – a valid user name to login to the specified database
7. PASSWORD – The password corresponding to the above user
8. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
9. DBLOCALE - Specify the language/locale in which your database is installed. This will allow you to return the query in the language supported by your database. By default this attribute is set to “none”, which means that by default, queries are returned in the English locale.
10. CLIENTLOCALE - Specify the language/locale in which the eG agent is running. Normally, when you use an internal agent to monitor Informix, the CLIENTLOCALE and DBLOCALE will be same. However, the CLIENTLOCALE may differ when you are monitoring the Informix tests using a remote agent (agentless monitoring). By default, the CLIENTLOCALE is set to “none”, indicating that the eG agent runs in the English locale, by default.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every virtual processor class on the Informix server being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Number of processors:
The total number of virtual processors available for this class.</td><td>Number</td><td></td></tr>
<tr><td>User CPU usage:
The percentage of CPU time taken by the user processes during the last measurement period.</td><td>Number</td><td>A high value may be indicative of excessive load on this class of virtual processors. If this value remains very high for a long period of time, then configure additional virtual processors for this class.</td></tr>
<tr><td></td><td>System CPU usage:
The percentage of CPU time taken by the Informix system during the last measurement period</td><td>Number</td><td>A high value may be indicative of excessive load on this class of virtual processors. If this value remains very high for a long period of time, then configure additional virtual processors for this class.</td></tr>
</table>

7.2 The IFX Memory Structures Layer

The tests associated with this layer track how well the Informix server performs the following activities:

- Caching and buffer flushing
- Lock activity
- Maintaining logical and physical logs

Figure 7.3: The tests associated with the IFX Memory Structures layer

7.2.1 Informix Buffers Test

The IfxBuffer test reports statistics pertaining to the caching and buffer flushing activities of the Informix database server.

Purpose	Reports statistics pertaining to the caching and buffer flushing activities of the Informix database server
Target of the test	An Informix Dynamic server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Informix Dynamic server
3. PORT – The port on which the server is listening
4. DB – the name of a database on the server
5. INSTANCE - the Informix server instance being monitored
6. USER – a valid user name to login to the specified database
7. PASSWORD – The password corresponding to the above user
8. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
9. DBLOCALE - Specify the language/locale in which your database is installed. This will allow you to return the query in the language supported by your database. By default this attribute is set to “none”, which means that by default, queries are returned in the English locale.
10. CLIENTLOCALE - Specify the language/locale in which the eG agent is running. Normally, when you use an internal agent to monitor Informix, the CLIENTLOCALE and DBLOCALE will be same. However, the CLIENTLOCALE may differ when you are monitoring the Informix tests using a remote agent (agentless monitoring). By default, the CLIENTLOCALE is set to “none”, indicating that the eG agent runs in the English locale, by default.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Informix server being monitored</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Read cache hit ratio:
The ratio of number of buffer reads to disk reads.</td><td>Percent</td><td>This measure is an indicator of the overall performance of your database server. This value should always be high.</td></tr>
<tr><td>Write cache hit ratio:
The ratio of number of buffer writes to disk writes.</td><td>Percent</td><td>This measure is an indicator of the overall performance of your database server. This value should always be high.</td></tr>
<tr><td>Flushes:
The rate at which buffer-pool buffers were flushed to disk.</td><td>Flushes/Sec</td><td></td></tr>
<tr><td></td><td>Foreground writes:
The number of foreground writes per second during the last measurement period</td><td>Writes/Sec</td><td>Foreground writes should be avoided. They slow down the performance of your database server. If you find that foreground writes are occurring on a regular basis, tune the value of the page-cleaning parameters. Either increase the number of page cleaners or decrease the value of LRU_MAX_DIRTY.</td></tr>
</table>

	LRU writes: The number of LRU writes per second during the last measurement period	Writes/Sec	Compare this value with Foreground_writes and Chunk_writes to get an understanding of how the buffer flushing occurs in Informix.
	Chunk writes: The number of chunk writes per second during the last measurement period	Writes/Sec	Chunk writes are commonly performed by page-cleaner threads during a checkpoint or, possibly, when every page in the shared-memory buffer pool is modified. Chunk writes, which are performed as sorted writes, are the most efficient writes available to the database server. Compare this value with Foreground_writes and LRU writes to get an understanding of how the buffer flushing occurs in Informix.

7.2.2 Informix Locks Test

This test reports the lock related measures of an Informix database server.

Purpose	Reports the lock related measures of an Informix database server
Target of the test	An Informix Dynamic server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Informix Dynamic server
3. PORT – The port on which the server is listening
4. DB – The name of a database on the server
5. INSTANCE - The Informix server instance being monitored
6. USER – A valid user name to login to the specified database
7. PASSWORD – The password corresponding to the above user
8. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
9. DBLOCALE - Specify the language/locale in which your database is installed. This will allow you to return the query in the language supported by your database. By default this attribute is set to “none”, which means that by default, queries are returned in the English locale.
10. CLIENTLOCALE - Specify the language/locale in which the eG agent is running. Normally, when you use an internal agent to monitor Informix, the CLIENTLOCALE and DBLOCALE will be same. However, the CLIENTLOCALE may differ when you are monitoring the Informix tests using a remote agent (agentless monitoring). By default, the CLIENTLOCALE is set to “none”, indicating that the eG agent runs in the English locale, by default.
11. DETAILED DIAGNOSIS - To make diagnosis more efficient and accurate, the eG Enterprise system 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 against DETAILED DIAGNOSIS. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Informix server being monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Lock requests:
The number of lock requests received.</td><td>Number</td><td>A high value indicates that there is high locking activity in the system and may need close scrutiny for the type of locks being requested. The detailed diagnosis of this measure, if enabled, provides a list of top 10 user sessions holding maximum number of locks.</td></tr>
</table>

	Lock waits: The number of lock waits.	Number	A high value of waits can have an adverse impact on application performance. Possible reasons for this behavior could be: ▪ Inadequate number of locks available in the database ▪ Unusually high locking behavior of applications accessing the database ▪ Improper database application design, etc.
	Lock timeouts: The number of locks that timed out.	Number	Lock timeouts can be useful for removing tasks that acquire some locks, and then wait for long periods of time blocking other users.
	Deadlocks: The number of deadlocks.	Number	A deadlock may arise due to various situations including bad design of queries and deficient coding practices. A deadlock is a situation where both/all the lock requestors are in a mutual or a multi-way tie. Any deadlocks are detrimental to database application performance. The detailed diagnosis of this measure, if enabled, will report the details of user sessions performing deadlocks.

7.2.3 Informix Logical Logs Test

The logical logs are one of the most important resources of the database server. Your database server will be blocked if they become full because they could not be backed up. You might also lose transactions in the case of a failure, if your logical logs have not been backed up. Thus, observing the behaviour of the logical logs is essential. This test collects and reports measures related to the logical logs.

Purpose	Collects and reports measures related to the logical logs of an Informix database server
Target of the test	An Informix Dynamic server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Informix Dynamic server
3. PORT – The port on which the server is listening
4. INSTANCE - The Informix server instance being monitored
5. USER – A valid Informix user name
6. PASSWORD – The password corresponding to the above user
7. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
8. DBLOCALE - Specify the language/locale in which your database is installed. This will allow you to return the query in the language supported by your database. By default this attribute is set to "none", which means that by default, queries are returned in the English locale.
9. CLIENTLOCALE - Specify the language/locale in which the eG agent is running. Normally, when you use an internal agent to monitor Informix, the CLIENTLOCALE and DBLOCALE will be same. However, the CLIENTLOCALE may differ when you are monitoring the Informix tests using a remote agent (agentless monitoring). By default, the CLIENTLOCALE is set to "none", indicating that the eG agent runs in the English locale, by default.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Informix server being monitored</td></tr>
<tr><td rowspan="5">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total size:
The total size of all logical log files.</td><td>Pages</td><td></td></tr>
<tr><td>Total free space:
The total space available in all logical log files.</td><td>Pages</td><td></td></tr>
<tr><td>Percent used space:
The percentage of space used in all logical log files.</td><td>Percent</td><td></td></tr>
<tr><td>Current log file size:
The size of the log file currently in use.</td><td>Pages</td><td></td></tr>
<tr><td></td><td>Free pages in current log:
The space available in the current log file.</td><td>Pages</td><td></td></tr>
</table>

	Current log used space: The percentage of space used in the current log file.	Percent	
	Logical log buffer records: The rate at which transaction log records were written to the logical log buffer.	Records/Sec	
	Logical log buffer writes: The rate at which log pages were written to the logical log buffer.	Pages/Sec	
	Logical log writes: The rate at which logical log buffers were written to the logical log files.	Writes/Sec	

7.2.4 Informix Physical Logs Test

This test collects and reports measures related to the physical logs of an Informix database server.

Purpose	Collects and reports measures related to the physical logs of an Informix database server
Target of the test	An Informix Dynamic server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Informix Dynamic server
3. PORT – The port on which the server is listening
4. INSTANCE - The Informix server instance being monitored
5. USER – A valid Informix user name
6. PASSWORD – The password corresponding to the above user
7. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
8. DBLOCALE - Specify the language/locale in which your database is installed. This will allow you to return the query in the language supported by your database. By default this attribute is set to "none", which means that by default, queries are returned in the English locale.
9. CLIENTLOCALE - Specify the language/locale in which the eG agent is running. Normally, when you use an internal agent to monitor Informix, the CLIENTLOCALE and DBLOCALE will be same. However, the CLIENTLOCALE may differ when you are monitoring the Informix tests using a remote agent (agentless monitoring). By default, the CLIENTLOCALE is set to "none", indicating that the eG agent runs in the English locale, by default.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Informix server being monitored</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Physical log buffer writes:
The rate at which pages were written to the physical log buffer.</td><td>Pages/Sec</td><td></td></tr>
<tr><td>Physical log writes:
The rate at which physical log buffers were written to the physical log files.</td><td>Writes/Sec</td><td></td></tr>
<tr><td>Checkpoints:
The number of checkpoints occurred.</td><td>Checkpoints/Sec</td><td></td></tr>
<tr><td></td><td>Checkpoint waits:
The number of times the threads waited for a checkpoint to finish to enter a critical section during a checkpoint.</td><td>Waits/Sec</td><td></td></tr>
</table>

7.3 The IFX Chunks Layer

A chunk is the largest unit of physical disk dedicated to database server data storage. Using the IfxChunk test associated with it, the **IFX Chunks** layer monitors the space usage and I/O activity on the chunks.

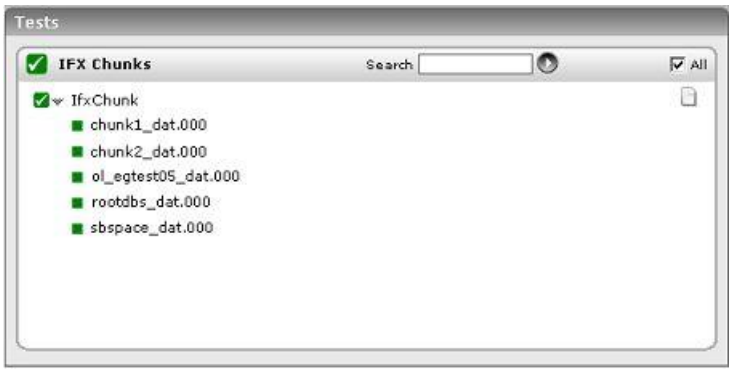


Figure 7.4: The tests associated with the IFX Chunks layer

7.3.1 Informix Chunks Test

This test collects the performance statistics pertaining to the disk space usage and disk I/O of a chunk.

Purpose	Reports statistics pertaining to the disk space and disk I/O of a chunk
Target of the test	An Informix Dynamic server
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Informix Dynamic server 3. **PORT** – The port on which the server is listening 4. **INSTANCE** - The Informix server instance being monitored 5. **USER** – A valid Informix user name 6. **PASSWORD** – The password corresponding to the above user 7. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 8. **DBLOCALE** - Specify the language/locale in which your database is installed. This will allow you to return the query in the language supported by your database. By default this attribute is set to "none", which means that by default, queries are returned in the English locale. 9. **CLIENTLOCALE** - Specify the language/locale in which the eG agent is running. Normally, when you use an internal agent to monitor Informix, the **CLIENTLOCALE** and **DBLOCALE** will be same. However, the **CLIENTLOCALE** may differ when you are monitoring the Informix tests using a remote agent (agentless monitoring). By default, the **CLIENTLOCALE** is set to "none", indicating that the eG agent runs in the English locale, by default.

Outputs of the test	One set of results for every chunk in an Informix database server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Chunk size: The total size of a chunk.	Pages	
	Free pages: The amount of unused space in a chunk.	Pages	
	Percent used: The percentage of space used in a chunk.	Percent	
	Status: The status of a chunk.	Number	Possible values are: ▪ 0 - Normal ▪ 1 - Recovering ▪ 2- Offline ▪ 3- Inconsistent
	Disk reads: The rate of disk reads.	Reads/Sec	
	Disk writes: The rate of disk writes.	Writes/Sec	
	Pages read: The number of pages read from the chunk per second.	Pages/Sec	
	Pages written: The number of pages written to the chunk per second.	Pages/Sec	

a.

7.4 The IFX Data Spaces Layer

A dbspace is a logical unit used to store databases, tables, logical log files and physical log files. It contains one or more chunks physically. The IfxDbSpace test, which is mapped to the **IFX Data Spaces** layer, monitors the dbspaces on the Informix server, and reports the size of each dbspace in terms of the number of chunks it constitutes, and the space usage and I/O activity on every dbspace.

Figure 7.5: The tests associated with the IFX Data Spaces layer

7.4.1 Informix Database Space Test

This test reports the space details of a dbspace.

Purpose	Reports the space usage of a dbspace		
Target of the test	An Informix Dynamic server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Informix Dynamic server 3. **PORT** – The port on which the server is listening 4. **INSTANCE** - The Informix server instance being monitored 5. **USER** – A valid Informix user name 6. **PASSWORD** – The password corresponding to the above user 7. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 8. **DBLOCALE** - Specify the language/locale in which your database is installed. This will allow you to return the query in the language supported by your database. By default this attribute is set to "none", which means that by default, queries are returned in the English locale. 9. **CLIENTLOCALE** - Specify the language/locale in which the eG agent is running. Normally, when you use an internal agent to monitor Informix, the **CLIENTLOCALE** and **DBLOCALE** will be same. However, the **CLIENTLOCALE** may differ when you are monitoring the Informix tests using a remote agent (agentless monitoring). By default, the **CLIENTLOCALE** is set to "none", indicating that the eG agent runs in the English locale, by default.		
Outputs of the test	One set of results for every dbspace in an Informix database server		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Total size:** The total size of a dbspace.	Pages	
	Free space: The amount of unused space in a particular dbspace.	Pages	If one of your dbspaces runs out of space, some applications could stop working.
	Percent used: The percentage of space used in a dbspace.	Percent	
	Number of chunks: The total number of chunks present in a specific Dbspace.	Number	
	Disk reads: The rate at which data pages were read from the disk.	Pages/Sec	
	Disk writes: The rate at which data pages were read from the disk.	Pages/Sec	

7.5 The IFX Service Layer

The tests mapped to the **IFX Service** layer track how well the Informix server serves user requests.

Figure 7.6: The tests associated with the IFX Service layer

7.5.1 Informix Access Test

The IfxAccess test reports statistics pertaining to the sequential scans and table sorts performed on an Informix database server.

<table>
<tr><td>Purpose</td><td colspan="3">Reports statistics pertaining to the sequential scans and table sorts performed on an Informix database server</td></tr>
<tr><td>Target of the test</td><td colspan="3">An Informix Dynamic server</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An internal agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Informix Dynamic server
3. PORT – The port on which the server is listening
4. INSTANCE - The Informix server instance being monitored
5. USER – A valid Informix user name
6. PASSWORD – The password corresponding to the above user
7. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
8. DBLOCALE - Specify the language/locale in which your database is installed. This will allow you to return the query in the language supported by your database. By default this attribute is set to “none”, which means that by default, queries are returned in the English locale.
9. CLIENTLOCALE - Specify the language/locale in which the eG agent is running. Normally, when you use an internal agent to monitor Informix, the CLIENTLOCALE and DBLOCALE will be same. However, the CLIENTLOCALE may differ when you are monitoring the Informix tests using a remote agent (agentless monitoring). By default, the CLIENTLOCALE is set to “none”, indicating that the eG agent runs in the English locale, by default.
10. DETAILED DIAGNOSIS - To make diagnosis more efficient and accurate, the eG Enterprise system 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 against DETAILED DIAGNOSIS. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Informix server being monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Sequential scans:
The number of sequential scans performed per second.</td><td>Scans/Sec</td><td>The detailed diagnosis of this measure, if enabled, will list the top 10 tables performing more number of sequential scans.</td></tr>
</table>

	Memory sorts: The percentage of sorts done in memory.	Percent	This value must always be high.
	Disk sorts: The number of disk sorts performed per second.	Sorts/Sec	This value must always be high.

7.5.2 Informix Response Test

This test reports the availability and response time of an Informix database server.

Purpose	Reports the availability and response time of an Informix database server		
Target of the test	An Informix Dynamic server		
Agent deploying the test	An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the target Informix server is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the target Informix server is listening.		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Informix Dynamic server 3. **PORT** – The port on which the server is listening 4. **DB** – The name of the database to be monitored 5. **INSTANCE** - The Informix server instance being monitored 6. **USER** – A valid user name to login to the specified database 7. **PASSWORD** – The password corresponding to the above user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 9. **QUERY** - The select query to execute. The default query is 'select * from sysprofile'. 10. **DBLOCALE** - Specify the language/locale in which your database is installed. This will allow you to return the query in the language supported by your database. By default this attribute is set to "none", which means that by default, queries are returned in the English locale. 11. **CLIENTLOCALE** - Specify the language/locale in which the eG agent is running. Normally, when you use an internal agent to monitor Informix, the **CLIENTLOCALE** and **DBLOCALE** will be same. However, the **CLIENTLOCALE** may differ when you are monitoring the Informix tests using a remote agent (agentless monitoring). By default, the **CLIENTLOCALE** is set to "none", indicating that the eG agent runs in the English locale, by default.		
Outputs of the test	One set of results for every Informix database server		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Availability:** The availability of the database server.	Percent	The availability is 100% when the server is responding to a request, and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the database server, or if the server has not been started.
	Response time: The time taken by the database to respond to a user query.	Secs	A sudden increase in response time is indicative of a bottleneck at the database server.

7.5.3 Informix Sessions Test

This test reports the session related information of an Informix database server.

Purpose	Reports the session related information of an Informix database server
Target of the test	An Informix Dynamic server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Informix Dynamic server
3. PORT – The port on which the server is listening
4. INSTANCE - The Informix server instance being monitored
5. USER – A valid Informix user name
6. PASSWORD – The password corresponding to the above user
7. CONFIRM PASSWORD – Confirm the PASSWORD by retyping it here.
8. DBLOCALE - Specify the language/locale in which your database is installed. This will allow you to return the query in the language supported by your database. By default this attribute is set to "none", which means that by default, queries are returned in the English locale.
9. CLIENTLOCALE - Specify the language/locale in which the eG agent is running. Normally, when you use an internal agent to monitor Informix, the CLIENTLOCALE and DBLOCALE will be same. However, the CLIENTLOCALE may differ when you are monitoring the Informix tests using a remote agent (agentless monitoring). By default, the CLIENTLOCALE is set to "none", indicating that the eG agent runs in the English locale, by default.
10. DETAILED DIAGNOSIS - To make diagnosis more efficient and accurate, the eG Enterprise system 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 against DETAILED DIAGNOSIS. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every Informix database server</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Current sessions:
The total number of current user sessions.</td><td>Number</td><td>The detailed diagnosis of the Current sessions measure, if enabled, will provide the following information about each user session:
▪ Session id
▪ User name
▪ Database name
▪ Host name of the user
▪ Session start time
Note: This list will contain only recently connected 100 user sessions.</td></tr>
</table>

	Blocked sessions: The number of sessions waiting for various database objects.	Number	

7.5.4 Informix Transactions Test

This test reports the transaction related statistics of an Informix database server.

Purpose	The transaction related statistics of an Informix database server		
Target of the test	An Informix Dynamic server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Informix Dynamic server 3. **PORT** – The port on which the server is listening 4. **INSTANCE** - The Informix server instance being monitored 5. **USER** – A valid Informix user name 6. **PASSWORD** – The password corresponding to the above user 7. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 8. **DBLOCALE** - Specify the language/locale in which your database is installed. This will allow you to return the query in the language supported by your database. By default this attribute is set to "none", which means that by default, queries are returned in the English locale. 9. **CLIENTLOCALE** - Specify the language/locale in which the eG agent is running. Normally, when you use an internal agent to monitor Informix, the **CLIENTLOCALE** and **DBLOCALE** will be same. However, the **CLIENTLOCALE** may differ when you are monitoring the Informix tests using a remote agent (agentless monitoring). By default, the **CLIENTLOCALE** is set to "none", indicating that the eG agent runs in the English locale, by default.		
Outputs of the test	One set of results for every Informix database server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Commits: The rate of user commits.	Commits/Sec	
	Rollbacks: The rate of user rollbacks.	Rollbacks/Sec	

	Long transactions: The number of long running transactions.	Number	Long running transactions should be avoided. They consume major resources of your database server. If a long transaction exclusive high water mark(LTXEHWM) is reached, all other write activities will be blocked.

8

Monitoring the Intersystems Cache Database

CACHÉ is a multidimensional database that combines robust objects and robust SQL, thus eliminating object-relational mapping. Cache enables rapid web application development, quick transaction processing, scalability, and real-time queries against transactional data.

Caché is a post-relational database - the "relational" part of "post-relational" refers to the fact that Caché is a full-featured relational database. All the data within a Caché database is available as true relational tables and can be queried and modified using standard SQL via ODBC, JDBC, or object methods. The "post" part of "post-relational" refers to the fact that Caché offers a range of features that go beyond the limits of relational databases, while still supporting a standard relational view of data. These features include:

- The ability to model data as objects (each with an automatically created and synchronized native relational representation) while eliminating both the impedance mismatch between databases and object-oriented application environments as well as reducing the complexity of relational modeling.
- A simpler, object-based concurrency model.
- User-defined data types.
- The ability to take advantage of methods and inheritance, including polymorphism, within the database engine.
- Object-extensions for SQL to handle object identity and relationships.
- The ability to intermix SQL and object-based access within a single application, using each for what they are best suited.
- Control over the physical layout and clustering used to store data in order to ensure the maximum performance for applications.

Another unique feature of Cache is its Unified Data Architecture. Whenever a database object class is defined, Caché automatically generates a SQL-ready relational description of that data. Similarly, if a DDL description of a relational database is imported into the Data Dictionary, Caché automatically generates both a relational and an object description of the data, enabling immediate access to objects.

At the core of the Cache architecture, is the Cache Multi-dimensional Database Engine that provides the complete set of services—including data storage, concurrency management, transactions, and process management—needed to build complex database management systems. Data can be stored and accessed from the database engine using objects, SQL, or through direct access to multidimensional structures. Figure 8.1 depicts the access methodologies.

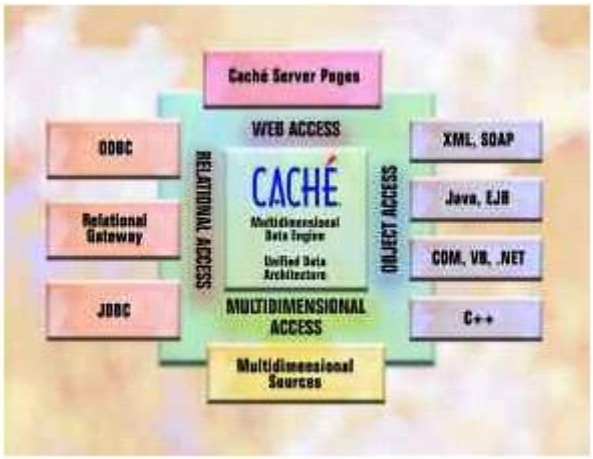


Figure 8.1: Access to the data stored by the data engine

Regardless of the access method, all data in Cache's database is stored in Cache's multidimensional arrays. These multi-dimensional arrays enable efficient data storage; further, the absence of joins and table-hopping hastens query execution. In addition, its ability to distribute data and application logic across multiple server systems, simplifies the development of reliable applications. Owing to these capabilities, Cache is a key ingredient in the design and delivery of a wide variety of applications ranging from single-user embedded systems to large, multi-server, multi-user installations (such as those required by banks, hospitals, etc.) providing essential services to end-users.

This dependence on Cache for performing business-critical tasks and for developing mission-critical applications could only mean that even a wafer-thin deviation in its performance could cause an enterprise to lose millions. Database administrators are thus faced with the daunting task of ensuring the 24x7 availability of the Cache database and the optimal performance of all its components.

eG Enterprise offers a specialized monitoring model for the Cache database (see Figure 8.2) that monitors the database 24 x 7 and proactively alerts administrators of probable issues in its operations, so that issues are trapped very early and resolved before its too late.

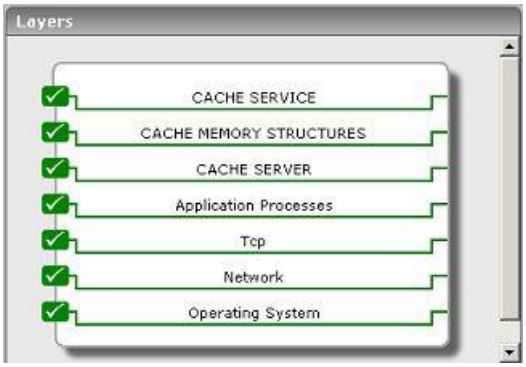


Figure 8.2: Layer model of the Cache database server

Each of the layers depicted by the hierarchical model above, is associated with a wide variety of tests that measure the health of various factors influencing database performance. The performance metrics reported by these tests shed light on the following:

Cache Service/Process Monitoring	➢ Is the cache instance up and running? ➢ How responsive is the cache database server to client requests? ➢ Are the critical processes running? What is their resource usage?

Cache System/ Database Monitoring	➢ What is the current status of the database - read-only or read-write? ➢ Is there sufficient free space in the Cache databases? ➢ Are there any severe/fatal errors logged in the console log? ➢ Is license usage optimal? ➢ Is data adequately referenced so as to enable easy retrieval? ➢ Is global referencing normal or are there way too many or too little global accesses? ➢ Have too many locks been created on any database instance? If so, what are the lock types? Are there any potentially dangerous lock types? ➢ Are too many processes contending for a lock on the database? ➢ Are resource seizes occurring? If so, what types of seizes are they? ➢ Were any critical changes made to the database, recently? ➢ Was the write daemon able to write all changes to the database? ➢ What is the current status of the write daemon?
Ecp Application/Data Server Monitoring	➢ Is data transmitted properly from the application and data servers? ➢ Are the global references affecting database performance? ➢ Do the requests from data servers affect traffic?
Cache Performance/Buffer Monitoring	➢ What is the workload of the data server in terms of lines & routine executed? ➢ Has sufficient memory been allocated to the buffers to reduce physical disk IO? ➢ Have the routine and database caches been adequately sized?

While the eG agent extracts some of these critical statistics from the SNMP MIB of Cache (eg., metrics related to free space in the database, buffer usage, ECP application and data servers, etc.), for a few other key metrics (eg., metrics related to the console log, resource seizes, lock counts, etc.) the agent executes the **cstat** utility that is provided with Caché. This utility, which is available in the install directory of a Cache instance, provides a wealth of information about the running Caché system.

Where the eG agents needs to contact the SNMP MIB for performance data, make sure that the Windows SNMP service is installed and started either automatically or manually on the Cache host. In addition, ensure the following:

- The **Cache Monitoring Service** should be enabled
- Activate critical SNMP base classes, so that eG agent can monitor some of the most significant aspects of Cache performance
- Configure the Cache SNMP agent to start automatically at Cache startup

Follow the steps given below to enable the **Cache Monitoring Service**:

1. Navigate to the Home -> Security Management -> Services page of the System Management Portal.
2. Click the **%System_Monitor** service in the **Services** page (see Figure 8.3).

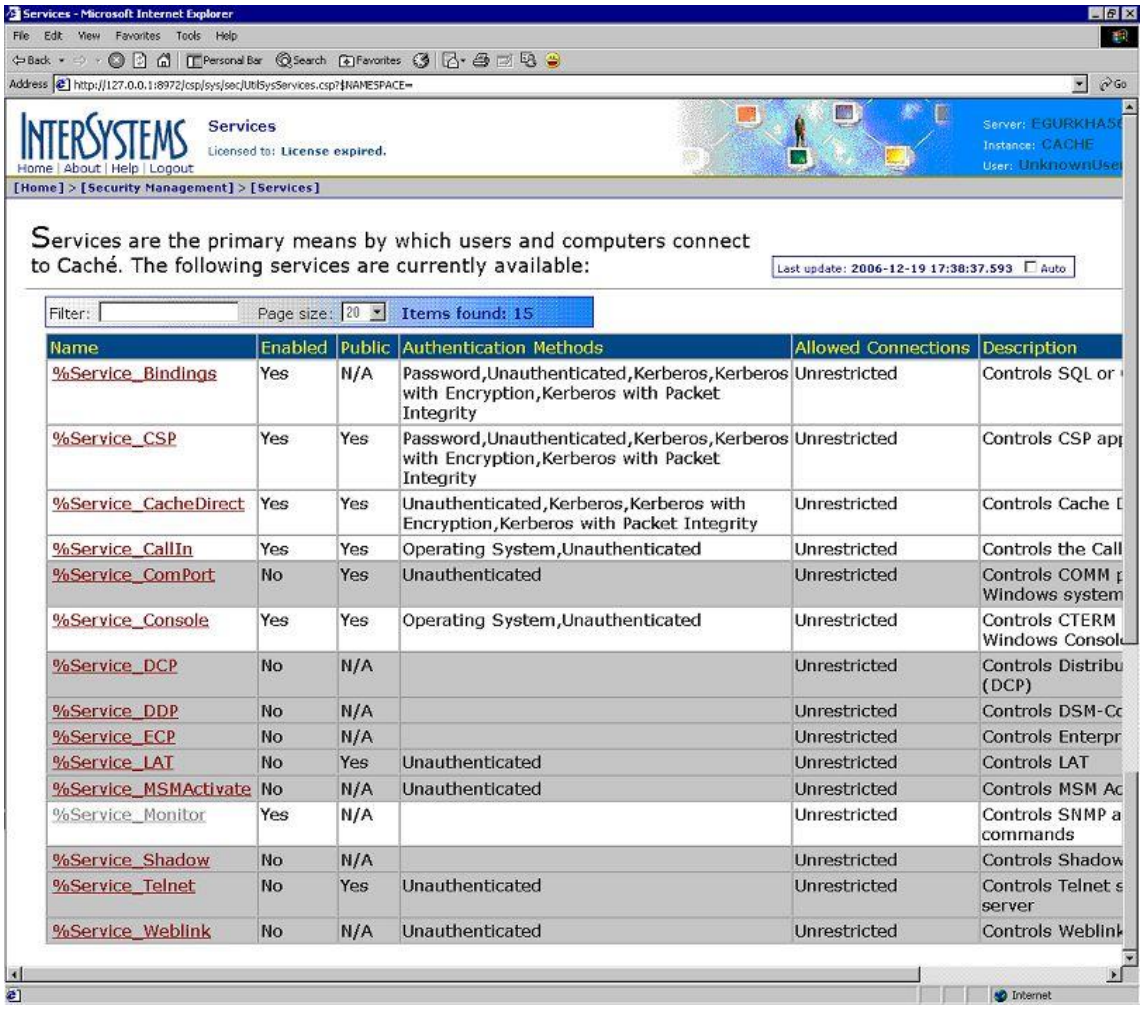


Figure 8.3: The Services page

3. Figure 8.4 then appears. The **Service enabled** check box will be deselected, by default. Click on the check box to select it and click the **Save** button to register the changes.

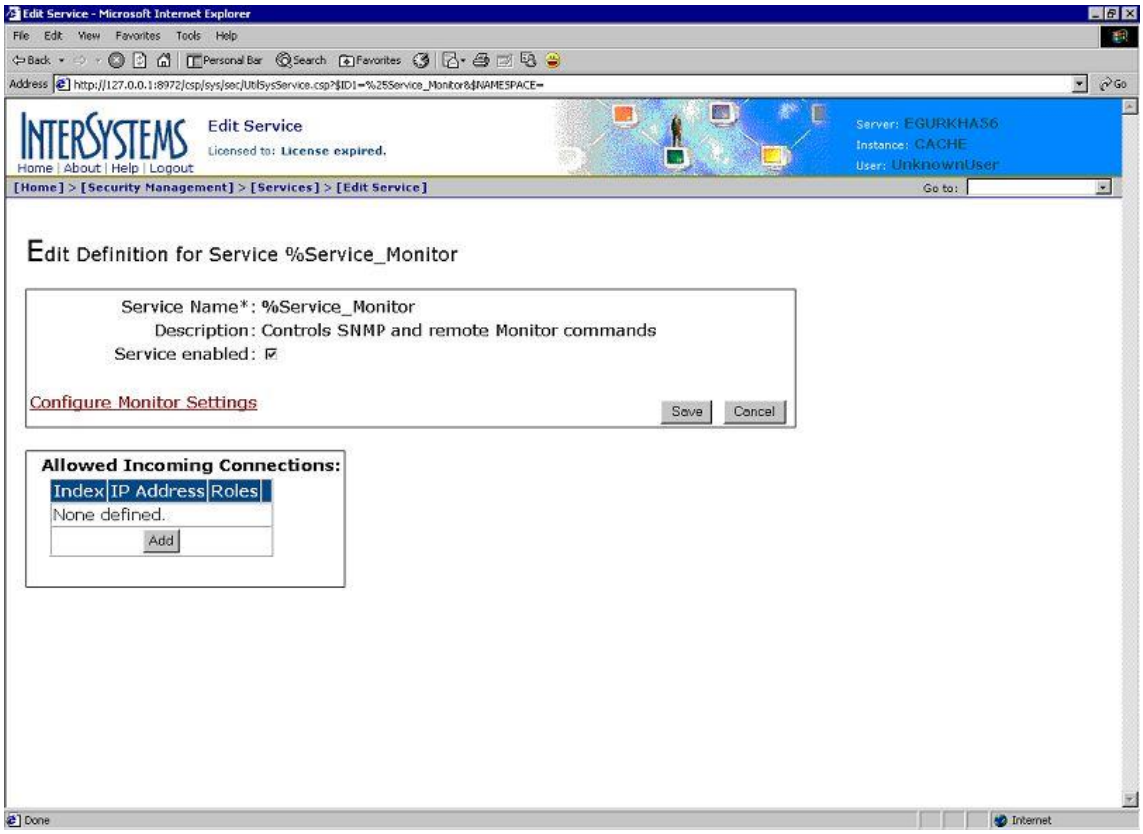


Figure 8.4: Enabling service monitoring

4. You will then return to the **Services** page, where you can verify whether the **%System_Monitor** service is enabled or not.

To enable the SNMP base classes that monitor critical aspects of Cache performance, do the following:

1. Start the Cache Terminal.
2. From the command prompt of the Terminal, issue the command, **DO ^%CD**, to switch to a different namespace.

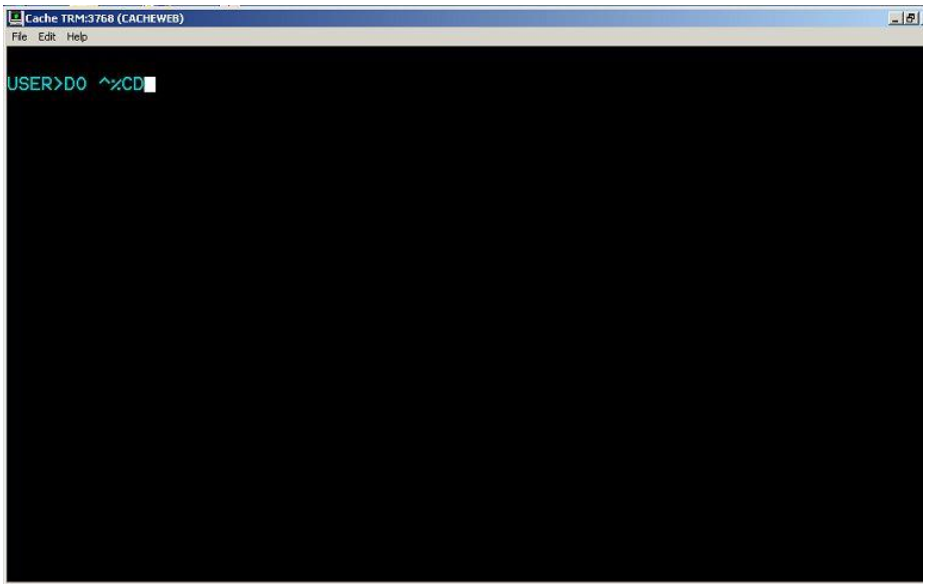


Figure 8.5: Switching to a different namespace

3. You will now be prompted for a namespace. Type **%SYS** therein.

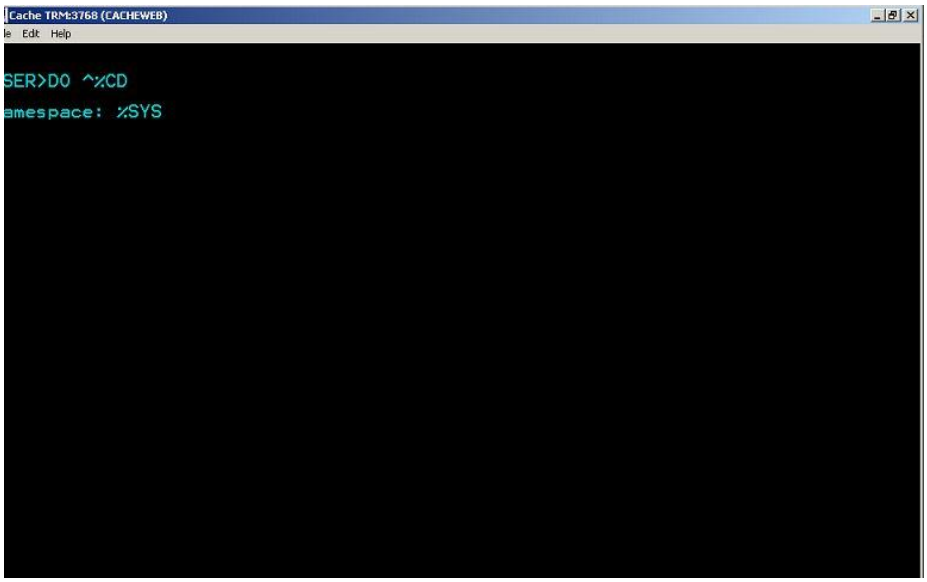


Figure 8.6: Specifying the Namespace to switch to

4. Execute the routine **%MONAPPMGR** as indicated by Figure 8.7.

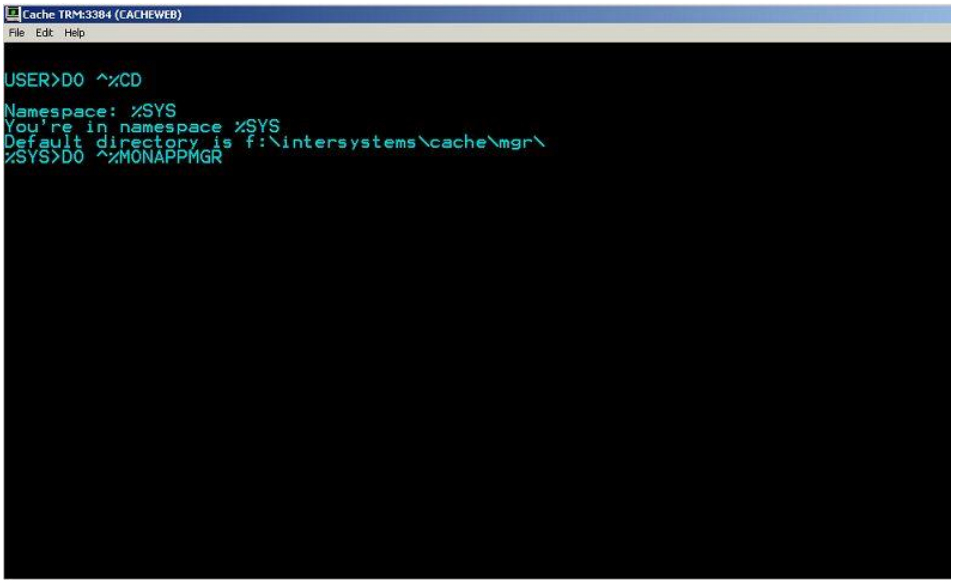


Figure 8.7: Executing the routine %MONAPPMGR

5. Upon execution, the routine requests you to select from 4 options. Select the **Manage Monitor Classes** option, by pressing **2**.

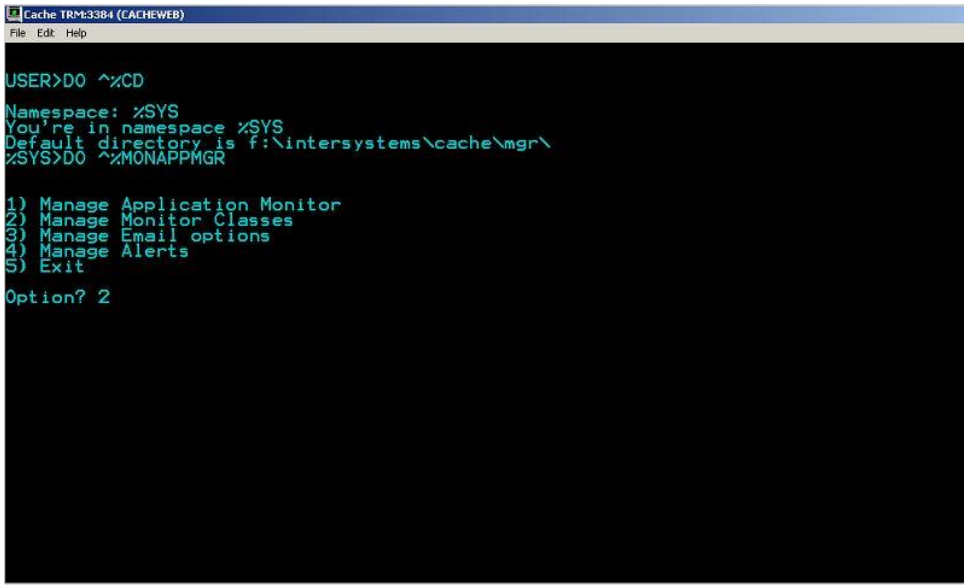


Figure 8.8: Selecting the Manage Monitor Classes option

6. You will now be prompted to choose the action that you want to perform on the Monitor Classes. Since we need to activate a monitoring capability (i.e., Freespace monitoring), select the **Activate/Deactivate a Monitor Class** option, by pressing **1** (see Figure 8.9).

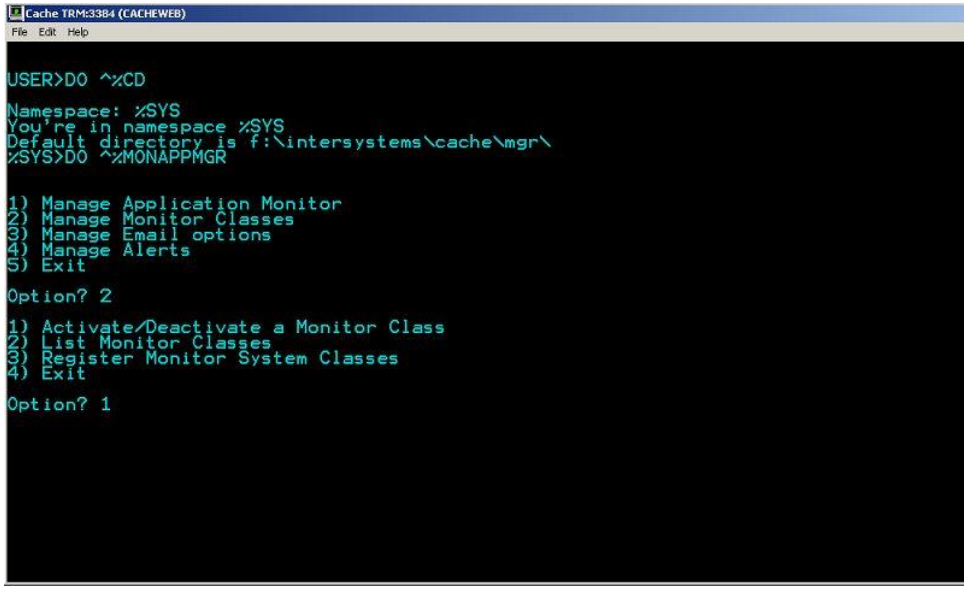


Figure 8.9: Choosing to activate/deactivate a monitor class

7. When prompted for the monitor class to activate/deactivate, specify **%Monitor.System.Freespace** as indicated by Figure 8.10. This class monitors the free space on every Cache database.

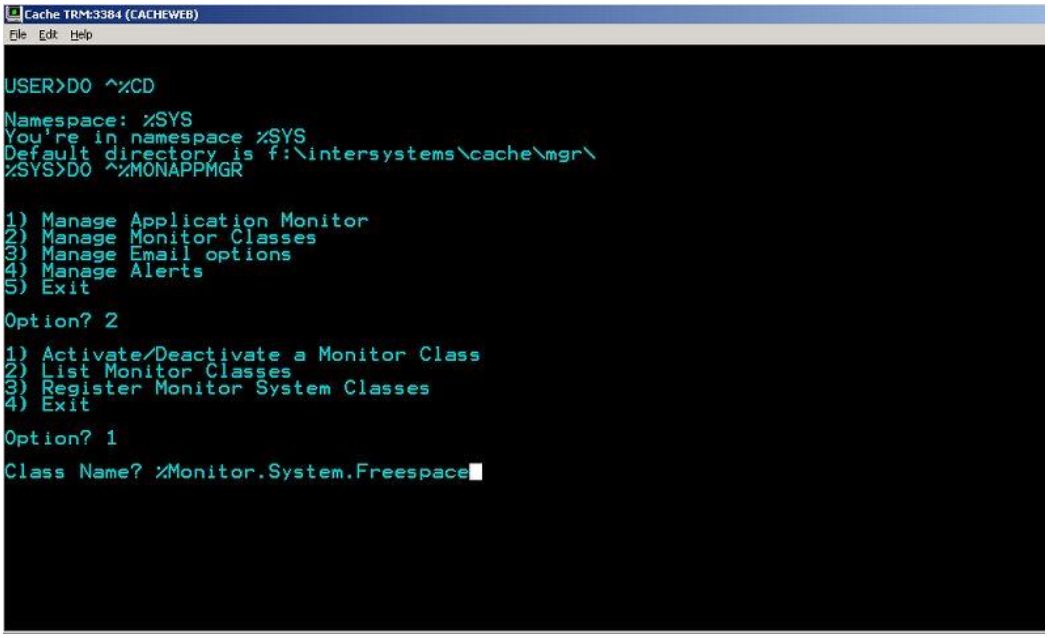


Figure 8.10: Specifying the name of the class to activate/deactivate

8. Then, choose to **Activate** the **%Monitor.System.Freespace** class by pressing **A** (see Figure 8.11).

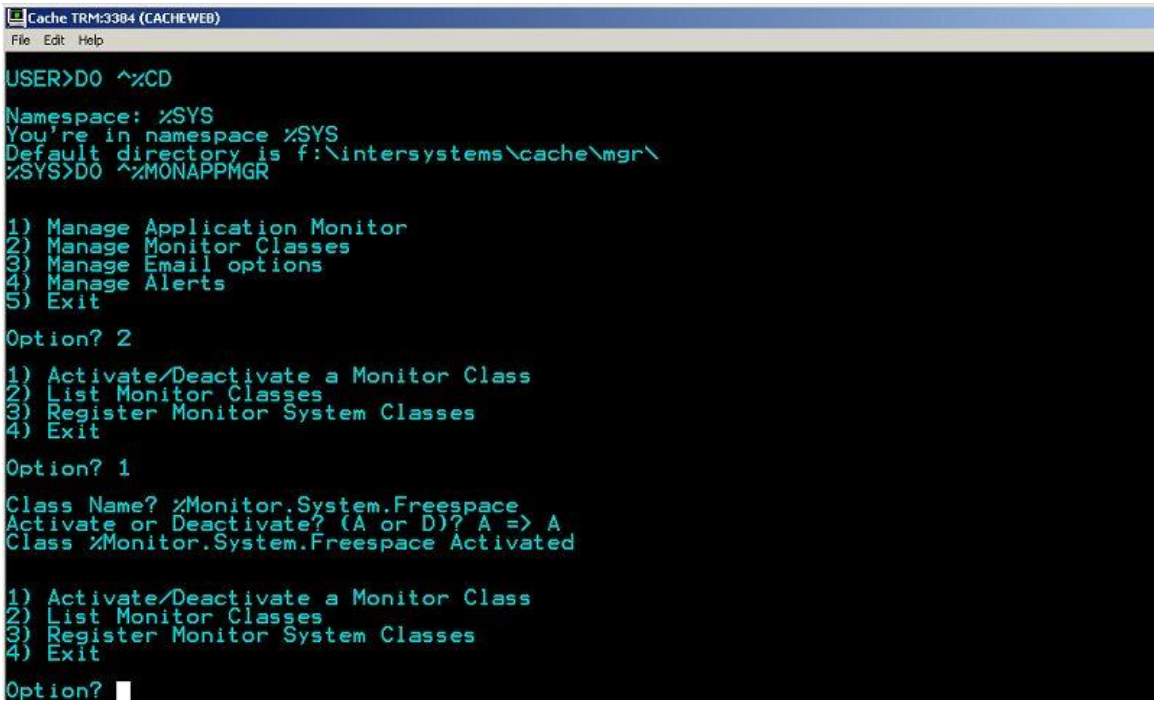


Figure 8.11: Activating the Monitor Class

9. Upon successful activation, a message to that effect will be displayed, followed by a list of options. Press **1** this time to activate another class.

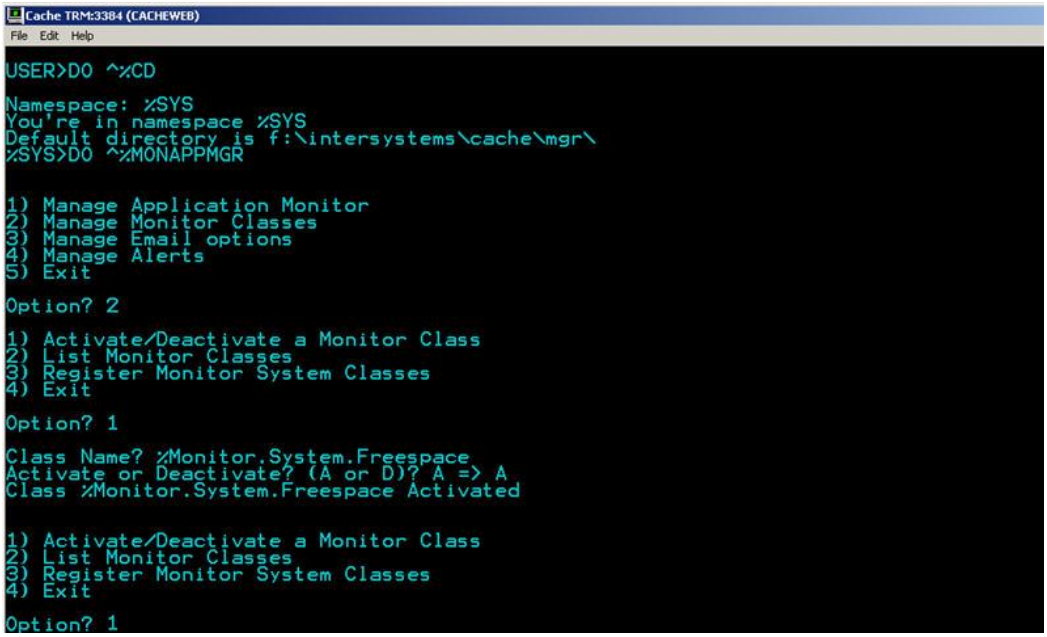


Figure 8.12: Selecting another class for activation

10. This time, activate the **%Monitor.System.Database** class so as to enable Cache database monitoring (see Figure 8.13).

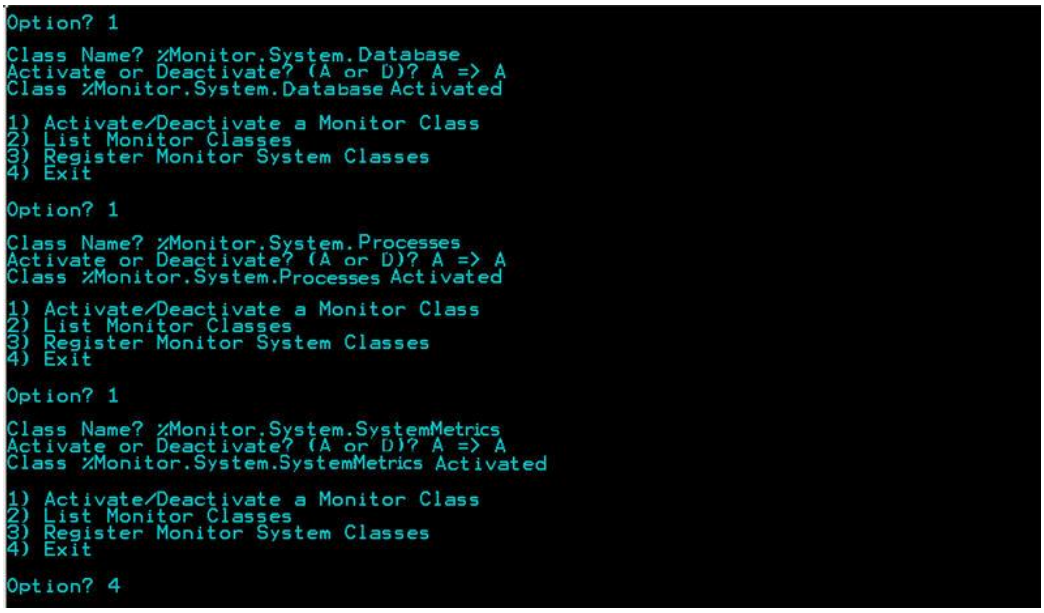


Figure 8.13: Activating the Database, Processes, and SystemMetrics class

11. Subsequently, activate the **%Monitor.System.Processes** and **%Monitor.System.SystemMetrics** classes (see Figure 8.108), so that the eG agent is able to extract Cache process-related metrics and system-related metrics.
12. Finally, exit the Cache Terminal by selecting option number **4** as indicated by Figure 8.108.

To configure the SNMP sub-agent to start automatically on startup, do the following:

1. Navigate to the Home -> Configuration -> Monitor Settings page of the System Management Portal (see Figure 8.14).

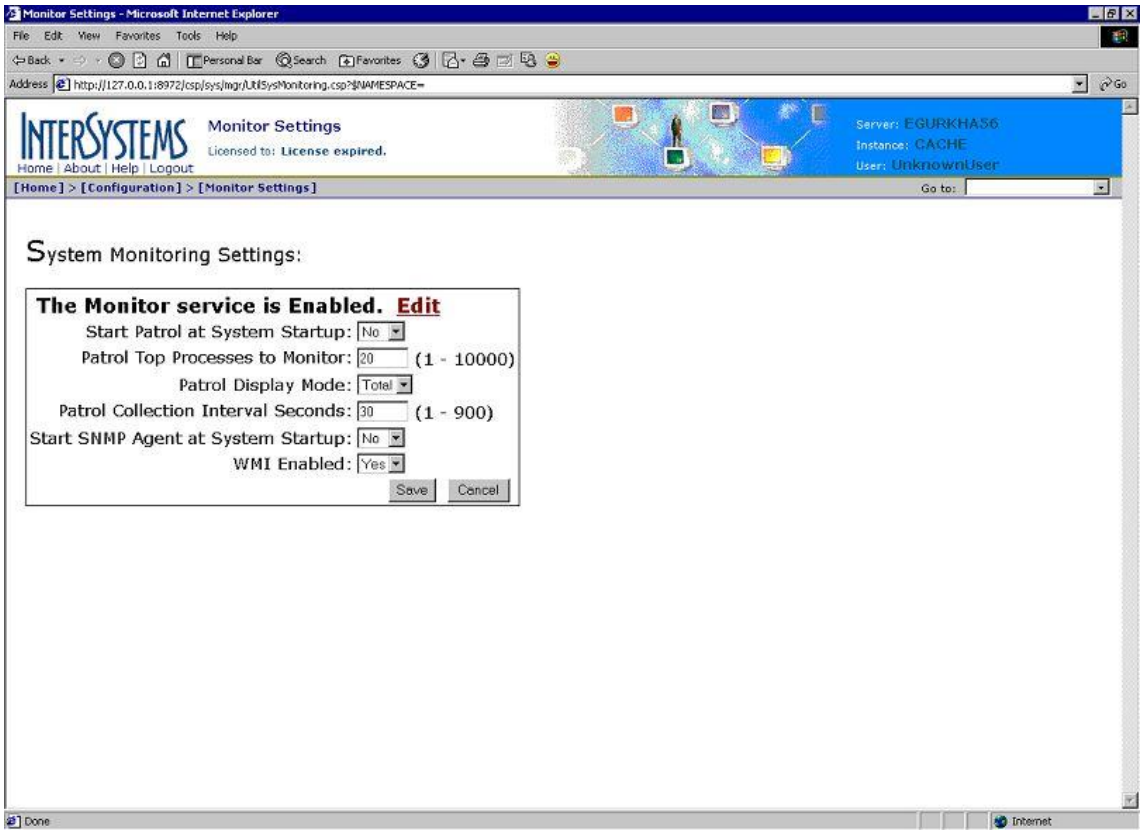


Figure 8.14: The Monitor Settings page of the System Management Portal

2. Set the **Start SNMP Agent at System Startup** flag to **Yes** (see Figure 8.15), and click the **Save** button to register the changes.

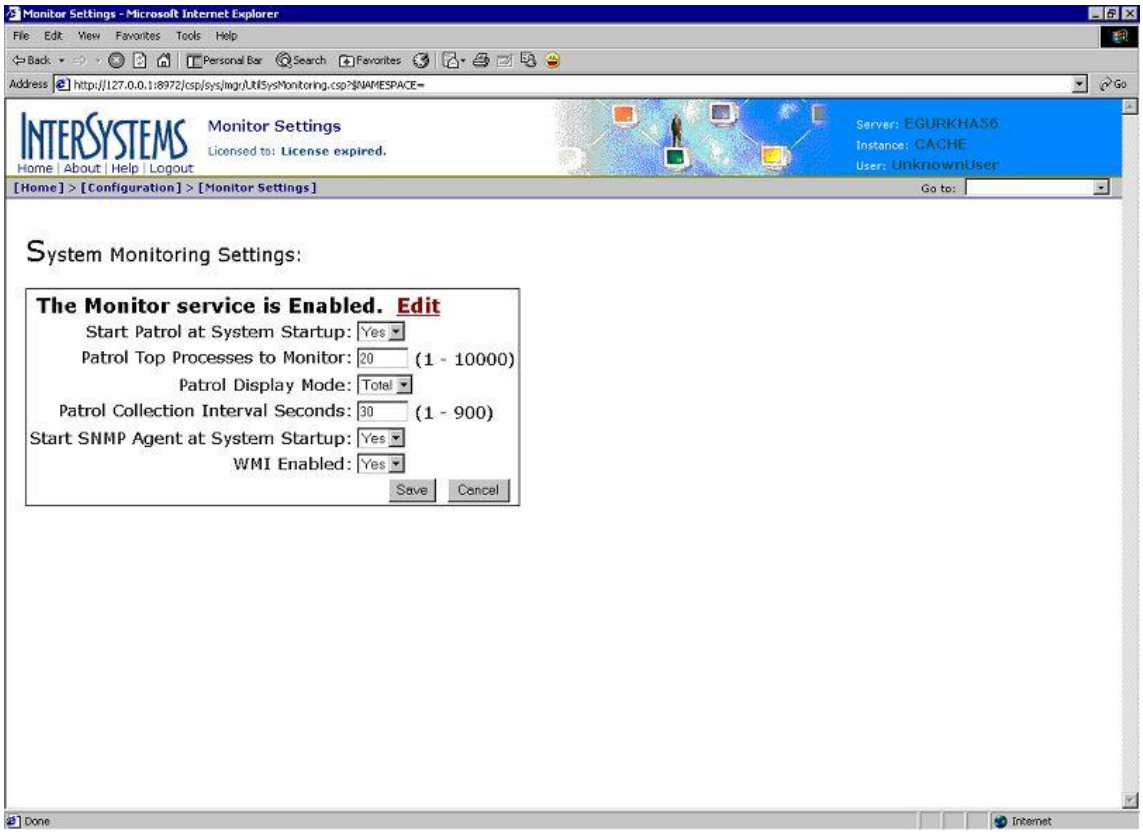


Figure 8.15: Enabling the SNMP Agent to start on system boot

3. Next, start the Cache Terminal.
4. From the command prompt of the Terminal, issue the command, **DO ^%CD**, to switch to a different namespace.
5. You will now be prompted for a namespace. Type **%SYS** therein, as only this namespace is vested with the right to start the SNMP agent service.

```
NAMESPACE: %SYS
```

6. The namespace then changes to %**SYS**.

```
Default directory is d:\cache\cachesys\mgr\

%SYS>
```

7. Next, execute the routine that would start the SNMP agent.

```
DO start^SNMP(705,20)
```

On the other hand, where the eG agent needs to execute **cstat** for collecting the required metrics, you need to ensure that the corresponding tests are configured with the exact Cache instance name and the correct path to that instance's home directory.

The eG agent then collects the metrics of interest from the SNMP MIB/cstat (as the case may be), and reports the performance data so collected to the eG manager. The manager then displays the data in the user interface using the layer model represented by Figure 8.2. The sections to come will elaborate on the tests that are executed on the top three layers of the monitoring model only.

8.1 The Cache Server Layer

The tests associated with the **Cache Server** layer reports a wide variety of statistics related to the following:

- Space usage of each of the Cache databases
- The number and nature of errors logged in the console log file
- Key system and application processes executing on the Cache instances
- Global activity and disk I/O on configured Cache instances
- License usage by the Cache server
- Health of the ECP application and data servers

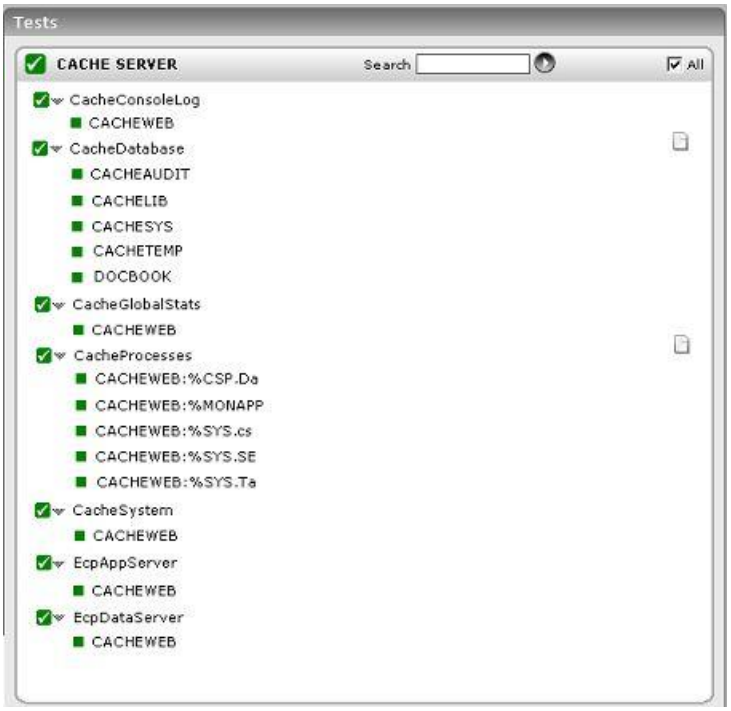


Figure 8.16: Tests associated with the Cache Server layer

8.1.1 Cache Database Test

This test monitors the space usage of every Cache database using SNMP.

Purpose	Monitors the usage of every database instance of the Cache server
Target of the test	A Cache Database server
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Cache database server 3. **PORT** – The port on which the server is listening 4. **SNMPPORT** - The port number through which the target server exposes its SNMP MIB. The default value is 161. 5. **SNMPVERSION** – By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the **SNMPVERSION** list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list. 6. **SNMPCOMMUNITY** – The SNMP community name that the test uses to communicate with the target server. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the **SNMPVERSION** chosen is **v3**, then this parameter will not appear. 7. **USERNAME** – This parameter appears only when **v3** is selected as the **SNMPVERSION**. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against the **USERNAME** parameter. 8. **AUTHPASS** – Specify the password that corresponds to the above-mentioned **USERNAME**. This parameter once again appears only if the **SNMPVERSION** selected is **v3**. 9. **CONFIRM PASSWORD** – Confirm the **AUTHPASS** by retyping it here. 10. **AUTHTYPE** – This parameter too appears only if **v3** is selected as the **SNMPVERSION**. From the **AUTHTYPE** list box, choose the authentication algorithm using which SNMP v3 converts the specified **USERNAME** and **PASSWORD** into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm 11. **ENCRYPTFLAG** – This flag appears only when **v3** is selected as the **SNMPVERSION**. By default, the eG agent does not encrypt SNMP requests. Accordingly, the **ENCRYPTFLAG** is set to **NO** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **YES** option. 12. **ENCRYPTTYPE** – If the **ENCRYPTFLAG** is set to **YES**, then you will have to mention the encryption type by selecting an option from the **ENCRYPTTYPE** list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard 13. **ENCRYPTPASSWORD** – Specify the encryption password here. 14. **CONFIRM PASSWORD** – Confirm the encryption password by retyping it here.
Outputs of the test	One set of results for every database on the Cache database server

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Read only status: Indicates the current status of the database - whether read-only or not	Boolean	The value '1' indicates that the database is currently in the "read-only" status, and the value '0' indicates that the database can be read and written into - i.e., 'read-write' status.
	Current database size: The current size of this database	KB	
	Free space: The amount of unused space currently available in this database	KB	
	Percent free space: The percentage of free space currently available in this database	Percent	Ideally, this value should be very high.

8.1.2 Cache Console Log Test

The primary information source for monitoring Caché is the console log (cconsole.log). Caché reports general messages, system errors, certain operating system errors, and network errors through an operator console facility. The console log file is a plain text file and may be viewed with any editor or text viewer. It is found in the MGR subdirectory of the location where Caché was installed. The CacheConsoleLogTest periodically monitors the console log of every configured Cache instance to report the number of normal, severe, and fatal errors encountered by the Cache database server.

Purpose	Monitors the console log of every configured Cache instance to report the number of normal, severe, and fatal errors encountered by the Cache database server
Target of the test	A Cache Database server
Agent deploying the test	An internal/remote agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Cache database server
3. PORT – The port on which the server is listening
4. INSTANCEDIRECTORY - Typically, the Cache console log file will be available in the install directory of a Cache instance. Therefore, in the INSTANCEDIRECTORY text box, specify the name of the instance being monitored and the install directory that holds the Cache console log file of that instance, in the following format: InstanceName:InstallDirectory. In case you want to monitor the console log files pertaining to multiple Cache instances, then provide a comma-separated list of InstanceName:InstallDirectory pairs in the INSTANCEDIRECTORY text box. For example: CACHEWEB:d:\Intersystems\CacheWeb,CACHE2:d:\Intersystems\Cache2..</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every InstanceName configured</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total errors:
Indicates the number of normal errors/warnings that currently occurred on this Cache instance</td><td>Number</td><td>The detailed diagnosis of this measure, when enabled, provides the detailed description for each of the normal errors that occurred on this Cache instance. A typical normal error/warning would be, ***WARNING***: THE CURRENT CONFIGURATION IS NOT THE SAME AS WHEN CACHE WAS LAST SHUT DOWN.</td></tr>
<tr><td>Severe errors:
Indicates the number of severe errors that occurred currently on this Cache instance</td><td>Number</td><td rowspan="2">A typical severe error would be: SNMP server failed to start: error signaling Windows SNMP agent. Some <NETWORK> errors are essentially fatal error conditions. Typically, if email notification of errors is setup on a monitored Cache database server, then, whenever a severe/fatal error is logged in the cconsole.log file, Cache automatically sends out emails to concerned administrators alerting them to the problem condition and enabling them to initiate corrective action. Starting with v5.1 of the Cache database server, if no email notification mechanism is setup, then an additional log file, alerts.log, is created whenever a severe or fatal error is detected in the cconsole.log file. This log contains only the severe and fatal error messages. You can check this log file for details pertaining to such errors.

Alternatively, if you enable the detailed diagnosis capability of the eG Enterprise suite, then the complete description of the severe/fatal errors will be available to you in the eG monitoring console itself.</td></tr>
<tr><td>Fatal errors:
Indicates the current number of fatal errors on this Cache instance</td><td>Number</td></tr>
</table>

8.1.3 Cache Global Stats Test

This test gathers global activity statistics and displays a variety of information about disk I/O operations.

Purpose	Gathers global activity statistics and displays a variety of information about disk I/O operations		
Target of the test	A Cache Database server		
Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Cache database server 3. **PORT** – The port on which the server is listening 4. **INSTANCEDIRECTORY** - This test uses the cstat utility to collect the required metrics. This utility is typically run from the install directory of a monitored Cache instance. Therefore, in the **INSTANCEDIRECTORY** text box, specify the name of the instance being monitored and the install directory of that instance, in the following format: *InstanceName:InstallDirectory*. To ensure that the test reports metrics for multiple Cache instances, provide a comma-separated list of *InstanceName:InstallDirectory* pairs in the **INSTANCEDIRECTORY** text box. For example: *CACHEWEB:d:|Intersystems|CacheWeb,CACHE2:d:|Intersystems|Cache2*.		
Outputs of the test	One set of results for every *InstanceName* configured		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Global references: Indicates the logical count of accesses to globals, including Sets, Kills, $Data, $Order, $Increment, $Query, and global references in expressions, during the last measurement period.	Number	If this number significantly jumps up or declines from the normal, there may be an application issue to research.
	Global update references: Indicates the number of calls to a routine during the last measurement period.	Number	
	Routine calls: Indicates the current number of fatal errors on this Cache instance.	Number	

	Routine buffer loads and saves: Indicates the total number of routine loads and saves as a result of ZLoad, ZSave, and running routines, during the last measurement period.	Number	In a well-tuned environment, this number increases slowly, since most routine loads are satisfied by the routine cache memory without accessing the disk. Each routine load or save transfers up to 32 KB of data (64 KB for Unicode).
	Block I/O reads: Indicates the number of physical database blocks (2-KB or 8-KB) read from disk during the last measurement period for both global and routine references.	Number	A high value for this measure indicates that direct disk accesses are high. In such a case your database might require some fine-tuning. Consider resizing your buffer pool to increase buffer accesses and reduce data retrievals from the disk.
	Block I/O writes: Indicates the number of physical database blocks (2-KB or 8-KB) written to disk during the last measurement period for both global and routine references.	Number	
	WIJ I/O writes: Indicates the number of 64-KB journal blocks written to the journal file during the last measurement period.	Number	
	Logical block requests: Indicates the number of database blocks read by the global database code during the last measurement period.	Number	In a well-tuned environment, many of these reads are satisfied without disk access.

8.1.4 Cache Processes Test

This test auto-discovers the processes on every configured Cache instance, and reports the number of instances of each process that are currently running; this way, the test verifies whether critical system and application processes are running or not.

Purpose	Reports the number of instances of each process that are currently running on a Cache server

<table>
<tr><td>Target of the test</td><td colspan="3">A Cache Database server</td></tr>
<tr><td>Agent deploying the test</td><td colspan="3">An internal/remote agent</td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Cache database server
3. PORT – The port on which the server is listening
4. INSTANCEDIRECTORY - This test uses the cstat utility to collect the required metrics. This utility is typically run from the install directory of a monitored Cache instance. Therefore, in the INSTANCEDIRECTORY text box, specify the name of the instance being monitored and the install directory of that instance, in the following format: InstanceName:InstallDirectory. To ensure that the test reports metrics for multiple Cache instances, provide a comma-separated list of InstanceName:InstallDirectory pairs in the INSTANCEDIRECTORY text box. For example: CACHEWEB:d:|Intersystems|CacheWeb,CACHE2:d:|Intersystems|Cache2.
5. DETAILED DIAGNOSIS - To make diagnosis more efficient and accurate, the eG Enterprise system 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 against DETAILED DIAGNOSIS. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every InstanceName:Processname pair discovered</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Process count:
Indicates the number of instances of this Cache process currently running on this Cache instance.</td><td>Number</td><td>The detailed diagnosis of this measure, if enabled, provides the complete process details such as the namespace to which the process belongs, the lines of code executed by the process, the number of global references made by the process, and the device to which the process belongs.</td></tr>
</table>

8.1.5 Cache System Test

Using this test, you can monitor,

- the users to the database instance
- the database and routine caching activities performed by the Cache database instance
- the license usage of the database instance

- errors (if any) that have been logged

Purpose	Monitor the users to the database instance, database and routine caching activities performed by the Cache database instance, license usage of the database instance, errors (if any) that have been logged
Target of the test	A Cache Database server
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Cache database server 3. **PORT** – The port on which the server is listening 4. **SNMPPORT** - The port number through which the target server exposes its SNMP MIB. The default value is 161. 5. **SNMPVERSION** – By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the **SNMPVERSION** list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list. 6. **SNMPCOMMUNITY** – The SNMP community name that the test uses to communicate with the target server. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the **SNMPVERSION** chosen is **v3**, then this parameter will not appear. 7. **USERNAME** – This parameter appears only when **v3** is selected as the **SNMPVERSION**. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against the **USERNAME** parameter. 8. **AUTHPASS** – Specify the password that corresponds to the above-mentioned **USERNAME**. This parameter once again appears only if the **SNMPVERSION** selected is **v3**. 9. **CONFIRM PASSWORD** – Confirm the **AUTHPASS** by retyping it here. 10. **AUTHTYPE** – This parameter too appears only if **v3** is selected as the **SNMPVERSION**. From the **AUTHTYPE** list box, choose the authentication algorithm using which SNMP v3 converts the specified **USERNAME** and **PASSWORD** into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: ➢ **MD5** – Message Digest Algorithm ➢ **SHA** – Secure Hash Algorithm 11. **ENCRYPTFLAG** – This flag appears only when **v3** is selected as the **SNMPVERSION**. By default, the eG agent does not encrypt SNMP requests. Accordingly, the **ENCRYPTFLAG** is set to **NO** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **YES** option. 12. **ENCRYPTTYPE** – If the **ENCRYPTFLAG** is set to **YES**, then you will have to mention the encryption type by selecting an option from the **ENCRYPTTYPE** list. SNMP v3 supports the following encryption types: ➢ **DES** – Data Encryption Standard ➢ **AES** – Advanced Encryption Standard 13. **ENCRYPTPASSWORD** – Specify the encryption password here. 14. **CONFIRM PASSWORD** – Confirm the encryption password by retyping it here.

	15. **TIMEOUT** – Specify the duration (in seconds) beyond which the SNMP query issues by this test should time out. The default period is 10 seconds.		
Outputs of the test	One set of results for every instance of the Cache database server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Users connected to Cache: Indicates the number of users who are currently using this database instance.	Number	This is a good indicator of server load.
	Routine cache size: Indicates the current size of the routine cache	MB	Ideally, this value should be high. A well-sized routine cache can serve a large number of routine loads, thereby considerably reducing direct disk accesses.
	Database cache size: Indicates the current size of the database cache.	MB	Ideally, this value should be high. A well-sized database cache can serve a large number of requests, thereby considerably reducing direct disk accesses.
	Licenses in use in Cache: Indicates the number of licenses currently used by this database instance.	Number	
	Peak license usage: Indicates the high water-mark of the license usage for this database instance.	Number	An abnormally high value could be a cause for concern, and could hence require further investigation. Alternatively, this could also be an effective indicator of the popularity of the database instance, and might have to be considered while planning the future license requirements.
	Error detected in Cache: Indicates the last 'severe' error message logged in the console log for this Cache instance.	Boolean	

8.1.6 Ecp Application Server Test

One of the most powerful and unique features of Caché is the ability to efficiently distribute data and application logic among a number of server systems. The underlying technology behind this feature is the Enterprise Cache Protocol (ECP): a distributed data caching architecture that manages the distribution of data and locks among a heterogeneous network of server systems. Unlike other "multi-tier" architectures, ECP is primarily a configuration option. That is, you do not have to use special code or development techniques to create distributed database applications.

Furthermore, the architecture and operation of ECP is conceptually simple. ECP provides a way to efficiently share data, locks, and executable code among multiple Caché systems. Data and code are stored remotely, but are cached locally to provide efficient access with minimal network traffic.

An ECP configuration consists of a number Caché systems that are visible to one another across a TCP/IP-based network. There are two roles a Caché system can play in an ECP configuration:

- **ECP Data Server** — a Caché system that is providing data for one or more ECP application server systems.
- **ECP Application Server** — a Caché system that is consuming data provided by one or more ECP data server systems.

A Caché system can simultaneously act as both an ECP data server and an ECP application server. However, one Caché instance cannot act as an ECP data server for the data it receives as an application server of another ECP data server.

In an ECP configuration, each *ECP application server* is responsible for the following:

- Establishing connections to a specific ECP data server whenever an application requests data that is stored on that server.
- Tracking the status of all connections to ECP data servers. If a connection is broken, or encounters any trouble, the ECP application server attempts to recover the connection.
- Maintaining, in its cache, data retrieved across the network

The EcpAppServer test monitors the caching and data management functions performed by the ECP application server.

Purpose	Monitors the caching and data management functions performed by the ECP application server
Target of the test	A Cache Database server
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Cache database server 3. **PORT** – The port on which the server is listening 4. **SNMPPORT** - The port number through which the target server exposes its SNMP MIB. The default value is 161. 5. **SNMPVERSION** – By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the **SNMPVERSION** list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list. 6. **SNMPCOMMUNITY** – The SNMP community name that the test uses to communicate with the target server. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the **SNMPVERSION** chosen is **v3**, then this parameter will not appear. 7. **USERNAME** – This parameter appears only when **v3** is selected as the **SNMPVERSION**. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against the **USERNAME** parameter. 8. **AUTHPASS** – Specify the password that corresponds to the above-mentioned **USERNAME**. This parameter once again appears only if the **SNMPVERSION** selected is **v3**. 9. **CONFIRM PASSWORD** – Confirm the **AUTHPASS** by retyping it here. 10. **AUTHTYPE** – This parameter too appears only if **v3** is selected as the **SNMPVERSION**. From the **AUTHTYPE** list box, choose the authentication algorithm using which SNMP v3 converts the specified **USERNAME** and **PASSWORD** into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: ➢ **MD5** – Message Digest Algorithm ➢ **SHA** – Secure Hash Algorithm 11. **ENCRYPTFLAG** – This flag appears only when **v3** is selected as the **SNMPVERSION**. By default, the eG agent does not encrypt SNMP requests. Accordingly, the **ENCRYPTFLAG** is set to **NO** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **YES** option. 12. **ENCRYPTTYPE** – If the **ENCRYPTFLAG** is set to **YES**, then you will have to mention the encryption type by selecting an option from the **ENCRYPTTYPE** list. SNMP v3 supports the following encryption types: ➢ **DES** – Data Encryption Standard ➢ **AES** – Advanced Encryption Standard 13. **ENCRYPTPASSWORD** – Specify the encryption password here. 14. **CONFIRM PASSWORD** – Confirm the encryption password by retyping it here.

	15. **TIMEOUT** – Specify the duration (in seconds) beyond which the SNMP query issues by this test should time out. The default period is 10 seconds.		
Outputs of the test	One set of results for every instance of the Cache database server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Global references not from cache: Indicates the number of global references that were not served by the cache since the last measurement period.	Number	Ideally, this value should be low. A high value of this measure could indicate that many global references were served by directly accessing remote data, thus increasing the network overheads.
	Data sent by ECP Application server: Indicates the amount of data sent by the application server since the last measurement period.	KB	
	Data received by ECP Application server: Indicates the data received by the application server since the last measurement period.	KB	

8.1.7 Ecp Data Server Test

One of the most powerful and unique features of Caché is the ability to efficiently distribute data and application logic among a number of server systems. The underlying technology behind this feature is the Enterprise Cache Protocol (ECP): a distributed data caching architecture that manages the distribution of data and locks among a heterogeneous network of server systems. Unlike other "multi-tier" architectures, ECP is primarily a configuration option. That is, you do not have to use special code or development techniques to create distributed database applications.

Furthermore, the architecture and operation of ECP is conceptually simple. ECP provides a way to efficiently share data, locks, and executable code among multiple Caché systems. Data and code are stored remotely, but are cached locally to provide efficient access with minimal network traffic.

An ECP configuration consists of a number Caché systems that are visible to one another across a TCP/IP-based network. There are two roles a Caché system can play in an ECP configuration:

- **ECP Data Server** — a Caché system that is providing data for one or more ECP application server systems.
- **ECP Application Server** — a Caché system that is consuming data provided by one or more ECP data server systems.

A Caché system can simultaneously act as both an ECP data server and an ECP application server. However, one Caché instance cannot act as an ECP data server for the data it receives as an application server of another ECP data server.

In an ECP configuration, each *ECP data server* is responsible for the following:

- Storing data in its local database
- Maintaining the coherency of the various ECP application server system database caches so that application servers do not see stale data
- Managing the distribution of locks across the network

The EcpDataServer test monitors how well the data server manages data.

Purpose	Monitors how well the data server manages data
Target of the test	A Cache Database server
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Cache database server 3. **PORT** – The port on which the server is listening 4. **SNMPPORT** - The port number through which the target server exposes its SNMP MIB. The default value is 161. 5. **SNMPVERSION** – By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the **SNMPVERSION** list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list. 6. **SNMPCOMMUNITY** – The SNMP community name that the test uses to communicate with the target server. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the **SNMPVERSION** chosen is **v3**, then this parameter will not appear. 7. **USERNAME** – This parameter appears only when **v3** is selected as the **SNMPVERSION**. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against the **USERNAME** parameter. 8. **AUTHPASS** – Specify the password that corresponds to the above-mentioned **USERNAME**. This parameter once again appears only if the **SNMPVERSION** selected is **v3**. 9. **CONFIRM PASSWORD** – Confirm the **AUTHPASS** by retyping it here. 10. **AUTHTYPE** – This parameter too appears only if **v3** is selected as the **SNMPVERSION**. From the **AUTHTYPE** list box, choose the authentication algorithm using which SNMP v3 converts the specified **USERNAME** and **PASSWORD** into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: ➢ **MD5** – Message Digest Algorithm ➢ **SHA** – Secure Hash Algorithm 11. **ENCRYPTFLAG** – This flag appears only when **v3** is selected as the **SNMPVERSION**. By default, the eG agent does not encrypt SNMP requests. Accordingly, the **ENCRYPTFLAG** is set to **NO** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **YES** option. 12. **ENCRYPTTYPE** – If the **ENCRYPTFLAG** is set to **YES**, then you will have to mention the encryption type by selecting an option from the **ENCRYPTTYPE** list. SNMP v3 supports the following encryption types: ➢ **DES** – Data Encryption Standard ➢ **AES** – Advanced Encryption Standard 13. **ENCRYPTPASSWORD** – Specify the encryption password here. 14. **CONFIRM PASSWORD** – Confirm the encryption password by retyping it here.

	15. **TIMEOUT** – Specify the duration (in seconds) beyond which the SNMP query issues by this test should time out. The default period is 10 seconds.		
Outputs of the test	One set of results for every instance of the Cache database server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Global references returned by ECP data server: Indicates the number of global references that were served by the data stored in the ECP data server.	Number	Ideally, this value should be low. A high value of this measure could indicate that the ECP application server cache was ineffective in servicing many requests, and has hence redirected them to data server, thereby consuming a lot of network bandwidth.
	Requests received by ECP data server: Indicates the number of requests received by the data server since the last measurement period.	Number	
	Blocks sent by ECP data server: Indicates the number of blocks of data sent by the data server.	Number	
	Data sent from ECP data server: Indicates the data sent by the data server since the last measurement period.	KB	
	Data received by ECP data server: Indicates the data received by the data server since the last measurement period.	KB	

8.2 The Cache Memory Structures Layer

Using the tests mapped the **Cache Memory Structures** layer, administrators can assess how well the Cache database server managers its buffer pools and locks.

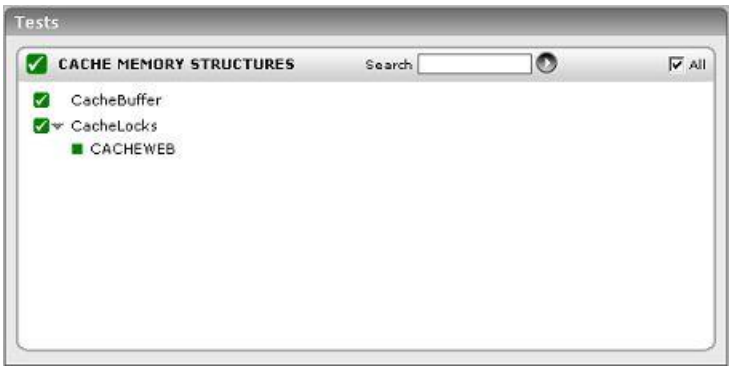


Figure 8.17: The tests associated with the Cache Memory Structures layer

8.2.1 Cache Buffer Test

Caché maintains a buffer pool — an in-memory cache of frequently referenced data blocks — to reduce the cost of fetching blocks from disk. By default, 16 KB is the memory allocated to the database cache. Close monitoring of the buffer pool is essential to determine the adequacy of the memory allocation to the cache. By increasing the memory allocation, administrators can considerably increase cache hits, reduce disk accesses, and consequently, save the processing overheads.

The CacheBuffer test monitors the usage of the buffer pool of every database instance of the Cache database server. In the process, the test also observes the behavior of the interactive and batch buffer queues. Any Cache process created in the Cache database is set to execute either in the 'interactive' or in the 'batch' mode. Since an 'interactive' process expects user inputs, it has to be run in the foreground and is also, resource-intensive. A 'batch' process on the other hand can only be run in the background, and uses less resources.

Purpose	Monitors the usage of the buffer pool of every database instance of the Cache database server
Target of the test	A Cache Database server
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Cache database server 3. **PORT** – The port on which the server is listening 4. **SNMPPORT** - The port number through which the target server exposes its SNMP MIB. The default value is 161. 5. **SNMPVERSION** – By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the **SNMPVERSION** list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list. 6. **SNMPCOMMUNITY** – The SNMP community name that the test uses to communicate with the target server. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the **SNMPVERSION** chosen is **v3**, then this parameter will not appear. 7. **USERNAME** – This parameter appears only when **v3** is selected as the **SNMPVERSION**. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against the **USERNAME** parameter. 8. **AUTHPASS** – Specify the password that corresponds to the above-mentioned **USERNAME**. This parameter once again appears only if the **SNMPVERSION** selected is **v3**. 9. **CONFIRM PASSWORD** – Confirm the **AUTHPASS** by retyping it here. 10. **AUTHTYPE** – This parameter too appears only if **v3** is selected as the **SNMPVERSION**. From the **AUTHTYPE** list box, choose the authentication algorithm using which SNMP v3 converts the specified **USERNAME** and **PASSWORD** into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: ➢ **MD5** – Message Digest Algorithm ➢ **SHA** – Secure Hash Algorithm 11. **ENCRYPTFLAG** – This flag appears only when **v3** is selected as the **SNMPVERSION**. By default, the eG agent does not encrypt SNMP requests. Accordingly, the **ENCRYPTFLAG** is set to **NO** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **YES** option. 12. **ENCRYPTTYPE** – If the **ENCRYPTFLAG** is set to **YES**, then you will have to mention the encryption type by selecting an option from the **ENCRYPTTYPE** list. SNMP v3 supports the following encryption types: ➢ **DES** – Data Encryption Standard ➢ **AES** – Advanced Encryption Standard 13. **ENCRYPTPASSWORD** – Specify the encryption password here. 14. **CONFIRM PASSWORD** – Confirm the encryption password by retyping it here.

	15. **TIMEOUT** – Specify the duration (in seconds) beyond which the SNMP query issues by this test should time out. The default period is 10 seconds.		
Outputs of the test	One set of results for the Cache database server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Buffer size Indicates the current size of the buffer	Bytes	A well-tuned environment is one where the buffer is sufficiently sized. A low value of this measure is unhealthy, as light weight buffers force a large number of direct disk accesses causing the database to incur excessive processing overheads.
	Buffer count: Indicates the number of buffers of buffer size	Number	
	Batch queue offset: Indicates the current offset to the start of the LRU (least recently used) queue.	Number	
	Interactive buffers: Indicates the number of buffers currently in the interactive portion of the LRU queue	Number	
	Max interactive buffers: Indicates the maximum number of buffers in the interactive portion of the LRU queue.	Number	
	Requeue limit interactive: Indicates the threshold for requeuing an interactive buffer.	Number	
	Requeue limit batch: Indicates the threshold for requeueing a batch buffer.	Number	
	Interactive write queue: Indicates the number of interactive buffers in the current write cycle.	Number	

	Buffers write queue: Indicates the number of buffers requeued in the batch LRU queue.	Number	
	Write queue wait max: Indicates the threshold for waking the interactive write daemon.	Number	
	Available interactive buffers: Indicates the current number of available interactive buffers.	Number	Ideally, this value should be high. If the value is low, then the buffer might be unable to serve all interactive job requests, thus forcing direct disk accesses. This could ultimately increase processing overheads.
	Minimum interactive buffers: Indicates the minimum number of buffers in the interactive portion of the LRU queue.	Number	
	Buffer usage: Indicates the percentage of buffer memory utilized.	Percent	A value close to 100% is a cause of concern, as it indicates that the buffer pool is rapidly filling up. This implies that there may not be enough space for additional buffers in the buffer pool; this may force direct disk accesses, which are I/O-intensive. For optimal performance therefore, the value of this measure should be low.

8.2.2 Cache Locks Test

Caché locks are created when a Caché process issues a Lock command on a Caché entity, such as a local or global variable, as long as the entity is not already locked by another process. Periodic monitoring of the locking activity on every Cache instance is imperative to ensure that no application-critical Cache entity is locked, as this can sometimes cause serious application errors. Similarly, the number of locks used by a Cache system can also impact Cache performance. Besides the number, it would be good practice to keep an eye out for the lock type too, so that potentially dangerous lock types are detected and released in time. The CacheLocks test that the eG agent executes on a Cache server serves all the above-mentioned purposes. This test reports the number of locks on every Cache instance, and additionally reveals the lock type, reference, and owner of the lock, so as to aid further diagnosis.

Purpose	Reports the number of locks on every Cache instance
Target of the test	A Cache Database server
Agent deploying the test	An internal/remote agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Cache database server
3. PORT – The port on which the server is listening
4. INSTANCEDIRECTORY - This test uses the cstat utility to collect the required metrics. This utility is typically run from the install directory of a monitored Cache instance. Therefore, in the INSTANCEDIRECTORY text box, specify the name of the instance being monitored and the install directory of that instance, in the following format: InstanceName:InstallDirectory. To ensure that the test reports metrics for multiple Cache instances, provide a comma-separated list of InstanceName:InstallDirectory pairs in the INSTANCEDIRECTORY text box. For example: CACHEWEB:d:\Intersystems\CacheWeb,CACHE2:d:\Intersystems\Cache2.
5. DETAILED DIAGNOSIS - To make diagnosis more efficient and accurate, the eG Enterprise system 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 against DETAILED DIAGNOSIS. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every InstanceName configured</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Lock entries:
Indicates the current number of locks on this Cache instance.</td><td>Number</td><td>If the value of this measure is very high or is increasing consistently, then you might have to enlarge the size of the lock table. Also, you might want to view the detailed diagnosis of this measure, so that you can determine the process that is holding each of the locks, the lock reference, and most importantly, the lock type.
An 'incremental lock', for instance, is potentially dangerous because it can lead to a situation known as "deadlock". This situation occurs when two processes each assert an incremental lock on a variable already locked by the other process. Because the attempted locks are incremental, the existing locks are not released. As a result, each process hangs while waiting for the other process to release the existing lock.</td></tr>
</table>

The lock type/mode can be any one of the following:

Type of Lock	Description

Exclusive	Exclusive lock mode
Shared	Share lock mode
LockZA	ZALLOCATE lock mode
WaitLock	Waiting for exclusive lock mode
WaitShare	Waiting for share lock mode
WaitLockZA	Waiting for ZALLOCATE lock mode
LockPending	Exclusive lock pending, waiting for server to grant the exclusive lock
SharePending	Share lock pending, waiting for server to grant the share lock
DelockPending	Delock pending, waiting for server to release the lock
Lost	Lock lost due to network reset

8.3 The Cache Service Layer

The **Cache Service** layer evaluates how well the Cache server performs critical operations such as global referencing, journaling, locking and unlocking, resource seizing, etc. Besides, external tests are also included in this layer to determine the availability and responsiveness of the Cache.

Figure 8.18: The tests associated with the Cache Service layer

8.3.1 Cache Performance Test

This test monitors the critical determinants of the performance of a Cache database server. This includes:

- Code processing speed
- Routine loads and saves

- Global jobs
- Logical block accesses
- Journal entries
- Locking activity

Purpose	Monitors the critical determinants of the performance of a Cache database server
Target of the test	A Cache Database server
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Cache database server 3. **PORT** – The port on which the server is listening 4. **SNMPPORT** - The port number through which the target server exposes its SNMP MIB. The default value is 161. 5. **SNMPVERSION** – By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the **SNMPVERSION** list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list. 6. **SNMPCOMMUNITY** – The SNMP community name that the test uses to communicate with the target server. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the **SNMPVERSION** chosen is **v3**, then this parameter will not appear. 7. **USERNAME** – This parameter appears only when **v3** is selected as the **SNMPVERSION**. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against the **USERNAME** parameter. 8. **AUTHPASS** – Specify the password that corresponds to the above-mentioned **USERNAME**. This parameter once again appears only if the **SNMPVERSION** selected is **v3**. 9. **CONFIRM PASSWORD** – Confirm the **AUTHPASS** by retyping it here. 10. **AUTHTYPE** – This parameter too appears only if **v3** is selected as the **SNMPVERSION**. From the **AUTHTYPE** list box, choose the authentication algorithm using which SNMP v3 converts the specified **USERNAME** and **PASSWORD** into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm 11. **ENCRYPTFLAG** – This flag appears only when **v3** is selected as the **SNMPVERSION**. By default, the eG agent does not encrypt SNMP requests. Accordingly, the **ENCRYPTFLAG** is set to **NO** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **YES** option. 12. **ENCRYPTTYPE** – If the **ENCRYPTFLAG** is set to **YES**, then you will have to mention the encryption type by selecting an option from the **ENCRYPTTYPE** list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard 13. **ENCRYPTPASSWORD** – Specify the encryption password here. 14. **CONFIRM PASSWORD** – Confirm the encryption password by retyping it here.

	15. **TIMEOUT** – Specify the duration (in seconds) beyond which the SNMP query issues by this test should time out. The default period is 10 seconds. 16. **INSTANCENAME** – By default, this is set to *All*, indicating that all Cache instances will be monitored. To monitor specific instances, provide a comma-separated list of Cache names.		
Outputs of the test	One set of results for every *InstanceName* configured		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Lines executed: Indicates the number of lines of code per second that were executed on the server.	Lines/Sec	This serves as a good measure of the processing capability of the database server.
	Routine loads/saves: Indicates the rate at which routine loads and saves occurred.	Routines/Sec	*Routines* are small programs that run within a Cache database server. In a well-tuned environment, this number increases slowly, since most routine loads are satisfied by the routine cache memory without accessing the disk. Each routine load or save transfers up to 32 KB of data.
	New global references: Indicates the rate of at which new global references were set in this database instance.	Refs/Sec	A *global* is a named multidimensional array that is used for storing data in a physical Caché database. Data can be spread across many globals that exist in different databases on the same system or remote systems. Using a *namespace*, you can group closely related globals and databases, so that data can be easily referenced. Cross-referencing of data ensures better data storage and efficient query execution.
	New global sets: Indicates the rate at which new globals were created in this database instance.	Sets/Sec	This measure considers the rate of **SET** operations performed on this database instance since the last measurement period.
	New global kills: Indicates the rate at which new globals were killed in this database instance.	Kills/Sec	This measure considers the number of **KILL** operations per second performed on this database instance since the last measurement period.
	Logical database block reads: Indicates the rate at which data blocks were read from the disk.	Blocks/Sec	Globals are stored on disk within a series of *data blocks*; the size of each block (typically 8KB) is determined when the physical database is created. A high value for this measure indicates that direct disk accesses are high. In such a case your database might require some fine-tuning. Consider resizing your buffer pool to increase buffer accesses and reduce data retrievals from the disk.

	Physical database block reads: Indicates the rate at which physical database blocks (2-KB or 8-KB) were read from disk for both global and routine references.	Blocks/Sec	A high value for this measure indicates that direct disk accesses are high. In such a case your database might require some fine-tuning. Consider resizing your buffer pool to increase buffer accesses and reduce data retrievals from the disk.
	Physical database block writes: Indicates the rate at which physical database blocks (2-KB or 8-KB) were written to disk for both global and routine references.	Blocks/Sec	
	New database journal entries: Indicates the number of entries recorded in the journal during this measurement period.	Number	To provide database integrity and reliability, Caché includes a number of *journaling* subsystems that keep track of physical and logical database updates. The journal management technology is also used to provide transaction support (a journal is used to perform transaction rollback operations) as well as database shadowing (a journal is used to synchronize a shadow server with a primary data server). This measure hence aids in effective audit tracking of your database, as it can tell you whether any modifications have been performed on the database - be it a set/kill/transaction event. Since the events for which journaling is needed is configurable, you might want to track critical changes to your database by enabling journaling.
	Lock commands: Indicates the number of **LOCK** commands that are issued in the last measurement period.	Number	
	Lock successes: Indicates the number of **LOCK** commands that succeeded in the last measurement period.	Number	When the attempt to lock a transaction succeeds, it means that other processes will be prevented from seeing the modifications to the transaction, until the transaction is committed.

	Lock failures: Indicates the number of **LOCK** commands that have failed in the last measurement period.	Number	A lock command typically fails when it is unable to acquire a lock on a transaction - this happens when the transaction to be locked is already locked by another process.
	Current global jobs: Indicates the number of jobs currently counted 'in global'.	Number	
	Max global jobs: Indicates the maximum number of jobs currently counted 'in global'.	Number	
	Throttle wait jobs: Indicates the number of jobs required to wait for a throttle.	Number	

	Write update status: Indicates whether /not the Write Daemon has successfully updated the database.	Boolean	Rather than writing directly from memory to the database, the Caché write daemon system process (**WRTDMN**) uses an intermediate file, the write image journal. This file is usually named **cache.wij**. Write image journaling technology uses a two-phase process of writing to the database, as follows: ▪ In the first phase, Caché records the changes needed to complete the update in the write image journal. Once all updates to the write image journal have been entered, a flag is set in the file and the second phase begins. ▪ In the second phase, the Write daemon writes the changes recorded in the write image journal to the database on disk. When this second phase completes, the Write daemon sets a flag in the write image journal to indicate it is empty. When Caché starts, it automatically checks the write image journal and runs a recovery procedure if it detects that an abnormal shutdown occurred. When Caché indicates successful completion, the internal integrity of the database is restored. Recovery is necessary if a system crash or other major system malfunction occurs at either of the following points in the two-phase write protocol process: ▪ Before the Write daemon has completed writing the update to the write image journal. In this case, recovery discards the incomplete entry. ▪ After the update to the write image journal is complete but before the update(s) to the database is complete. In this case, recovery rewrites the updates from the write image journal file to the database. **Note:** The two-phase write protocol safeguards structural database integrity, but it does not prevent data loss. If the system failure occurs prior to a complete write of an update to the write image journal, Caché does not have all the information it needs to perform a complete update to disk. That data is lost.

			If the value of this measure is '1', it indicates that the daemon has written to the database successfully. The value '0', on the other hand, indicates an unsuccessful attempt to perform global updates.
	Users updating: Indicates the number of users performing global updates.	Number	
	Write daemon status: Indicates the status of the write daemon.	Boolean	The value 1 indicates that the write demon is running, and the value 0 indicates that write demon is waiting for wakeup. If this value becomes 2, it indicates that Cache stopped the write daemon to prevent further harm.

8.3.2 Cache Resources Test

A number of areas of the core Caché code require protection from simultaneous access by multiple processes. These areas are dubbed resources. Sometimes, when a resource requested by a Cache process is already in use, then a **spin lock** is created on the resource - in this case, the process simply waits in a loop (i.e., spins) and repeatedly checks the lock status of the resource, until a lock on the resource is available to it. A few other times, the process, instead of creating a spin lock, might wait for a while to see if the resource is released, and if not, switch to the sleep mode. This process of acquiring a lock on a resource (with or without spinning) is also known as seizing. The CacheResource test auto-discovers the resources on every configured Cache instance, and reports the number of times seizes have occurred on the discovered resources, thereby enabling administrators to determine whether there is a serious contention for resources on the Cache instance. Moreover, the detailed diagnosis of this test additionally reveals the number of seizes per seize state, thus isolating Cache instances where highly expensive seize operations are being performed.

Purpose	Auto-discovers the resources on every configured Cache instance, and reports the number of times seizes have occurred on the discovered resources
Target of the test	A Cache Database server
Agent deploying the test	An internal/remote agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the Cache database server
3. PORT – The port on which the server is listening
4. INSTANCEDIRECTORY - This test uses the cstat utility to collect the required metrics. This utility is typically run from the install directory of a monitored Cache instance. Therefore, in the INSTANCEDIRECTORY text box, specify the name of the instance being monitored and the install directory of that instance, in the following format: InstanceName:InstallDirectory. To ensure that the test reports metrics for multiple Cache instances, provide a comma-separated list of InstanceName:InstallDirectory pairs in the INSTANCEDIRECTORY text box. For example: CACHEWEB:d:|Intersystems|CacheWeb,CACHE2:d:|Intersystems|Cache2.
5. DETAILED DIAGNOSIS - To make diagnosis more efficient and accurate, the eG Enterprise system 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 against DETAILED DIAGNOSIS. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every InstanceName:Resource pair discovered</td></tr>
<tr><td></td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

<table>
<tr>
<td>Measurements made by the test</td>
<td>Resource seizures:
Indicates the number of times this resource was seized.</td>
<td>Number</td>
<td>Ideally, this value should be low. If the value is high, then, you can use the detailed diagnosis of this measure (if enabled), to know the seize state, and the number of seizes in every state, so that you can analyze the implications of the high seize count effectively. The various seize states are as follows:
<table>
<tr><th>Seize State</th><th>Description</th></tr>
<tr><td>A Seize</td><td>Number of times spun before acquiring lock on the resource</td></tr>
<tr><td>N Seize</td><td>Number of times spun before failing to acquire a lock on the resource</td></tr>
<tr><td>B Seize</td><td>Number of times spun and switched to sleep mode before acquiring a lock on the resource</td></tr>
<tr><td>BusySet</td><td>Other waiters spinning should wait behind this one</td></tr>
</table>
N Seizes are typically expensive on SMP systems.</td>
</tr>
</table>

8.3.3 Cache Network Test

This test connects to the Cache database server, executes a query, and determines the availability and responsiveness of the Cache database server.

This test requires JDK 1.5 for execution. However, only JDK 1.3 is bundled with the eG agent. Therefore, to ensure the smooth execution of the test, you will have to do the following before attempting to monitor the Cache database server:

- Install JDK 1.5 on the host of the external agent that will be executing this test.
- Edit the **debugon.bat** and **debugoff.bat** files in the **<EG_AGENT_INSTALL_DIR>\lib** directory.
- These files contain path specifications to the default JRE 1.3.1 (which is in the **<EG_AGENT_INSTALL_DIR>\JRE** directory). Replace these path specifications with that of JRE 1.5 that you just installed on the host. In the extract below, the text in **Bold** indicates where and what changes have to be made to these files:

```
set
path="<JDK1.5_install_dir>\bin";C:\eGurkha\bin;C:\eGurkha\lib;C:\eGurkha\bin
\ic;D:\oracle\ora92\bin;C:\Program Files\Oracle\jre\1.3.1\bin;C:\Program
Files\Oracle\jre\1.1.8\bin;%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\S
ystem32\Wbem;C:\Program Files\Common Files\InterSystems\Cache;C:\Program
Files\Microsoft SQL Server\80\Tools\BINN;C:\Program Files\Microsoft SQL
Server\80\Tools\Binn\;%EGURKHA_PATH%

net stop eGurkhaAgent
```

```
js -uninstall eGurkhaAgent

js -install eGurkhaAgent "<JDK1.5_install_dir>\jre\bin\client\jvm.dll" -Xrs
-Djava.class.path=%classpath% -
Djava.library.path=C:\eGurkha\lib;C:\eGurkha\bin -start EgMainAgent -params
-manager 192.168.10.12 -port 7077 -dir C:\eGurkha -ssl false -highSecurity
false -path "<JDK1.5_install_dir>\jre\bin"

exit
```

- Save the changes.
- Run the **debugoff.bat** file to reinstall the agent service to use the new JRE.
- Finally, restart the eG agent that will be executing this test.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Cache Database* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the availability and responsiveness of the Cache database server		
Target of the test	A Cache Database server		
Agent deploying the test	An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the target Intersystems Cache database server is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the target server is listening.		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the Cache database server 3. **PORT** – The port on which the server is listening 4. **DB** - The database to which the test should connect. The default value is **%SYS**. 5. **USER** - The user name to be used by the test for connecting to the specified **DB**. The default value is **_SYSTEM**. 6. **PASSWORD** - Specify the password of the **USER**. 7. **CONFIRM PASSWORD** - Confirm the password by retyping it in this space. 8. **QUERY** - The query to be executed on the **DB** to ascertain the availability and responsiveness of the server		
Outputs of the test	One set of results for the Cache database server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Availability: Whether the database server is available or not	Number	While the value 0 indicates that the server is not running, 100 indicates that the server is alive and well. Availability problems may be caused by a misconfiguration/malfunctioning of the database instance, or because the instance is using an invalid user account.

	Response time: Indicates the time taken by the server to respond to requests	Secs	A sudden increase in response time is indicative of a bottleneck at the database server.

9

Externally Monitoring Oracle servers

Previously, we elaborately discussed about the wide variety of internal metrics that the eG agent collects from the Oracle server, and how these metrics impact the internal health of the server. However, some administrators might not have access to Oracle servers in their environment, and might hence be unable to install agents on them. These administrators might still want to monitor external health indicators such as the availability and responsiveness of the Oracle server. In order to enable administrators to collect such external metrics in a non-intrusive manner, eG Enterprise offers the *External Oracle* model (see Figure 9.1). To use this model, only a single eG external agent is required; this agent sits on a remote host and determines the health of the target Oracle server from an external perspective.

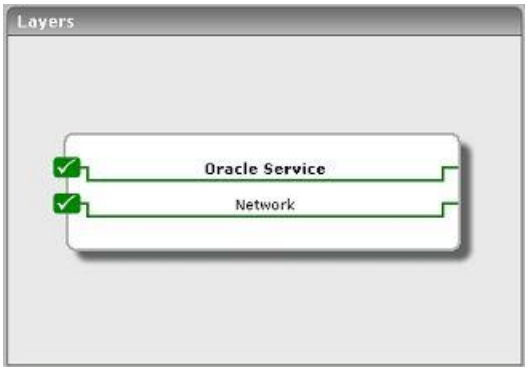


Figure 9.1: Layer model of the External Oracle server

The **Network** test associated with the **Network** layer runs network-level pings to check whether the Oracle server can be accessed over the network. Besides, the test also looks out for abnormal/very high packet loss, undue network delays, etc. To know more about the **Network** test, refer to the *Monitoring Unix and Windows Servers* document. The **Oracle Service** layer is mapped to an **OraSqlNet** test, which emulates a query to the Oracle database server from an external location, to determine the availability of the Oracle server and the speed with which it executes the query. For more details on this test, refer to Chapter 2 of this document.

10

Monitoring the Oracle RAC

Oracle Real Application Clusters (RAC) allows Oracle Database to run any packaged or custom application, unchanged across a server pool. This provides the highest levels of availability and the most flexible scalability. If a server in the pool fails, the Oracle database continues to run on the remaining servers. When you need more processing power, simply add another server to the pool without taking users offline. With Oracle Real Application Clusters, Oracle de-couples the Oracle Instance (the processes and memory structures running on a server to allow access to the data) from the Oracle database (the physical structures residing on the storage storing the data, commonly referred to as the datafiles).

An Oracle RAC database is a clustered database. A cluster can be described as a pool of independent servers that co-operate as a single system. A clustered database is a single database that can be accessed by multiple instances, where each instance runs on a separate server in the server pool. A server pool is made up of 1 or more servers, each having a public LAN connection, an interconnect connection, and must be connected to a shared pool of storage. Server Pools provide improved fault resilience and modular incremental system growth over single symmetric multi-processor (SMP) systems. In the event of a system failure, clustering ensures high availability to users. Also, when additional resources are required, additional servers and instances can easily be added to the server pool with no downtime. Each server in a server pool is called a cluster node.

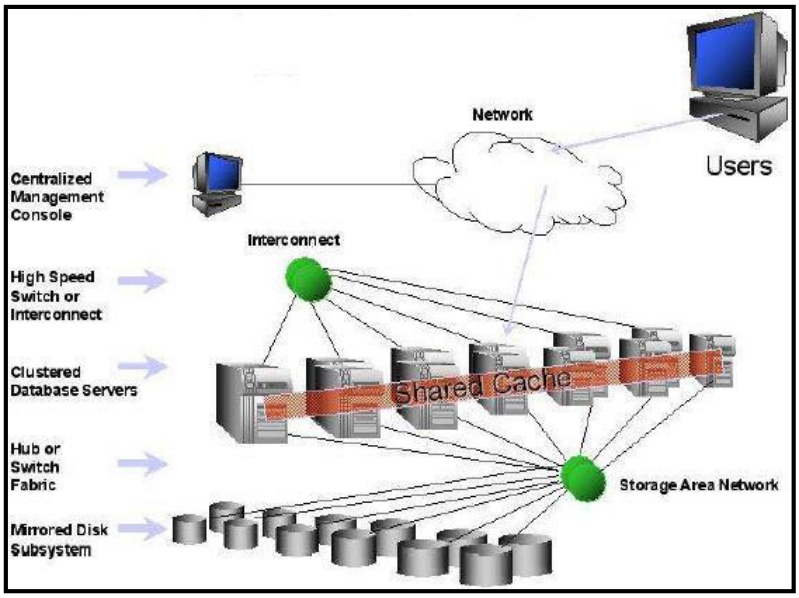


Figure 10.1: The Oracle RAC architecture

Besides cluster nodes, an Oracle RAC database requires Oracle clusterware and shared pool storage. Oracle Clusterware provides a complete clustering solution and supports any application. It is a prerequisite for all Oracle RAC implementations, and monitors and manages Oracle Real Application Cluster databases. When a server in the server pool is started, all instances, listeners and services are automatically started. If an instance fails, the Oracle Clusterware will automatically restart the instance so that the service is often restored before the administrator notices it was down.

Oracle Real Application Clusters is a shared everything architecture. All nodes in the cluster share all storage used for an Oracle RAC database. The type of storage pool used can be network attached storage (NAS), storage area network (SAN), or SCSI disks. Your storage choice is dictated by the server hardware choice and the hardware supported by your hardware vendor. The key to choosing an appropriate storage pool is choosing a storage system that will provide

scaleable I/O for your application and an I/O system that will scale as additional servers are added to the pool.

To connect to an Oracle RAC database, applications use a virtual IP address. This IP address is assigned to each server in the cluster so that, if a node fails, the virtual IP is failed over to another node in the cluster to provide an immediate "node down"-response to incoming connection requests. This increases the availability for applications.

Owing to the high availability and load-balancing capabilities, clustered databases are often the preferred backend in many mission-critical IT infrastructures delivering key end-user services. This is why, the non-availability of a clustered database to an application, or a tablespace contention experienced by the clustered database, or an unusually high locking activity on the database, may cause the dependent end-user services to suffer. To avoid this, the Oracle RAC database should be continuously monitored.

eG Enterprise provides a dedicated *Oracle Cluster* model for monitoring the Oracle RAC. This model requires that the eG agent be installed on any node of the cluster (in the case of an agent-based approach) or on any remote Windows host in the environment (in the case of an agentless approach). This agent should then be configured to communicate with the cluster via the virtual IP address of one of the cluster nodes (in the case of the agent-based approach, this will be the node on which the agent has been deployed) to report the availability of the cluster service, how load is balanced across the nodes of the service, the wait events that occur on each instance of the service, tablespace usage, lock behavior, and more!

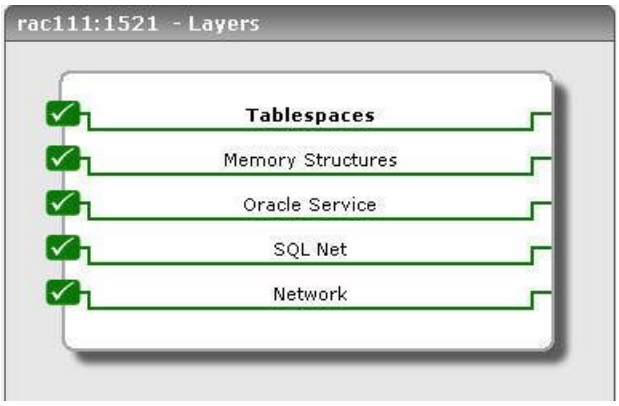


Figure 10.2: The layer model of the Oracle Cluster service

Each layer of Figure 10.2 above is mapped to tests that report a wealth of metrics that will help cluster administrators find quick and accurate answers for the following performance queries:

- Is the cluster service available? If so, how quickly does it respond to user requests?
- Is session load uniformly balanced across all the instances of the Oracle RAC?
- Are too many wait events occurring on any instance? What type of wait events are these - checkpoint events, log file syn waits, log file parallel write events, Db file parallel write events, or Db file sequential read events?
- Is any session/session module on an instance frequently waiting for CPU, cluster resources, I/O, locks, or latches? If so, which module/session is this and which instance is it associated with?
- Has the eG agent captured on a session/session module any wait event that should typically not occur on any system?
- What is the target and estimated MTTR (Mean Time to Recovery) of each of the instances managed by the RAC?

- Is the estimate of recovery I/O and the redo blocks that must be processed by an instance during recovery too high? Will it affect MTTR?
- Are the redo log files adequately sized to avoid unnecessary checkpointing?
- Is the checkpoint auto-tuning mechanism functioning effectively?
- Is any instance unncessarily holding many transaction locks for long periods of time?
- Is any tablespace experiencing a space drain? What type of objects (tables, indexes, partitions, LOB segments, etc.) are consuming too much space in this tablespace?
- Are temporary tablespaces adequately sized?
- Is the undo tablespace of any instance taking too long to execute queries? Is it because one/more queries are inefficient? Which queries are these?
- How long does an instance take to perform undo retention? Were any bottlenecks detected in this process?

The sections that follow will discuss each of the top 3 layers of Figure 10.2 elaborately, as the remaining layers have already been dealt with in the *Monitoring Oracle Database Servers* chapter in this document.

10.1 The Oracle Service Layer

Besides monitoring the session load on the cluster service and pinpointing issues in load-balancing across cluster instances, this layer monitors the following:

- The availability and responsiveness of the cluster service;
- The number and nature of wait events that occur on each instance of the cluster, and the sessions/session modules affected by these waits;
- The target and estimated MTTR (Mean Time to Recovery) for every instance;
- Accesses to the undo tablespace of an instance.

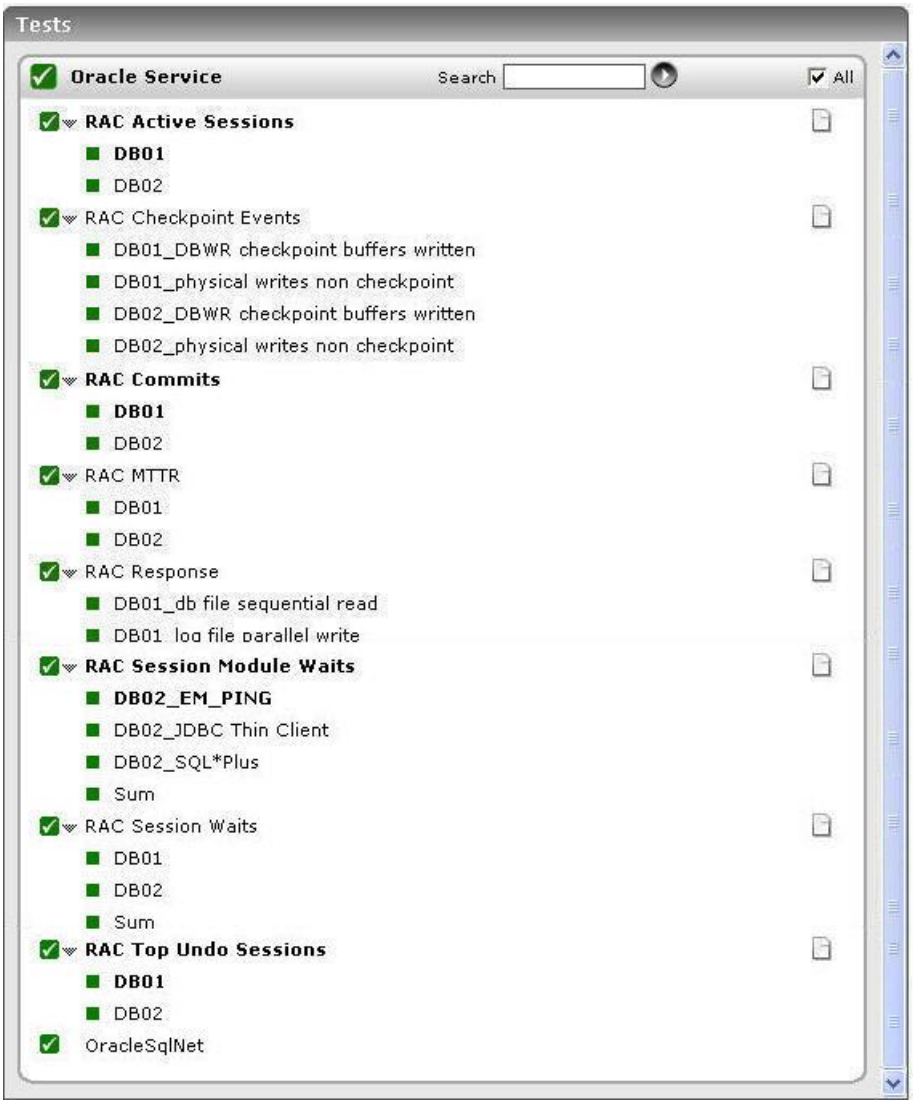


Figure 10.3: The Oracle Service Layer

10.1.1 Oracle RAC Active Sessions Test

This test reports the number of sessions currently active on each database in the Oracle RAC, and thus points you to overloaded instances.

Purpose	Reports the number of sessions currently active on each instance in the Oracle RAC, and thus points you to overloaded instances
Target of the test	Oracle RAC
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">9. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every instance monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Number of sessions:
Indicates the number of sessions currently active on this instance.</td><td>Number</td><td>This measure is a good indicator of the session load on an instance. Comparing the value of this measure across instances will help you identify the overloaded instance and also shed light on load-balancing irregularities.
You can use the detailed diagnosis of this measure to view the details of the active sessions. With the help of these details, you can figure out whether a session has remained open for too long a time due to prolonged wait events.</td></tr>
</table>

10.1.2 Oracle RAC Checkpoint Events Test

The checkpoint process is responsible for updating file headers in the database datafiles. A checkpoint occurs when Oracle moves new or updated blocks (called dirty blocks) from the RAM buffer cache to the database datafiles. A checkpoint keeps the database buffer cache and the database datafiles synchronized. This synchronization is part of the mechanism that Oracle uses to ensure that your database can always be recovered.

Check-pointing is an important Oracle activity which records the highest system change number (SCN), so that all data blocks less than or equal to the SCN are known to be written out to the data files. If there is a failure and then subsequent cache recovery, only the redo records containing changes at SCN(s) higher than the checkpoint need to be applied during recovery.

Key checkpoint-related activities may generate wait events. For instance, SQL statements may have to wait for processing until the DBWR (database writer) finishes writing dirty blocks in the buffer cache to the datafiles. If too many such wait events occur on an instance, it may cause the performance of the Oracle cluster to deteriorate. It is hence essential to keep close tabs on the checkpoint-related wait events and the activity responsible for them.

The **RAC Checkpoint Events** test auto-discovers the wait event types related to the checkpoint process, and reports

the number of events of each type that have occured in each instance of an Oracle RAC.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Auto-discovers the wait event types related to the checkpoint process, and reports the number of events of each type that have occured in each instance of an Oracle RAC
Target of the test	Oracle RAC
Agent deploying the test	An internal/remote agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Outputs of the test	One set of results for every wait event type discovered on each instance of the monitored RAC		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Event Count: Indicates the number of wait events of this type that have occurred on this instance during the last measurement period.	Number	Ideally, the value of this measure should be low. A consistent increase in this value is a cause of concern, as it indicates that a checkpoint-related activity is not getting completed, resulting in the generation of numerous wait events and degrading the overall performance of the Oracle RAC. Compare the value of this measure across the event types to determine which type of wait event has occurred most frequently on an instance.

10.1.3 Oracle RAC Commits Test

A wait class is a grouping of wait events, and every wait event belongs to a wait class. The main wait classes of the Oracle database server are:

- Administrative
- Application
- Cluster
- Commit
- Concurrency
- idle
- Network
- Other
- Scheduler
- System I/O
- User I/O

The Commit wait class comprises of only one wait event - wait for redo log write confirmation after a commit (that is, 'log file sync' event). Commit is not complete until LGWR (log writer) writes log buffers including commit redo records to log files. In a nutshell, after posting LGWR to write, user or background processes waits for LGWR to signal back with 1 sec timeout. User process charges this wait time as 'log file sync' event.

This test reports the number of sessions to each instance, which are waiting for a redo log write confirmation after a commit. This way, the test sheds light on the open sessions to a instance, and the reason for the sessions remaining This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED**

TESTS list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the number of sessions to each instance that are waiting for a redo log write confirmation after a commit
Target of the test	Oracle RAC
Agent deploying the test	An internal/remote agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 10. **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.		
Outputs of the test	One set of results for every instance in the monitored Oracle RAC		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Number of Sessions:** Indicates the number of sessions to this instance that are waiting for redo log confirmation after commit.	Number	A steady increase in the value of this measure is a cause of concern, as it indicates the following: ss. Many sessions are forced to stay open owing to the commit wait events, and this may cause a session overload; tt. Many "log file sync" wait events are occuring, causing the performance of the Oracle RAC to deteriorate. The root cause for 'log file sync' waits are as follows: o LGWR is unable to complete writes fast enough - this could be because, the disk I/O performance to log files is not good enough or, the LGWR is starving for CPU resources or, the LGWR paged out due to memory starvation issues or, due to file system or unix buffer cache limitations o LGWR is unable to post the processes fast enough, due to excessive commits. o IMU undo/redo threads o LGWR is suffering from other database contention such as enqueue waits or latch contention o Various bugs Use the detailed diagnosis of this measure to view the details of the sessions affected by log file sync waits.

10.1.4 Oracle RAC MTTR Test

Instance recovery, which is the process of recovering the redo thread from the failed instance, is a critical component affecting availability. When using Oracle RAC, the SMON process in one surviving instance performs instance recovery of the failed instance. The sooner this happens and lesser the I/O that is consumed during recovery, the better will be the user experience with the Oracle RAC.

Mean time to recovery (MTTR) is the average time that the Oracle server will take to recover from any failure. In order to limit recovery I/O and optimize cluster performance, you need to understand the MTTR target your system is currently achieving and what your potential MTTR target could be, given the I/O capacity. This test provides you with this understanding by reporting the target and estimated MTTR, and by monitoring the key factors affecting MTTR such as the redo log size and the number of redo blocks to be processed.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS**

page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the target and estimated MTTR, and monitors the key factors affecting MTTR such as the redo log size and the number of redo blocks to be processed.
Target of the test	Oracle RAC
Agent deploying the test	An internal/remote agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running.

	Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every instance in the monitored Oracle RAC		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Target MTTR: Indicates the effective mean time to recover (MTTR) this instance.	Secs	Usually, the value of this measure should be equal to the value of the **FAST_START_MTTR_TARGET** initialization parameter. **FAST_START_MTTR_TARGET** specifies a target for the expected mean time to recover (MTTR), that is, the time (in seconds) that it should take to start up the instance and perform cache recovery. After **FAST_START_MTTR_TARGET** is set, the database manages incremental checkpoint writes in an attempt to meet that target. If **FAST_START_MTTR_TARGET** is set to such a small value that it is impossible to do a recovery within its time frame, then the the value of this measure will be larger than **FAST_START_MTTR_TARGET**. If **FAST_START_MTTR_TARGET** is set to such a high value that even in the worst-case (the whole buffer cache is dirty) recovery would not take that long, then the value of this measure will be the same as the *Estimated MTTR*. If **FAST_START_MTTR_TARGET** is not specified, then again, the value of this measure will be the same as the value of the *Estimated MTTR* measure.
	Estimated MTTR: Indicates the current estimated mean time to recover (MTTR).	Secs	This measure is calculated based on the number of dirty buffers and log blocks (0 if **FAST_START_MTTR_TARGET** is not specified). Basically, this value tells you how long you could expect recovery of the instance to take place based on the work your system is doing at the time of testing. This measure reports the estimated mean time to recovery based on the current state of the running database. If the database has just opened, the system may contain only a few dirty buffers, so not much cache recovery would be required if the instance failed at this moment. That is why the value of this measure can, for the moment, be lower than the minimum possible *Target MTTR*.

	Recovery Estimated IOs: Indicates the estimated number of dirty buffers in the buffer cache of this instance.	Number	
	Target Redo blocks: Indicates the target number of redo blocks that must be processed while recovering this instance.	Number	Instance recovery is nothing more than using the contents of the online log files to rebuild the database buffer cache to the state it was in before the crash. This will replay all changes extracted from the redo logs that refer to blocks that had not been written to disk at the time of the crash. Though instance recovery guarantees no corruption, it may take a considerable time to do its roll forward before the database can be opened. This time is dependent on two factors: how much redo has to be read and how many read/write operations will be needed on the datafiles as the redo is applied. The values of these measures serve as good indicators of the amount of redo reading work that needs to be performed as part of the recovery process, and are hence useful while determining the MTTR.
	Actual Redo blocks: Indicates the actual number of redo blocks that are required by this Oracle instance to recover.	Number	
	Writes logfile size: Indicates the number of writes driven by the smallest redo log file size for each oracle instance.	Number	This measure is used to drive the checkpoint process, if your redo log file size is under sized. Since the **FAST_START_MTTR_TARGET** parameter is set to limit the instance recovery time, Oracle automatically tries to checkpoint as frequently as necessary. Under such a condition, the size of the log files should be large enough to avoid additional checkpoint due to under sized log files.

	Writes auto tune: Indicates the number of writes due to auto-tune checkpointing.	Number	The checkpoint auto-tuning mechanism inspects statistics on machine utilization, such as the rate of disk I/O and CPU usage, and if it appears that there is spare capacity, it will use this capacity to write out additional dirty buffers from the database buffer cache, thus pushing the checkpoint position forward. The result is that even if the **FAST_START_MTTR_TARGET** parameter is set to a high value (the highest possible is 3600 seconds—anything above that will be rounded down), actual recovery time may well be much less. Enabling checkpoint auto-tuning with a high target should result in your instance always having the fastest possible recovery time that is consistent with maximum performance.

10.1.5 Oracle RAC Waits Response Test

This test reports the key performance statistics pertaining to the following wait events in each Oracle instance:

- log file parallel write: This event occurs when writing redo records to the redo log files from the log buffer. Writing redo records to the redo log files from the log buffer.
- Db file parallel write: This event occurs in the DBWR. It indicates that the DBWR is performing a parallel write to files and blocks. When the last I/O has gone to disk, the wait ends.
- log file sync: When a user session commits, the session's redo information needs to be flushed to the redo logfile. The user session will post the LGWR to write the log buffer to the redo log file. When the LGWR has finished writing, it will post the user session.
- Db file sequential read: The session waits while a sequential read from the database is performed. This event is also used for rebuilding the control file, dumping datafile headers, and getting the database file headers.

Effective wait analysis helps determine on which wait event the instance spends most of its time, and which current connections are responsible for the above-mentioned wait events.

Purpose	Reports key performance statistics pertaining to the following wait events in each Oracle instance: log file parallel write, Db file parallel write, log file sync, and Db file sequential read
Target of the test	Oracle RAC
Agent deploying the test	An internal/remote agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">9. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every wait event type captured on every instance in the monitored Oracle RAC</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total waits:
Indicates the total number of times this wait event has occurred since the last measurement period.</td><td>Number</td><td>If the value of this measure is very high, then you can drill down further using the detailed diagnosis capability (if enabled) of the eG Enterprise suite to figure out which current connections may be responsible for this. The detailed diagnosis of this measure reveals the session IDs of the sessions that caused the wait events to occur, the users who initiated the sessions, and the total number of waits, wait time, and the maximum wait time for every session.</td></tr>
<tr><td>Time waited:
Indicates the total time for which the events of this type were in existence on this instance.</td><td>Secs</td><td>Ideally, the value of this measure should be low.</td></tr>
<tr><td></td><td>Average wait time:
Indicates the average duration of time in which this wait event was persistent since the last measurement period.</td><td>Secs</td><td>Ideally, the value of this measure should be low. A very high value or a consistent increase in this value is indicative of a problem condition which requires further investigation. Use the detailed diagnosis capability to zoom into the session that has contributed to the abnormal increase in wait time.</td></tr>
</table>

10.1.6 Oracle RAC Session Module Waits Test

For each session module on a monitored instance, this test reports the number and nature of wait events that occurred during the last measurement period. In addition, the test also reports the total number of events (of a type) that occurred across all modules and instances. With the help of these metrics, administrators can figure out how much

time an instance has spent waiting and what it was waiting for; a high value is a cause for concern, as it indicates that the instance has waited too long, and could have consequently suffered significant processing delays.

Purpose	Reports the number and nature of wait events that occurred during the last measurement period. In addition, the test also reports the total number of events (of a type) that occurred across all modules and instances.
Target of the test	Oracle RAC
Agent deploying the test	An internal/remote agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">9. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

10. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every session module on each instance of the monitored Oracle RAC</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Cpu waits:
Indicates the number of times since the last measurement period this session module on this instance has waited for CPU.</td><td>Number</td><td>Ideally, the value of this measure should be low.</td></tr>
<tr><td>User IO waits:
Indicates the total number of waits that have occurred in this session module on this instance for user I/O, since the last measurement period.</td><td>Number</td><td>Some of the user I/O wait events include: BFILE read, buffer read retry, db file parallel read, db file scattered read, db file sequential read, and db file single write.
Ideally, the value of this measure should be low.</td></tr>
<tr><td>Cluster waits:
Indicates the total number of waits pertaining to cluster resources that have occurred in this session module on this instance since the last measurement period.</td><td>Number</td><td>A low value is desired for this measure.</td></tr>
<tr><td></td><td>Commited waits:
Indicates the total number of commit waits that have occurred in this session module on this instance since the last measurement period.</td><td>Number</td><td>A low value is desired for this measure.
If this measure shows a high value, refer to the detailed diagnosis capability of this measure to identify the sessions that are affected by committed waits.</td></tr>
</table>

	Transaction lock waits: Indicates the number of times this session module on this instance waited for transaction locks, during the last measurement period.	Number	A low value is desired for this measure. If this measure shows a high value, use the detailed diagnosis of the *RAC Transaction Locks* test to obtain the detailed report on all the sessions that are affected by the lock waits.
	Latch waits: Indicates the total number of latch waits that have occurred in this session module since the last measurement period.	Number	A low value is desired for this measure.
	Other waits: Indicates the total number of times since the last measurement period waits that should typically not occur on a system (eg., 'wait for EMON to spawn') occurred in this session module on this instance.	Number	Ideally, the value of this measure should be low.

10.1.7 Oracle RAC Session Waits Test

This test reports the number and nature of session wait events that occurred on each instance of the monitored RAC during the last measurement period. In addition, the test also reports the total number of events (of a type) that occurred across all instances. With the help of these metrics, administrators can figure out how much time an instance has spent waiting and what it was waiting for; a high value is a cause for concern, as it indicates that the instance has waited too long, and could have consequently suffered significant slowdowns.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the number and nature of session wait events that occurred on each instance of the monitored RAC during the last measurement period. In addition, the test also reports the total number of events (of a type) that occurred across all instances
Target of the test	Oracle RAC
Agent deploying the test	An internal/remote agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">9. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

10. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each instance of the monitored Oracle RAC</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Cpu waits:
Indicates the number of times since the last measurement period the sessions on this instance waited for CPU.</td><td>Number</td><td>Ideally, the value of this measure should be low.</td></tr>
<tr><td>Io waits:
Indicates the total number of times the sessions on this instance waited for user I/O, since the last measurement period.</td><td>Number</td><td>Some of the user I/O wait events include: BFILE read, buffer read retry, db file parallel read, db file scattered read, db file sequential read, and db file single write.
Ideally, the value of this measure should be low.</td></tr>
<tr><td>Cluster waits:
Indicates the total number of times the sessions on this instance waited for cluster resources since the last measurement period.</td><td>Number</td><td>A low value is desired for this measure.</td></tr>
<tr><td></td><td>Commited waits:
Indicates the total number of commit waits that have occurred in the sessions of this instance since the last measurement period.</td><td>Number</td><td>A low value is desired for this measure.
If this measure shows a high value, refer to the detailed diagnosis capability of this measure to identify the sessions that are affected by committed waits.</td></tr>
</table>

	Transaction lock waits: Indicates the number of times the sessions on this instance waited for transaction locks, during the last measurement period.	Number	A low value is desired for this measure. If this measure shows a high value, use the detailed diagnosis of the *RAC Transaction Locks* test to obtain the detailed report on all the sessions that are affected by the lock waits.
	Latch waits: Indicates the total number of latch waits that have occurred in the sessions of this instance since the last measurement period.	Number	A low value is desired for this measure.
	Configuration waits: Indicates the total number of configuration wait events that occurred in the sessions of this instance during the last measurement period.	Number	Configuration waits are waits caused by inadequate configuration of database or instance resources (for example, undersized log file sizes, shared pool size). Ideally, the value of this measure should be low.
	System IO waits: Indicates the total number of system I/O wait events that occurred in the sessions of this instance during the last measurement period.	Number	System I/O waits are waits for background process I/O (for example, DBWR wait for 'db file parallel write'). Ideally, the value of this measure should be low.
	Network waits: Indicates the total number of times since the last measurement period waits related to network messaging (for example, 'SQL*Net more data to dblink', occurred in the sessions of this instance.	Number	A low value is desired for this measure.
	Other waits: Indicates the total number of times since the last measurement period waits that should typically not occur on a system (eg., 'wait for EMON to spawn') occurred in the sessions of this instance.	Number	Ideally, the value of this measure should be low.

	Subtotal waits: Indicates the overall number of waits that have occurred since the last measurement period in the sessions of this Oracle instance.	Number	Ideally, the value of this measure should be low.

10.1.8 Oracle RAC Top Undo Sessions Test

Every Oracle Database must have a method of maintaining information that is used to roll back, or undo, changes to the database. Such information consists of records of the actions of transactions, primarily before they are committed. These records are collectively referred to as **undo**.

Undo records are used to:

- Roll back transactions when a **ROLLBACK** statement is issued
- Recover the database
- Provide read consistency
- Analyze data as of an earlier point in time by using Oracle Flashback Query
- Recover from logical corruptions using Oracle Flashback features

When a **ROLLBACK** statement is issued, undo records are used to undo changes that were made to the database by the uncommitted transaction. During database recovery, undo records are used to undo any uncommitted changes applied from the redo log to the datafiles. Undo records provide read consistency by maintaining the before image of the data for users who are accessing the data at the same time that another user is changing it.

Oracle provides a fully automated mechanism, referred to as automatic undo management, for managing undo information and space. In this management mode, you create an **undo tablespace**, and the server automatically manages undo segments and space among the various active sessions.

Each instance in the RAC system can only use one undo tablespace at a time. In other words, instances cannot share undo tablespaces. Each instance in the cluster, being an independent transaction-processing environment, maintains its own UNDO area for undo management. The RAC system allows the creation and use of several undo tablespaces. When the instance is started, it uses the first available undo tablespace. A second instance will use another undo tablespace. Thus, each instance in a RAC system will have exclusive access to a particular undo tablespace at a given time. The undo tablespace cannot be shared among the instances at the same time. Only once an undo tablespace is released by an instance, it can be assigned to another instance. However, all instances can read blocks from any or all undo tablespaces for the purpose of constructing read-consistency images.

You need to closely observe how the sessions to each RAC instance use the undo tablespaces; this will enable you to proactively detect unusually high/long usage conditions. The **RAC Top Undo Sessions** test brings such anomalies to light. This test reports the number of sessions (per instance) accessing the undo tablespace and the duration of usage of these sessions, thus indicating excessive usage (if any) of the undo tablespace. The detailed diagnosis capability of the test turns the spotlight on those sessions that are the leading users of the undo tablespace, and provides pointers to the query executed by these sessions. With the help of this information you can identify inefficient queries and fine-tune them, so that potential processing delays and consequent instance slowdowns/crashes can be averted.

Purpose	Reports the number of sessions (per instance) accessing the undo tablespace and the duration of usage of these sessions, thus indicating excessive usage (if any) of the undo tablespace.

Target of the test	Oracle RAC
Agent deploying the test	An internal/remote agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 10. **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.		
Outputs of the test	One set of results for each instance of the monitored Oracle RAC		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Average duration:** Indicates the average time taken by the sessions to this instance, to execute queries on the undo tablespace.	Secs	Ideally, the value of this measure should be low. An unusually high value for this measure could indicate that one/more sessions are using the undo tablespace for too long a time. Use the detailed diagnosis of this measure to identify the top sessions in terms of duration of usage of the undo tablespace, and determine the SQL ID of the query executed by each session on that tablespace. Inefficient queries can thus be isolated. Fine-tuning these queries will enable the optimal usage of the undo tablespace. Query processing will also be delayed if the undo tablespace is improperly sized. If the undo tablespace has insufficient space, many transactions to the tablespace may terminate before completion, and many more transactions may even hang; this will result in a long line of long-running queries. Also, since Oracle automatically tunes the undo retention period based on undo tablespace size and system activity, if the undo tablespace runs out of space, it will grossly affect the auto retention capability of Oracle, once again causing query failures. When available space for new transactions becomes short, the database begins to overwrite expired undo. If the undo tablespace has no space for new transactions after all expired undo is overwritten, the database may begin overwriting unexpired undo information. If any of this overwritten undo information is required for consistent read in a current long-running query, the query could fail with the **snapshot too old** error message.
	Number of sessions: Indicates the number of sessions to this instance that are utilizing the undo tablespace.	Number	This serves as a good indicator of the load on the undo tablespace.

10.1.9 Oracle RAC SQL Network Test

This test executes an external test that emulates a query to the cluster to determine its availability and responsiveness. The test sends the emulated request to the virtual cluster server (i.e., the *Oracle Cluster*), which will promptly forward the request to that node in the cluster that currently owns the cluster server. If at least one node in the cluster is currently active, then the query will successfully execute on that node, and report the good health of the cluster. On the other hand, if none of the nodes in the cluster are active, then the query will be unable to execute, and the test will hence report the non-availability of the cluster.

Purpose	Executes an external test that emulates a query to the cluster to determine its availability and responsiveness
Target of the test	Oracle RAC
Agent deploying the test	An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the target Oracle cluster is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the target Oracle cluster is listening.

Configurable parameters for 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 on which the server is listening 4. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 5. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 6. **PASSWORD** – Password of the specified database user 7. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 8. **ORASID** - The variable name of the oracle instance. 9. **TIMEOUT** - Specify the duration (in seconds) beyond which the test will timeout if no response is received from the server. The default value is 30 seconds.

<table>
<tr><td></td><td colspan="3">10. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
11. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID (instance) monitored.</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Oracle server availability:
Whether the cluster is responding to requests.</td><td>Percent</td><td>The availability is 100% when the cluster is responding to a request and 0% when it is not. Availability problems may be caused if none of the nodes of the cluster are currently operational.</td></tr>
<tr><td>Total response time:
The time taken by the cluster to respond to a user query.</td><td>Secs</td><td>A sudden increase in response time is indicative of a bottleneck at the cluster.</td></tr>
</table>

10.1.10 Oracle RAC Cluster Interconnects Test

A cluster database comprises two or more nodes that are linked by an interconnect. The interconnect serves as the communication path between the nodes in the cluster database. Each Oracle instance uses the interconnect for the messaging that synchronizes each instance's use of shared resources. Oracle also uses the interconnect to transmit data blocks that the multiple instances share.

The non-availability of the interconnect on any cluster node can impair that node's communication with other nodes in the cluster. As a result, fail-over operations will be hampered and the cluster service will be forced to distribute session/request load across the remaining clusters in the node; this in turn may overload the other nodes in the cluster. In the aftermath of this, mission-critical business services using the clustered resources may experience prolonged outages or slowdowns, resulting in considerable loss of revenue and reputation.

To avoid this, administrators need to continuously monitor the availability of the cluster interconnect on each node, analyze how session/process load is distributed across the nodes via the interconnect, and proactively detect the following:

- The sudden unavailability of the interconnect on a node;
- How the unavailability of an interconnect affects the load on the other nodes in the cluster;

For this purpose, you can use the **Oracle Cluster Interconnects** test. This test periodically verifies whether the nodes in

the cluster are able to communicate via the cluster interconnect, and promptly reports the non-availability of the interconnect. In addition, the test also keeps tabs on the session and process load on each node in the cluster, thus promptly revealing the impact of the unavailability of a cluster interconnect on the load and performance of other nodes in the cluster.

Purpose	Periodically verifies whether the nodes in the cluster are able to communicate via the cluster interconnect, and promptly reports the non-availability of the interconnect. In addition, the test also keeps tabs on the session and process load on each node in the cluster, thus promptly revealing the impact of the unavailability of a cluster interconnect on the load and performance of other nodes in the cluster.
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance. 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **HIDE IP** – This test reports a set of metrics for that IP address on each cluster node using which a node communicates with other nodes in the cluster. The descriptors of this test therefore will be of the following format by default: *clusternodeID_<IP_used_for_internode_communication>*. Accordingly, the **HIDE IP** parameter is set to **No** by default. High security environments however may not want to expose the IP address that cluster nodes use for internal communication. In such environments, you can set the **HIDE IP** flag to **No**, so that the descriptors of this test do not include the *<IP_used_for_internode_communication>*. In such cases therefore, only the *clusternodeID* will be displayed as the descriptors of this test. 10. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for each *clusternodeID_<IP_address_used_for_internode_communication>* in the Oracle cluster		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Cluster interconnect percentage: Indicates whether the cluster interconnect isavailable on this node or not.	Percent	The value 0 for this measure indicates that this node is unable to communicate with other nodes in the cluster via the cluster interconnect. The value 100 indicates that the interconnect is available and is enabling this node to communicate with the other cluster nodes.
	Logon rate : Indicates the rate at which user logons occurred on this node.	Number/sec	
	Processes utilization : Indicates the number of processes currently running on this cluster node.	Number	As long as the value of this measure is much lower than the value of the **PROCESSES** setting in the database parameter file, the node will be able to handle the process load.

	Processes utilization percentage : Of the maximum number of processes this node can handle, what percentage is currently active on this cluster node.	Percent	Ideally, the value of this measure should be low. If this measure value is close to 100%, it could mean that the node is about to exhaust its processing limit and may not be able to handle any more processes. On the other hand, if the value of this measure is consistently high for a cluster node, then check the **PROCESSES** setting in the database parameter file to figure out whether/not the node has been configured with adequate processing capability. If this check reveals that the node has been configured with a limited number of processes than it can handle, you may want to increase the **PROCESSES** setting to suit the node's capacity.
	Session utilization : Indicates the number of sessions that are currently active on this node.	Number	As long as the value of this measure is much lower than the value of the **SESSIONS** setting in the database parameter file, the node will be able to handle the session load. If the value of this measure is unusually high for any cluster node, then compare the value of this measure across nodes to figure out whether/not load is uniformly distributed across all cluster nodes. If session load on most of the cluster nodes is high, then the sudden increase in session load could be attributed to an unavailable cluster interconnect. Because of the unavailability, the cluster service may not have been unable to contact the affected cluster node and may have been compelled to distribute the load amongst the remaining cluster nodes. This may have caused load on the other nodes to suddenly increase. To confirm this, check the value of the *Interconnect availability percentage* measure of all nodes. On the other hand, if no interconnect is unavailable, and if *Session utilization* is abnormally high on a particular node only, it could mean that that node is indeed overloaded.

	Session utilization percentage : Of the maximum number of sessions this node can handle, what percentage is currently active on this cluster node.	Percent	Ideally, the value of this measure should be low. If this measure value is close to 100%, it could mean that the node may not be able to handle any more sessions. On the other hand, if the value of this measure is consistently high for a cluster node, then check the **SESSIONS** setting in the database parameter file to figure out whether/not the node has been configured with adequate session-handling capability. If this check reveals that the node has been configured with a limited number of sessions than it can handle, you may want to increase the **SESSIONS** setting to suit the node's true capacity.

10.1.11 Oracle RAC CR Block Requests Test

Data blocks requested from the Global Cache are of two types: current and consistent-read (CR) blocks. When you update data in the database, Oracle Database must locate the most recent version of the data block that contains the data, which is called the current block. If you perform a query, only data committed before the query began is visible to the query. Data blocks that were changed after the start of the query are reconstructed from data in the undo segments, and the reconstructed data is made available to the query in the form of a consistent-read block.

Whenever a session requests for a CR block, Oracle first checks whether it has that block in its local cache. If the block does not exist in the local cache but is available in the remote cache, then it is transferred from the remote to local cache via the interconnect. The time that elapses between when a CR block is requested and when the session receives it should be tracked continuously, so that global cache block access latencies (if any) are detected proactively and resolved promptly. Use the **Oracle CR Block Requests** test to perform this tracking.

This test, at configured intervals, monitors requests for CR blocks and reports how long it took for the requests to be serviced by the buffer caches. This sheds light on request processing delays (if any).

Purpose	Monitors requests for CR blocks and reports how long it took for the requests to be serviced by the buffer caches. This sheds light on request processing delays (if any)
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the Oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every Oracle cluster		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**

	Average cr block request time: Indicates the time taken to service requests for cr blocks.	Centisecs	Ideally the value of this measure should be low. A high value for this measure is indicative of high latencies while accessing global cache blocks. This can be caused by any of the following: • A high number of requests caused by SQL statements that are not tuned. • A large number of processes in the queue waiting for CPU, or scheduling delays. • Slow, busy, or faulty interconnects. In these cases, check your network connection for dropped packets, retransmittals, or cyclic redundancy check (CRC) errors. When global cache requests cause a performance problem, optimizing SQL plans and the schema to improve the rate at which data blocks are located in the local buffer cache, and minimizing I/O is a successful strategy for performance tuning.

10.1.12 Oracle RAC Current Block Requests Test

Data blocks requested from the Global Cache are of two types: current and consistent-read (CR) blocks. When you update data in the database, Oracle Database must locate the most recent version of the data block that contains the data, which is called the current block. If you perform a query, only data committed before the query began is visible to the query. Data blocks that were changed after the start of the query are reconstructed from data in the undo segments, and the reconstructed data is made available to the query in the form of a consistent-read block.

Whenever a session requests for a current block, Oracle first checks whether the block is present in the local cache. If not, it looks for the same in the remote cache. If the block is available in the remote cache,it pins the block in the exclusive mode, flushes it from the cache, and sends it across over the interconnect. The time that elapses between when a current block is requested and when the session receives it should be tracked continuously, so that global cache block access latencies (if any) are detected proactively and resolved promptly. Use the **Oracle Current Block Requests** test to perform this tracking.

This test, at configured intervals, monitors requests for current blocks and reports how long it took for the requests to be serviced by the buffer caches. This sheds light on request processing delays (if any).

Purpose	Monitors requests for current blocks and reports how long it took for the requests to be serviced by the buffer caches. This sheds light on request processing delays (if any)
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the Oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every Oracle cluster		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Average current block request time: Indicates the time taken to service requests for current blocks.	Centisecs	Ideally the value of this measure should be low. A high value for this measure is indicative of high latencies while accessing the global cache. This can be caused by any of the following: • A high number of requests caused by SQL statements that are not tuned. • A large number of processes in the queue waiting for CPU, or scheduling delays. • Slow, busy, or faulty interconnects. In these cases, check your network connection for dropped packets, retransmittals, or cyclic redundancy check (CRC) errors. When global cache requests cause a performance problem, optimizing SQL plans and the schema to improve the rate at which data blocks are located in the local buffer cache, and minimizing I/O is a successful strategy for performance tuning.

10.1.13 Oracle RAC Global Cache Corrupt Blocks Test

This test reports the number of blocks that were corrupted while being transferred between Oracle instances through

the private interconnect in this Oracle RAC.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the number of blocks that were corrupted while being transferred between Oracle instances through the private interconnect in this Oracle RAC
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the Oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every Oracle cluster		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Block corrupted count: Indicates the number of blocks that were corrupted during transfer through the private interconnect.	Number	Ideally the value of this measure should be zero. A high value indicates the possibility of an IPC, network or hardware problem.

10.1.14 Oracle RAC Global Cache Lost Blocks Test

This test reports the number of blocks that were lost during transfer from one Oracle instance to another through private interconnects in this Oracle RAC.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the number of blocks that were lost during transfer from one Oracle instance to another through private interconnects in this Oracle RAC
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the Oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every Oracle cluster		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Block lost count: Indicates the number of blocks that were lost during transfer from one Oracle instance to another.	Number	Ideally the value of this measure should be zero. A high value is indicative of network problems. The use of an unreliable IPC protocol such as UDP may result in the value for global cache blocks lost being non-zero. This ratio should be as small as possible. Many times, a non-zero value for global cache blocks lost does not indicate a problem because Oracle will retry the block transfer operation until it is successful.

10.1.15 Oracle RAC Scans Test

Full table scans on a database instance can degrade the performance of the database. This test monitors the extent of full table scans happening on each database in the Oracle RAC.

Purpose	Monitors the extent of full table scans happening on each database in the Oracle RAC
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the Oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for each database on an Oracle cluster		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Percent long table scans: Indicates the percentage of long table scans happening on this database.	Percent	Ideally, this value should be lower than 10%. If more than 20% of scans are happening on long tables, the database/accesses to the database may need to be tuned. Full table scans may happen due to several reasons. For instance, the indexes of a table may not be used properly in queries. By tuning the queries, the full table scans can be reduced and the database performance significantly improved.
	Long table scans: Indicates the number of long table scans that happened on each database in this Oracle RAC during the last measurement period.	Number	
	Short table scans: Indicates the number of short table scans that happened on each database in this Oracle RAC during the last measurement period.	Number	

10.1.16 Oracle RAC Cluster Nodes Test

This test reports the count of nodes in the Oracle cluster and indicates the number and names of those nodes that are currently accessible. This way, the nodes that are inaccessible/unavailable can be identified. The test also reports the percentage of available nodes, and thus indicates if only very few nodes in the cluster are able to service the client requests to the cluster. This signals a potential overload.

Purpose	Reports the count of nodes in the Oracle cluster and indicates the number and names of those nodes that are currently accessible
Target of the test	Oracle RAC

Agent deploying the test	An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the target Oracle cluster is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the target Oracle cluster is listening.

Configurable parameters for 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 on which the server is listening 4. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 5. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 6. **PASSWORD** – Password of the specified database user 7. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 8. **ORASID** - The variable name of the oracle instance. 9. **TIMEOUT** - Specify the duration (in seconds) beyond which the test will timeout if no response is received from the server. The default value is 30 seconds.

<table>
<tr><td></td><td colspan="3">10. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
11. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every SID (instance) monitored.</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total nodes:
Indicates the number of nodes in the cluster.</td><td>Number</td><td></td></tr>
<tr><td>Available nodes:
Indicates the number of nodes in the cluster that are currently accessible.</td><td>Number</td><td>Use the detailed diagnosis of this measure to know which nodes in the cluster are currently accessible.</td></tr>
<tr><td>Unavailable nodes:
Indicates the number of nodes in the cluster that are currently unavailable.</td><td>Number</td><td></td></tr>
<tr><td></td><td>Percentage of available nodes:
Indicates the percentage of nodes in the cluster that are available.</td><td>Percent</td><td>A high value is desired for this measure.</td></tr>
</table>

10.2 The Memory Structures Layer

A transaction lock is acquired when a transaction initiates its first change and is held until the transaction does a COMMIT or ROLLBACK. It is used mainly as a queuing mechanism so that other sessions can wait for the transaction

to complete.

The test mapped to this layer reports the number and duration of transaction locks held by each instance of the cluster.

Figure 10.4: The tests mapped to the Memory Structures layer

10.2.1 Oracle RAC Transaction Locks Test

A transaction lock held for too long a time will prevent other sessions from accessing the database object, thereby stalling critical database operations. It is hence imperative to monitor the transaction locks to each database instance in an Oracle RAC. Using the **RAC Transaction Locks** test, you can determine the number of transaction locks held by each database instance and the duration of these locks, so that you can quickly identify the instance holding a large number of transaction locks and that which is holding locks for an unreasonably long time.

Purpose	Helps you determine the number of transaction locks held by each database instance and the duration of these locks, so that you can quickly identify the instance holding a large number of transaction locks and that which is holding locks for an unreasonably long time
Target of the test	Oracle RAC
Agent deploying the test	An internal/remote agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running.

	Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for each instance of the monitored Oracle RAC		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Number of Locks: Indicates the number of locks held by this instance.	Number	A high value may indicate one of the following: • Too many transactions happening • Locked resources not being released properly • Locks are being held unnecessarily.
	Average wait time: Indicates the time for which the locks were held by this instance.	Secs	A high value may indicate one of the following: • Too many transactions happening • Locked resources not being released properly • Locks are being held unnecessarily.

10.3 The Tablespaces Layer

The tests mapped to this layer proactively alert administrators to potential space constraints in the tablespaces and temporary tablespaces of the cluster, and reveals long-running queies to the undo tablespaces of the cluster.

Figure 10.5: The tests mapped to the Tablespaces layer

10.3.1 Oracle RAC Tablespaces Test

Tablespaces should be adequately sized. If not, then the tablespaces may frequently run very low on free space, causing all statements that attempt to acquire new space in the tablespace to fail. This in turn will result in serious performance issues ranging from slowdowns to shutdowns. Continuous monitoring of tablespace size and usage is hence important.

The **RAC Tablespaces** test auto-discovers the tablespaces managed by the Oracle RAC, and reports how well every tablespace has been utilized. In the process, the test also reveals the type of database objects (tables, indexes, partitions, LOB segments, etc.) that are occupying space in the tablespace. This way, you will be able to instantly identify the tablespace left with very little free space, and also zero-in on those objects that could be eroding space in that tablespace.

Purpose	Auto-discovers the tablespaces managed by the Oracle RAC, and reports how well every tablespace has been utilized
Target of the test	Oracle RAC
Agent deploying the test	An internal/remote agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for each tablespace managed by the monitored Oracle RAC		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Table size: Indicates the size of the tables present in this tablespace.	GB	A high value for these measures indicates that a tables and indexes consume a large chunk of space in the tablespace.
	Index size Indicates the space consumed by the indexes on the tables present in this tablespace.	GB	
	Table partition size: Indicates the size of all partitions of tables present in this tablespace.	GB	**Partitioning** addresses key issues in supporting very large tables and indexes by letting you decompose them into smaller and more manageable pieces called **partitions**. Each partition of a table or index must have the same logical attributes, such as column names, datatypes, and constraints, but each partition can have separate physical attributes such as pctfree, pctused, and tablespaces. A high value for these measures indicates that table and index partitions consume a large chunk of space in the tablespace.
	Index partiton size: Indicates the size of all partitions of indexes present in this tablespace.	GB	
	Lob segment size: Indicates the size of LOB segments in this tablespace.	GB	LOBs (Large Object) are Oracle's data structures designed to store and retrieve large amounts of unstructured data such as video, audio, photo images, etc within the database. Whenever a table containing a LOB column is created, two segments are created to hold the specified LOB column. These segments are of type LOBSEGMENT and LOBINDEX. The LOBINDEX segment is used to access LOB chunks/pages stored in the LOBSEGMENT segment. The values of these measures report the size of the LOBSEGMENT sements and the LOBINDEX segments (respectively). A high value for these measures, quiet naturally, indicates that too many large objects are stored in the tablespace.
	Lob index size: Indicates the size of LOB indexes in this tablespace.	GB	

	Lob partition size: Indicates the size of table partitions with LOBs in this tablespace.	GB	You can partition tables with LOBs. As a result, LOBs can take advantage of all of the benefits of partitioning. For example, LOB segments can be spread between several tablespaces to balance I/O load and to make backup and recovery more manageable. LOBs in a partitioned table also become easier to maintain. A high value for this measure indicates that table partitions containing LOB columns are consuming a large amount of space in this tablespace.
	Maximum size: Indicates the maximum extent upto which a tablespace can grow.	GB	
	Used size: Indicates the size upto which this tablespace has been utilized.	GB	If this value is very high, it indicates that the tablespace memory is almost full.
	Free size: Indicates the amount of unused space available in this tablespace.	GB	If this value is very low, then it indicates over-utilization of the tablespace.
	Free space usage: Indicates the space available for overall growth expressed as a ratio of *Free size* with respect to the *Maximum size* of the tablespace.	Percent	If this value is very low, then it indicates over-utilization of the tablespace. Also, if the value of this measure is below 80 %, then sufficient space must be allocated to the tablespace.

10.3.2 Oracle RAC Temp Tablespaces Test

A temporary tablespace, contrary to what the name might indicate, does exist on a permanent basis as do other tablespaces, such as the System and Sysaux tablespaces. However the data in a temporary tablespace is of a temporary nature, which persists only for the length of a user session. Oracle uses temporary tablespaces as work areas for tasks such as sort operations for users and sorting during index creation. Oracle does not allow users to create objects in a temporary tablespace. By definintion, the temporary tablespace holds data only for the duration of the user's session, and the data can be shared by all users.

Sufficient free space should be available in the temporary tablespace, as critical operations such as sorting and execution of hash-intensive queries may otherwise fail. Periodically checking the space usage in the temporary tablespaces will provide you with early warning signals of potential space contentions. The **RAC Temp Tablespaces** test monitors the usage of the temporary tablespace in each instance of the Oracle RAC, and proactively reports which temporary tablespace is running dangerously low on free space. Moreover, the test also reports the usage of the temporary tablespace across instances.

Purpose	Monitors the usage of the temporary tablespace in each instance of the Oracle RAC, and proactively reports which temporary tablespace is running dangerously low on free space

Target of the test	Oracle RAC
Agent deploying the test	An internal/remote agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running.

	Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for the temporary tablespace of each instance managed by the monitored Oracle RAC		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Used space: Indicates the space used by this temporary tablespace.	MB	If this value is very high, then it indicates that the memory of this tablespace is almost full.
	Free space: Indicates the amount of unused space in this temporary tablespace.	MB	If this value is very low, then it indicates over-utilization of the tablespace.
	Total space: Indicates the total amount of space allocated for this temporary tablespace.	MB	
	Max space: Indicates the maximum extent upto which this temporary tablespace can grow.	MB	
	Used percentage: Indicates the percentage of space used in this temporary tablespace.	Percent	If this value is very high, then it indicates over-utilization of the tablespace.
	Free percentage: Indicates the space available for overall growth expressed as a ratio of *Free space* with respect to the *Max space* of this temporary tablespace. The formula used is: *Free_space/Max_bytes*100*	Percent	If this value is very low, it indicates that more space needs to be allotted to this tablespace to ensure that critical operations do not fail.

10.3.3 Oracle RAC Undo Usage SqlId Test

Every Oracle Database must have a method of maintaining information that is used to roll back, or undo, changes to the database. Such information consists of records of the actions of transactions, primarily before they are committed. These records are collectively referred to as undo. Undo records are used to:

- Roll back transactions when a ROLLBACK statement is issued

- Recover the database
- Provide read consistency
- Analyze data as of an earlier point in time by using Oracle Flashback Query
- Recover from logical corruptions using Oracle Flashback features

When a ROLLBACK statement is issued, undo records are used to undo changes that were made to the database by the uncommitted transaction. During database recovery, undo records are used to undo any uncommitted changes applied from the redo log to the datafiles. Undo records provide read consistency by maintaining the before image of the data for users who are accessing the data at the same time that another user is changing it.

Oracle provides a fully automated mechanism, referred to as automatic undo management, for managing undo information and space. In this management mode, you create an undo tablespace, and the server automatically manages undo segments and space among the various active sessions.

Each instance in the RAC system can only use one undo tablespace at a time. In other words, instances cannot share undo tablespaces. Each instance in the cluster, being an independent transaction-processing environment, maintains its own UNDO area for undo management. The RAC system allows the creation and use of several undo tablespaces. When the instance is started, it uses the first available undo tablespace. A second instance will use another undo tablespace. Thus, each instance in a RAC system will have exclusive access to a particular undo tablespace at a given time. The undo tablespace cannot be shared among the instances at the same time. Only once an undo tablespace is released by an instance, it can be assigned to another instance. However, all instances can read blocks from any or all undo tablespaces for the purpose of constructing read-consistency images.

If instances take too long a time to read from or write to a undo tablespace, it will unnecessarily delay the recovery/rollback process and sometimes even abnormally terminate it, causing serious data inconsistencies. To avoid such adversities, it is imperative that you monitor how long the undo tablespace takes to execute queries, and promptly detect latencies in query execution. The **Oracle RAC Undo Usage SqlD** test enables you to achieve the same. The test also points you to time-consuming SQL queries to the undo tablespace, so that you can fine-tune them. In the process, the test also monitors the contents of the undo tablespace used by each instance and their undo retention period.

Purpose	Monitor how long the undo tablespace takes to execute queries, and promptly detect latencies in query execution; also monitors the contents of the undo tablespace used by each instance and their undo retention period
Target of the test	Oracle RAC
Agent deploying the test	An internal/remote agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running.

	Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for the undo tablespace of each instance managed by the monitored Oracle RAC		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Maximum query time: Indicates the time taken for executing a query on this tablespace in this instance.	Secs	Ideally, the value of this measure should be low. An unusually high value for this measure could indicate that one/more queries are taking too long to execute on the undo tablespace. Use the detailed diagnosis of this measure to identify the the SQL IDs of teh top queries to the undo tablespace in terms of duration of execution. Inefficient queries can thus be isolated. Fine-tuning these queries will enable the optimal usage of the undo tablespace. Query processing will also be delayed if the undo tablespace is improperly sized. If the undo tablespace has insufficient space, many transactions to the tablespace may terminate before completion, and many more transactions may even hang; this will result in a long line of long-running queries. Also, since Oracle automatically tunes the undo retention period based on undo tablespace size and system activity, if the undo tablespace runs out of space, it will grossly affect the auto retention capability of Oracle, once again causing query failures. When available space for new transactions becomes short, the database begins to overwrite expired undo. If the undo tablespace has no space for new transactions after all expired undo is overwritten, the database may begin overwriting unexpired undo information. If any of this overwritten undo information is required for consistent read in a current long-running query, the query could fail with the **snapshot too old** error message.
	Active blocks: Indicates the number of active blocks available in this undo tablespace.	Number	These blocks consist of undo data that supports active transactions and are required in the event of rollback.

	Unexpired blocks: Indicates the number of unexpired blocks available in this undo tablespace.	Number	After a transaction is committed, undo data is no longer needed for rollback or transaction recovery purposes. However, for consistent read purposes, long-running queries may require this old undo information for producing older images of data blocks. Such type of blocks that are needed for supporting the UNDO_RETENTION parameter is called as unexpired blocks.
	Expired blocks: Indicates the number of expired blocks available in this undo tablespace.	Number	The undo information that is no longer needed for rollback is stored in these expired blocks. These type of blocks will be over written by fresh information as and when required.
	Tuned undo retention: Indicates the time taken for undo retention of data blocks in this undo tablespace.	Secs	After a transaction is committed, undo data is no longer needed for rollback or transaction recovery purposes. However, for consistent read purposes, long-running queries may require this old undo information for producing older images of data blocks. Furthermore, the success of several Oracle Flashback features can also depend upon the availability of older undo information. For these reasons, it is desirable to retain the old undo information for as long as possible. When automatic undo management is enabled, there is always a current undo retention period, which is the minimum amount of time that Oracle Database attempts to retain old undo information before overwriting it. Old (committed) undo information that is older than the current undo retention period is said to be expired. Old undo information with an age that is less than the current undo retention period is said to be unexpired. Oracle Database automatically tunes the undo retention period based on undo tablespace size and system activity. You can specify a minimum undo retention period (in seconds) by setting the UNDO_RETENTION initialization parameter. The database makes its best effort to honor the specified minimum undo retention period, provided that the undo tablespace has space available for new transactions.

			When available space for new transactions becomes short, the database begins to overwrite expired undo. If the undo tablespace has no space for new transactions after all expired undo is overwritten, the database may begin overwriting unexpired undo information. If any of this overwritten undo information is required for consistent read in a current long-running query, the query could fail with the snapshot too old error message. The following points explain the exact impact of the UNDO_RETENTION parameter on undo retention: The UNDO_RETENTION parameter is ignored for a fixed size undo tablespace. The database may overwrite unexpired undo information when tablespace space becomes low. For an undo tablespace with the AUTOEXTEND option enabled, the database attempts to honor the minimum retention period specified by UNDO_RETENTION. When space is low, instead of overwriting unexpired undo information, the tablespace auto-extends. If the MAXSIZE clause is specified for an auto-extending undo tablespace, when the maximum size is reached, the database may begin to overwrite unexpired undo information.

10.3.4 Oracle RAC Flash Area Usage Test

The Flash Recovery Area is a specific area of disk storage that is set aside exclusively for retention of backup components such as datafile image copies, archived redo logs, and control file autobackup copies. These features include:

- **Unified Backup Files Storage.** All backup components can be stored in one consolidated spot. The Flash Recovery Area is managed via Oracle Managed Files (OMF), and it can utilize disk resources managed by Oracle Automated Storage Management (ASM). In addition, the Flash Recovery Area can be configured for use by multiple database instances if so desired.
- **Automated Disk-Based Backup and Recovery.** Once the Flash Recovery Area is configured, all backup components (datafile image copies, archived redo logs, and so on) are managed automatically by Oracle.
- **Automatic Deletion of Backup Components.** Once backup components have been successfully created, RMAN can be configured to automatically clean up files that are no longer needed (thus reducing risk of insufficient disk space for backups).
- **Disk Cache for Tape Copies.** Finally, if your disaster recovery plan involves backing up to alternate media, the Flash Recovery Area can act as a disk cache area for those backup components that are eventually copied to tape.

- **Flashback Logs.** The Flash Recovery Area is also used to store and manage flashback logs, which are used during Flashback Backup operations to quickly restore a database to a prior desired state.

Oracle recommends that the Flash Recovery Area should be sized large enough to include all files required for backup and recovery. Using this test, administrators can figure out whether the Flash Recovery Area is adequately sized or not, and accordingly make sizing recommendations.

This test is applicable only to clusters based on Oracle database server 10g (and above).

Purpose	Monitors the usage of the Flash recovery area
Target of the test	An Oracle database server 10g
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for the flash recovery area on each node in the Oracle cluster		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Used flash area: Indicates the space currently occupied by the flash recovery files on this node.	MB	
	Maximum flash area size: Indicates the maximum space allocated for flash recovery files on this node.	MB	
	Flash area usage: Indicates the percentage of space occupied by the flash recovery files on this node.	Percent	Oracle recommends that the Flash Recovery Area should be sized large enough to include all files required for backup and recovery. Therefore, ideally, the value of this measure should be very low. A value close to 100% indicates excessive usage of the recovery area; this implies that the flash recovery area could soon run out of space. in such a case you can resize the flash recovery area by reconfiguring the parameter "**db_recovery_file_dest_size**" in database parameter file, provided enough disk space is available. If not, then Oracle recommends that the flash area be sized at least large enough to contain any archived redo logs that have not yet been backed up to alternate media. Alternatively, you can remove the old files from the flash recovery area to create space for the new recovery files.
	Free flash area: Indicates the free space currently available for recovery files on this node.	MB	

10.3.5 Oracle RAC ASM Disk I/O Test

ASM is a volume manager and a file system for Oracle database files that supports single-instance Oracle Database

and Oracle Real Application Cluster (Oracle RAC) configuration. ASM is Oracle's recommended storage management solution that provides an alternative to conventional volume managers, file systems, and raw devices.

ASM uses disk groups to store datafiles; an ASM disk group is a collection of disks that ASM manages as a unit. Within a disk group, ASM exposes a file system interface for Oracle database files. The content of files that are stored in a disk group are evenly distributed, or striped, to eliminate hot spots and to provide uniform performance across the disks.

You need to periodically monitor the read-write activity on each disk in a disk group to make sure that I/O load is uniformly balanced across all disks in a group. The **ASM Disk I/O** test helps you do just that. At pre-configured intervals, this test monitors the I/O activity on each disk in every disk group of an Oracle cluster, reveals I/O-intensive and error-prone disks, and brings irregularities in load balancing to the fore.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors the I/O activity on each disk in every disk group of an Oracle cluster, reveals I/O-intensive and error-prone disks, and brings irregularities in load balancing to the fore
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

10. Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
Outputs of the test	One set of results for each *DiskGroup:Disk* pair in the Oracle cluster being monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Reads: Indicates the rate at which reads occur on this disk.	Reads/Sec	Compare the values of each of these measures across the disks in a disk group to identify the I/O-intensive disks in that group. In the process, you can also determine whether/not I/O load is equally balanced across all the disks in the group. If any irregularities are noticed in load-balancing are noticed, you may want to consider adding more disks to the group.
	Writes: Indicates the rate at which writes occur on this disk.	Writes/Sec	
	Read errors: Indicates the number of errors that occur per second while reading from this disk.	ReadErrors/Sec	The value 0 is desired for both these measures. A non-zero value is indicative of I/O errors. By comparing the values of each of these measures across disks and across disk groups, you can not only point to the error-prone disks and groups, but can also figure out when most of the errors occurred on the disk/group - when reading? or when writing?
	Write errors: Indicates the number of errors that occur per second when writing to this disk on this cluster node.	WriteErrors/Sec	

10.3.6 Oracle RAC ASM Disk Space Test

ASM is a volume manager and a file system for Oracle database files that supports single-instance Oracle Database and Oracle Real Application Cluster (Oracle RAC) configuration. ASM is Oracle's recommended storage management solution that provides an alternative to conventional volume managers, file systems, and raw devices.

ASM uses disk groups to store datafiles; an ASM disk group is a collection of disks that ASM manages as a unit. Within a disk group, ASM exposes a file system interface for Oracle database files. The content of files that are stored in a disk group are evenly distributed, or striped, to eliminate hot spots and to provide uniform performance across the disks.

To ensure that a disk group always has sufficient space to store the critical organizational data, you will have to continuously track the space usage of the disk group. This will provide you with early pointers to potential space contentions and help you swiftly provide more space to the group by adding more disks. The **ASM Disk Space** test enables you to achieve this end. This test closely monitors how each disk in a disk group uses the space available to

it, points you to the disks that are running out of space, and thus holds a mirror to space contentions on a disk group.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Closely monitors how each disk in a disk group uses the space available to it, points you to the disks that are running out of space, and thus holds a mirror to space contentions on a disk group
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for each *DiskGroup:Disk* pair on the Oracle server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Used space: Indicates the amount of space currently used in this disk.	MB	Ideally, the value of this measure should be low. A consistent increase in this value is a cause for concern.
	Free space: Indicates the amount of space in this disk that is currently free - i.e., available for use.	MB	Ideally, the value of this measure should be high. A consistent decrease in this value is a cause for concern.
	Space availability: Indicates the percentage of space in this disk that is currently unused.	Percent	A high value is typically desired for this measure. By comparing the value of this measure across disks and across disk groups, you can quickly isolate the disks/groups that are running short of space. If the free space is alarmingly low for all disks in a group, it indicates that the group requires more space. You can then consider making space by adding more disks to the group.
	Space usage: Indicates the percentage of space in this disk that is currently used.	Percent	A low value is typically desired for this measure. By comparing the value of this measure across disks and across disk groups, you can quickly isolate the disks/groups that are utilizing space excessively. If the used space is alarmingly high for all disks in a group, it indicates that the group is rapidly running out of space. You can then consider making space by adding more disks to the group.

	Used space growth: Indicates the growth in space usage of this disk since the last measurement period.	MB/Sec	If you observe the variations to this measure over time, you will be able to detect early whether the space in the disk is being steadily eroded or not. This way, you can initiate measures to conserve space much before the disk exhausts all the space available to it.

10.3.7 Oracle RAC Root Blockers Test

One common problem encountered with databases is blocking. Suppose that process A is modifying data that process B wants to use. Process B will be blocked until process A has completed what it is doing. This is only one type of blocking situation; others exist and are common. What matters to a database administrator is identifying when blocking is a problem and how to deal with it effectively. When blocking is bad enough, users will notice slowdowns and complain about it. With a large number of users, it is common for tens or hundreds of processes to be blocked when slowdowns are noticed. Killing these processes may or may not solve the problem because 10 processes may be blocked by process B, while process B itself is blocked by process A. Issuing 10 kill statements for the processes blocked by B probably will not help, as new processes will simply become blocked by B. Killing process B may or may not help, because then the next process that was blocked by B, which is given execution time, may get blocked by process A and become the process that is blocking the other 9 remaining processes. When you have lots of blocking that is not resolving in a reasonable amount of time you need to identify the root blocker, or the process at the top of the tree of blocked processes. Imagine again that you have 10 processes blocked by process B, and process B is blocked by process A. If A is not blocked by anything, but is itself responsible for lots of blocking (B and the 10 processes waiting on B), then A would be the root blocker. (Think of it as a traffic jam. Figure 2.8 will help) Killing A (via kill) is likely to unblock B, and once B completes, the 10 processes waiting on B are also likely to complete successfully.

The Oracle RAC Root Blockers test reports the number of root blocker processes in the clustered database. The detailed diagnosis of this test, provides the details of each of these blocker processes, thereby enabling you to identify the root blocker.

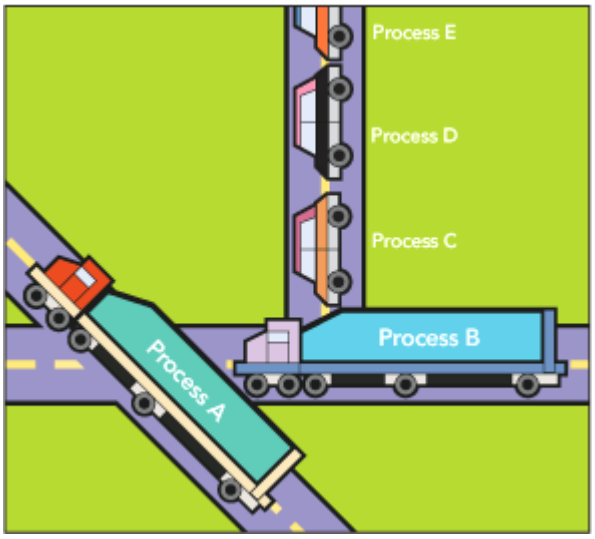


Figure 2.18: The traffic jam analogy representing blocking

Purpose	Monitors the number of blocker processes in a database
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">9. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. 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 enabled/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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Oracle cluster monitored.</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Root blockers:
Indicates the number of root blocker processes.</td><td>Number</td><td>If this value increases suddenly, this is a cause for concern. Likewise, if a process has been blocking other processes for a long time, it is a reason for further investigation. The detailed diagnosis for this test, if enabled, will indicate which process is blocking which other processes. Killing a blocker process that has been running for a long while may get the database running well again. Also, by carefully observing the details of the blocker processes, you can quickly identify the root blocker, and investigate the reason why it is blocking other processes.</td></tr>
</table>

10.3.8 Oracle RAC User Connections Test

This test reports the number and state of sessions of each user who is currently connected to the Oracle cluster. Using the metrics reported by this test, administrators can promptly isolate idle sessions, which are a drain on a cluster's resources.

Purpose	Reports the number and state of sessions of each user who is currently connected to the Oracle cluster
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">9. DD FREQUENCY - The DD FREQUENCY refers to the frequency with which detailed diagnosis measures are to be generated. 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. If you want the detailed diagnosis of this test to be generated at a different frequency, set a different DD FREQUENCY here. To disable the detailed diagnosis capability for a test, you can set this parameter to 0:0.
10. 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 enabled/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.
11. EXCLUDEUSER - In the EXCLUDEUSER text box, specify a comma-separated list of user names that need to be excluded from monitoring. By default, none is displayed here indicating that this test monitors connections initiated by all current users to the MS SQL server, by default.
12. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every user who is currently connected to the Oracle cluster</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total connections:
Indicates the total number of connections currently established by this user on the cluster.</td><td>Number</td><td></td></tr>
<tr><td>Active connections:
Indicates the number of connections of this user that are currently active.</td><td>Number</td><td>The detailed diagnosis of this measure, if enabled, will provide the complete details of the active sessions of a particular user. Using this information, you can understand how each of the connections were made - i.e., using which program - from where - i.e., from which host – and to which cluster node.</td></tr>
</table>

	Inactive connections: Indicates the number of sessions initiated by this user that are currently idle.	Number	Ideally, the value of this measure should be low. A high value is indicative of a large number of idle sessions, which in turn causes the unnecessary consumption of critical server resources. Idle sessions also unnecessarily lock connections from the connection pool, thereby denying other users access to the server for performing important tasks. The detailed diagnosis of this measure, if enabled, will provide the complete details of the idle sessions of a particular user. Using this information, you can understand how each of the idle connections were made - i.e., using which program - from where - i.e., from which host - and to which cluster node.
	Background connections: Indicates the number of background processes that were started when sessions are initiated by this user.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the background sessions of a particular user. Using this information, you can understand how each of the background connections were made - i.e., using which program - from where - i.e., from which host – and to which cluster node.

	Blocked connections: Indicates the number of sessions initiated by this user were blocked.	Number	Blocking occurs when one session holds a lock on a resource that another session is requesting. As a result, the requesting session will be blocked - it will hang until the holding session gives up the locked resource. In almost every case, blocking is avoidable. In fact, if you find that your session is blocked in an interactive application, then you have probably been suffering from the lost update bug as well, perhaps without realizing it. That is, your application logic is flawed and that is the cause of blocking. The five common DML statements that will block in the database are **INSERT**, **UPDATE**, **DELETE**, **MERGE** and **SELECT FOR UPDATE**. Ideally, the value of this measure should be low. A high value may cause unnecessary consumption of critical server resources thereby blocking access to potential active sessions. The detailed diagnosis of this measure, if enabled, will provide the complete details of the blocked sessions of a particular user. Using this information, you can understand how each of the blocked connections were made - i.e., using which program - and from where - i.e., from which host.
	Cached connections: Indicates the number of sessions of this user that were cached for future use.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the cached sessions of a particular user. Using this information, you can understand how each of the cached connections were made - i.e., using which program - from where - i.e., from which host – and to which cluster node.

	Killed connections: Indicates the number of sessions of this user that were terminated due to inactivity.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the killed sessions of a particular user. Using this information, you can understand how each of the killed connections were made - i.e., using which program - from where - i.e., from which host – and to which cluster node.
	Sniped connections: Indicates the number of sessions of this user that were idle for a period more than the profile's maximum idle time while waiting for a client's response.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the sniped sessions of a particular user. Using this information, you can understand how each of the sniped connections were made - i.e., using which program - from where - i.e., from which host – and to which cluster node.

10.3.9 Oracle RAC Cursor Usage Test

This test monitors the number of open cursors for every node of an Oracle cluster.

Purpose	Monitors the number of open cursors for a database instance
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running.

	Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every cluster node in the Oracle cluster monitored.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Current open cursors: The number of cursors currently opened by this node on the shared cluster database.	Number	Many open cursors can exist if any application does not properly close the ResultSets before closing a connection. Alternatively, many simultaneous queries to the database can also result in many open cursors. A continuous increase in open cursors is an indicator of a problem in an application's use of the database.
	Percent open cursors: This metric reports the average percentage of open cursors with respect to the total allowed limit.	Percent	If the percentage of open cursors nears 100%, then this could invoke the "maximum open cursors exceeded" error message. If the percentage is consistently near 100%, consider increasing the value of the 'open_cursors' parameter in the init file.

10.3.10 Oracle RAC Datafile Activity Test

This test indicates the level of read/write activity on each datafile in the shared cluster storage.

Purpose	Indicates the level of read/write activity on each datafile in every clustered database
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 1. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 6. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 7. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	8. **SHOW DATAFILE PATH** - This test reports a set of results for each datafile on the target Oracle database server. This means that every datafile is a descriptor of this test. By default, while displaying the descriptors of this test, the eG monitoring console does not prefix the datafile names with the full path to the datafiles. This is why, the **SHOW DATAFILE PATH** flag is set to **No** by default. If you want the data file names to be prefixed by the full path to the data files, then, set the **SHOW DATAFILE PATH** flag to **Yes**. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every datafile in the shared cluster storage monitored.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Physical block read rate: Indicates the rate at which disk blocks are being read from this datafile.	Blocks/Sec	A scenario in which more than 50% of blocks are being read from a single datafile could signify a problem.
	Physical block write rate: Indicates the rate at which disk blocks are being written to this datafile.	Blocks/Sec	A scenario in which more than 50% of blocks are being written to a single datafile could signify a problem. Too much activity to a specific datafile can result in reduced database performance. To improve performance, consider balancing I/O across disks, and reorganize tables across tablespaces to reduce activity to a specific datafile.
	Percent total I/O: Indicates the percentage of total I/O operations on the database server that were handled by this data file.	Percent	Disk reads and writes are expensive operations and all I/Os should be balanced across the different data files of an Oracle database for optimal performance. This metric reports the percentage of all I/O of an Oracle database that are happening on each of the data files of the Oracle database. This metric allows an Oracle administrator to determine which is/are the hot data file(s) (e.g., which data file is handling 80% of the total I/O).

10.3.11 Oracle RAC Data File Errors Test

The most common reasons for data file errors are corrupted blocks and invalid blocks. Both these can cause damage to portions of the database or the whole database, and can thus result in minimal to heavy loss of data. This is why, you should waste no time in identifying the error-prone data files and in doing all that is necessary to clear the errors and salvage the data. The **Oracle Data File Errors** test can play a key role in this exercise.

This test combs all the data files in the shared cluster storage for errors and reports the number of errors (if any). The test also provides the complete details of every error, thus enabling a speedy and effective resolution.

Purpose	Combs all the data files in the shared cluster storage for errors and reports the number of errors (if any)
Target of the test	An Oracle cluster

Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 2. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 6. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 7. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">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 enabled/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.

9. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Oracle cluster being monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Error files count:
Indicates the number of data file errors that have occurred.</td><td>Number</td><td>Ideally the value of this measure should be zero.

The detailed diagnosis of this measure indicates the File number, Status, Error, Recover and the Tablespace name for each error that has occurred in the datafiles.</td></tr>
</table>

10.3.12 Oracle RAC Database File Status Test

This test reports the status of each datafile in the shared cluster storage and the current access mode of every datafile.

Purpose	Reports the status of each datafile in the shared cluster storage and the current access mode of every datafile
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">9. SHOW DATAFILE PATH - This test reports a set of results for each datafile on the target Oracle database server. This means that every datafile is a descriptor of this test. By default, while displaying the descriptors of this test, the eG monitoring console does not prefix the datafile names with the full path to the datafiles. This is why, the SHOW DATAFILE PATH flag is set to No by default. If you want the data file names to be prefixed by the full path to the data files, then, set the SHOW DATAFILE PATH flag to Yes.
10. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every datafile in the shared storage of the Oracle cluster being monitored</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td></td><td>File status:
Indicates the current status of this datafile.</td><td></td><td>The table below indicates the values that this measure can report and their corresponding numeric equivalents:
<table><tr><th>Numeric Value</th><th>Measure Value</th></tr><tr><td>1</td><td>System</td></tr><tr><td>2</td><td>Online</td></tr><tr><td>3</td><td>Recover</td></tr><tr><td>4</td><td>Unknown</td></tr></table>
If a datafile is part of the SYSTEM tablespace, its status is SYSTEM (unless it requires recovery).
If a datafile in a non-SYSTEM tablespace is online, its status is ONLINE. If a datafile in a non-SYSTEM tablespace is offline, its status can be either OFFLINE or RECOVER.
Note:
By default, this measure reports the above-mentioned Measure Values while indicating the current status of a datafile. However, in the graph of this measure, data file states will be represented using the corresponding numeric equivalents only - i.e., 1 to 4.</td></tr>
</table>

	File access mode: Indicates the current access mode of this datafile.		The table below indicates the values that this measure can report and their corresponding numeric equivalents: **Numeric Value** / **Measure Value** 0 – Disabled 1 – Read Only 2 – Read Write 3 – Unknown **Note:** By default, this measure reports the above-mentioned **Measure Value**s while indicating the mode through which this datafile can be accessed. However, the graph of this measure will be represented using the corresponding numeric equivalents i.e., *0 to 3*.

10.3.13 Oracle RAC Database Growth Test

Periodic monitoring of the usage of the shared cluster storage is essential to ensure that the cluster is always adequately sized to handle current and future loads. The Oracle RAC Database Growth test monitors the usage of a shared storage, and indicates if it requires resizing.

Purpose	Monitors the usage of a shared storage, and indicates if it requires resizing
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 3. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 6. **PASSWORD** – Password of the specified database user This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components. 7. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

8. **ALTERNATE VIEW** – In large environments, where the volume of transactions to the Oracle database server is generally very high, this test may take time to execute and retrieve the desired results. To ensure that the test is faster and is resource-efficient, administrators of such environments can create an alternate 'view' on the target Oracle database server, and grant *select* privileges to the view to the special database **USER** mentioned above. Once the view is created, the test should be configured to use the alternate view for metrics collection; to achieve this, specify the name of the view in the **ALTERNATE VIEW** text box. By default, this text box is set to *none*, which implies that the alternate view is not used by default.

   This alternate 'view' should be created with the following structure:

```
CREATE OR REPLACE VIEW <VIEW_NAME> (
TABLESPACE_NAME,
FILE_ID,
BLOCK_ID,
BYTES,
BLOCKS,
RELATIVE_FNO
) AS
select /*+ use_hash (tsfi, fet2) */ tsfi.tablespace_name,
       tsfi.file_id,
       fet2.block_id,
       tsfi.blocksize * fet2.blocks,
       fet2.blocks,
       tsfi.relfile#

from   (select /*+ use_hash (ts, fi) */ ts.name tablespace_name,
               fi.file# file_id,
               ts.BLOCKSIZE,
               fi.relfile#,
               ts.ts#
        from   sys.ts$ ts,
               sys.file$ fi
        where  ts.ts# = fi.ts#
        and    ts.online$ in (1,4)) Tsfi,
       (select f.block# block_id,
               f.length blocks,
               f.file# file_id,
               f.ts#
        from   sys.fet$ f
        union all
        select f.ktfbfebno block_id,
               f.ktfbfeblks blocks,
               f.ktfbfefno,
               ktfbfetsn
        from   sys.x$ktfbfe f) Fet2
where  fet2.file_id = tsfi.relfile#
and    fet2.ts# = tsfi.ts# ;
```

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for the Oracle cluster monitored.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Allocated size of database: Indicates the amount of space currently allocated to the datafiles for use.	MB	
	Used space in allocated: Indicates the amount of allocated space currently used by the datafiles.	MB	
	Free space in allocated: Indicates the amount of allocated space that is still unused by the datafiles.	MB	
	Space usage in allocated: Indicates the percentage of allocated space that has been utilized by the datafiles.	Percent	If the **USE MAX SIZE** parameter of this test has been set to **No**, then the value of this measure will be computed using the following formula: *Used space/Total size of database * 100* If the **USE MAX SIZE** parameter of this test has been set to **Yes**, then the value of this measure will be computed using the following formula: *Used space/Maximum size upto which the database can grow * 100* Ideally, this value should be low. A value close to 100% is a cause for concern.
	Free percentage in allocated size: Indicates the percentage of allocated space that is unused.	Percent	
	Max size of database: Indicates the maximum size upto which the shared cluster storage can grow.	GB	
	Used space in max size: Indicates the amount of space used by datafiles.	GB	

	Free space in max size: Indicates the amount of space that is still unused by datafiles.	GB	
	Space usage in max size: Indicates the percentage of max space that is currently used by datafiles.	Percent	
	Free percentage in max size: Indicates the percentage of max space that is available for use.	Percent	
	Space free: Indicates the percentage of free space in this database instance.	Percent	If the **USE MAX SIZE** parameter of this test has been set to **No**, then the value of this measure will be computed using the following formula: *Free space/Total size of database * 100* If the **USE MAX SIZE** parameter of this test has been set to **Yes**, then the value of this measure will be computed using the following formula: *Free space/Maximum size upto which the database can grow * 100* Ideally, this value should be high. A sudden/consistent decrease in the value of this measure could indicate excessive utilization of the database caused by a sporadic/steady increase in database activity. Very low free space in a database instance could significantly deteriorate its performance. Under such circumstances therefore, you might want to check the measures reported by the Oracle Datafile GrowthTest to figure out which datafile is consuming too much space. You might then want to resize the datafile.

10.3.14 Oracle RAC DB Wait Time Test

Oracle's response time for an operation is composed of **time executing** (=CPU time) and **time spent waiting** (=Waiting time). An increase in either or both the above-mentioned factors will adversely impact the responsiveness of the Oracle cluster service.

When Oracle executes an SQL statement, it is not constantly executing. Sometimes it has to wait for a specific *event* to happen before it can proceed. For example, if Oracle (or the SQL statement) wants to modify data, and the corresponding database block is not currently in the SGA, Oracle waits for this block to be available for modification. The *Waiting time* refers to the time spent by the Oracle server waiting for such *events* to complete. Oracle has a bunch of events that it can wait for - eg., buffer busy waits, db file scattered read, db file sequential read.

Whenever users complaint of a slowdown while accessing databases in a cluster, it would be helpful to know which node is experiencing a slowdown and where it is spending too much time - is the time executing more than the time spent waiting, or vice-versa? To determine this, you should monitor both the *CPU time* and the *Waiting time* of each node of the cluster. This test enables you to perform 'half' this analysis. In other words, this test reports the percentage of time that every node spent on waiting for one/more events to complete. This way, the test helps you understand whether/not the *waiting time* is contributing to the poor responsiveness of the cluster service and which node has been waiting too long.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the percentage of time that every node in the cluster spent on waiting for one/more events to complete
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running.

	Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for each node in the Oracle cluster being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	DB time spent waiting: Indicates the percentage of time this node spent on waiting for one/more events to complete.	Percent	A high value is indicative of the following cases: • An increase in load (either more users, more calls, or larger transactions) • I/O performance degradation (I/O time increases and wait time increases, so DB time increases) • Application performance degradation • CPU-bound host (foregrounds accumulate active run-queue time, wait event times are artificially inflated)

10.3.15 Oracle RAC Defer Transactions Test

Oracle uses deferred transactions to propagate data-level changes asynchronously among master sites in an advanced replication system as well as from an updatable snapshot to its master table.

This test reports the number of deferred transactions in the shared cluster storage.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Reports the number of deferred transactions in the shared cluster storage
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for the Oracle cluster being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Defer transaction count: Indicates the number of deferred transactions in the shared cluster storage.	Number	When the advanced replication facility pushes a deferred transaction to a remote site, it uses a distributed transaction to ensure that the transaction has been properly committed at the remote site before the transaction is removed for the queue at the local site. If transactions are not being pushed to a given remote site, verify that the destination for the transaction was correctly specified. If you specify a destination database when calling *DBMS_DEFER_SYS.SCHEDULE_EXECUTION* using the *DBLINK* parameter, or *DBMS_DEFER_SYS.EXECUTE* using the *DESTINATION* parameter, make sure the full database link is provided.

10.3.16 Oracle RAC Index Fragmentation Test

Indexes are Oracle database objects that provide a fast, efficient method of retrieving data from database tables. The physical addresses of required rows can be retrieved from indexes much more efficiently than by reading the entire table. Effective indexing usually results in significant improvements to SQL performance.

Oracle's default index structure is B*-tree, which stands for "Balanced tree." It has a hierarchical tree structure. At the top is the header. This block contains pointers to the appropriate branch block for any given range of key values. The branch block points either to another branch block, if the index is big, or to an appropriate leaf block. Finally, the leaf block contains a list of key values and physical addresses (ROWIDs) of rows in the database. Theoretically, any row in a table, even a big one, could be retrieved in a maximum of three or four I/Os (input/output operations): one header block, one or two branch block(s), and one leaf block.

The advantages of indexing do not come without a cost. As database objects, indexes are created for tables only and they must be in sync with them: indexes must be updated by the database with every data manipulation language (q) operation—INSERT, DELETE, or UPDATE. Where there are a large number of tables with dynamic data, too many INSERTs, DELETEs, and UPDATEs on the tables can over time, fragment the index. When indexes are fragmented, queries take longer to pull out rows from tables, thereby significantly increasing disk I/O. This adversely impacts overall SQL performance.

The first step to resolving the performance threat posed by fragmented indexes is to identify which indexes are fragmented. The **Oracle RAC Index Fragmentation** test helps in this regard. This test scans a pre-configured index sample for high and very high levels of fragmentation, and reports the count of fragmented indexes. Using the detailed diagnosis capability of the test, you can also quickly drill down to the specific indexes that have been fragmented. You

can thus proceed to rebuild the fragmented indexes to reduce disk I/O.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Scans a pre-configured index sample for high and very high levels of fragmentation, and reports the count of fragmented indexes
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

9. **OBJECT NAME** - Specify a comma-separated list of tables, the indexes of which need to be checked for fragmentation. Every table name should be specified in the following format: *<DisplayName>:<schema_name>.<table_name>*, where *schema_name* refers to the name of the table owner, and *table_name* refers to the name of the table. The *DisplayName* in your specification will appear as the descriptor of this test. For instance, to monitor the indexes of the *alarm* and *history* tables owned by user *admin*, your specification would be: *AlarmMon1:admin.alarm,AlarmMon2:admin.history*. To monitor all tables in a schema, the specification would be of the following format: *<DisplayName>:<schema_name>.**. For example, to monitor all the tables in the *admin* schema, your specification would be: *AlarmMon:admin.**.

   You can also configure the **OBJECT NAME** to indicate what percentage of records in a table are to be considered by this test for running index fragmentation checks. To achieve this, your **OBJECT NAME** specification should be of the following format: *<DisplayName>:<schema_name>.<table_name>@<Percentage_of_records_in_the table>*. For instance, say that you want to configure this test to monitor the indexes that correspond to **20%** of the *alarm* table and **30%** of the *history* table. The **OBJECT NAME** specification in this case will be: *AlarmMon:admin.alarm@20,AlarmMon1:admin.history@30*. **It is recommended that you keep this 'percentage value' small, as higher values will make this test that much more resource-intensive**.

   Note:

   Make sure that you configure the **OBJECT NAME** parameter with only table names and not view names. This is because, indexes are available for tables alone and not views.

10. **QUERYTIMEOUT** - Specify the time period upto which a query has to wait to obtain the required result set from the database in the **QUERYTIMEOUT** text box. If the query is not successful or if the query waits for a time period exceeding the specified time limit, the test will automatically kill the query.

	11. **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 enabled/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. 12. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every *DisplayName* configured for the **OBJECT NAME** parameter of this test		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**

	Highly fragmented Oracle indexes: Indicates the number of highly fragmented indexes.		If 30% - 49% of an index is found to be fragmented, then such an index is counted as a highly fragmented index. Ideally, the value of this measure should be 0. A high value indicates high index fragmentation. Fragmentation is characterized by *splitting* and *spawning*. Splitting happens when an index node becomes full with keys and a new index node is created at the same level as a full node. This widens the B*-tree horizontally. Spawning is the process of adding a new level to an index. As a new index is populated, it begins life as a single-level index. As keys are added, a spawning takes place and the first-level node reconfigures itself to have pointers to lower-level nodes. Both these phenomenon are key performance degraders. This is why, a high value of this measure, if left unchecked, can cause disk I/O to mount, queries to run for long periods, and the overall performance of the database server to deteriorate. Use the detailed diagnosis of this measure to identify highly fragmented indexes, and proceed to rebuild them.

	Very highly fragmented Oracle indexes: Indicates the number of indexes that are very highly fragmented.	Number	If 50% or more of an index is found to be fragmented, then such an index is counted as a very highly fragmented index. Ideally, the value of this measure should be 0. A high value indicates very high index fragmentation. Fragmentation is characterized by *splitting* and *spawning*. Splitting happens when an index node becomes full with keys and a new index node is created at the same level as a full node. This widens the B*-tree horizontally. Spawning is the process of adding a new level to an index. As a new index is populated, it begins life as a single-level index. As keys are added, a spawning takes place and the first-level node reconfigures itself to have pointers to lower-level nodes. Both these phenomenon are key performance degraders. This is why, a high value of this measure, if left unchecked, can cause disk I/O to mount, queries to run for long periods, and erall performance of the database server to deteriorate. Use the detailed diagnosis of this measure to identify very highly fragmented indexes, and proceed to rebuild them.

10.3.17 Oracle RAC Jobs Test

This test monitors Oracle jobs and reports the number of jobs that have failed and those that are broken. The detailed diagnosis capability offered by this test enables administrators perform further diagnosis on failed/broken jobs, by additionally revealing the complete details of the failed and broken jobs.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Purpose	Monitors Oracle jobs and reports the number of jobs that have failed and those that are broken
Target of the test	An Oracle server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">9. 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 enabled/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.

10. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Oracle cluster</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td></td><td>Failed Oracle jobs:
Indicates the number of jobs that failed.</td><td>Number</td><td>Ideally, the value of this measure should be 0. Any value greater than zero, is a cause of concern, as it indicates the existence of a failed job. To know which job(s) has failed, use the detailed diagnosis capability of this measure.

Typically, if a job fails, Oracle attempts to run the job again 16 times, at fixed time intervals. You are advised to investigate the reason for the failure and fix it, by the time Oracle completes its 16th attempt. This is because, if the 16th attempt too fails, Oracle flags the job as a 'broken job', which can then be executed only manually.</td></tr>
</table>

	Broken Oracle jobs: Indicates the number of jobs broken.	Number	Ideally, the value of this measure should be 0. Any value greater than 0 is a problem, as it indicates the existence of one/more broken jobs. A job is considered broken, only if the 16th attempt made by Oracle to run the job fails. To know which jobs have broken, use the detailed diagnosis capability of this measure. Once the jobs are identified, you can proceed to manually run the broken jobs through the DBMS_JOB.RUN procedure after logging in as the owner of that job.

10.3.18 Oracle RAC Latches Test

Latches are mechanisms for protecting and managing SGA data structures and database objects being accessed concurrently. Unlike locks, latches provide exclusive access to protected data structures. Requests for latches are not queued. So, if a request fails, the requesting process may try later. Typically, latches are used to protect resources that are briefly needed.

An Oracle process can request a latch in one of the following two modes:

- **Willing-to-Wait Mode**: If the requested latch is not immediately available, the process will wait. When an attempt to get a latch in a willing-to-wait mode fails, the process will spin and try again. If the number of attempts reaches the value of the SPIN_COUNT parameter, the process sleeps. Sleeping is more expensive than spinning.
- **Immediate Mode (no-wait mode)**: In this case, the process will not wait if the requested latch is not available and it continues its processing.

Latch contention has a significant impact on performance when:

(i) Enough latches are not available

(ii) A latch is held for a relatively long time

Latch mechanisms most likely to suffer from contention involve requests to write data into the redo log buffer. To serve the intended purpose, writes to the redo log buffer must be serialized. There are four different groupings applicable to redo buffer latches: redo allocation latches and redo copy latches, each with immediate and willing-to-wait priorities.

The Oracle RAC Latches test is used to monitor latches in the shared cluster storage.

Purpose	Monitors the latches in the shared cluster storage
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">9. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every node in the Oracle cluster monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Willing-to-wait misses:
This measures the latch contention for requests that were willing to wait to acquire a latch. The value of this metric represents the ratio of the number of requests that could not acquire a latch, to those that could acquire a latch.</td><td>Percent</td><td rowspan="2">Both the above metrics should be 1% or less. For redo allocation latches, if the Willing_to_wait_misses is high, consider decreasing the LOG_SMALL_ENTRY_MAX_SIZE parameter in the INIT.ORA file. By making the max size for a redo allocation latch smaller, more redo log buffer writes qualify for a redo copy latch instead, thus better utilizing multiple CPU's for the redo log buffer writes. Even though memory structure manipulation times are measured in nanoseconds, a larger write still takes longer than a smaller write. If the size for remaining writes done via redo allocation latches is small enough, they can be completed with little or no redo allocation latch contention.
On a single CPU node, all log buffer writes are done via redo allocation latches. If log buffer latches are a significant bottleneck, performance can benefit from additional CPU's (thus enabling redo copy latches) even if the CPU utilization is not an operating system level bottleneck.
If the values for redo copy latches is > 1%, consider increasing the LOG_SIMULTANEOUS_COPIES parameter in the INIT.ORA file. This initialization parameter is the number of redo copy latches available. It defaults</td></tr>
<tr><td>Immediate misses:
This metric measures the latch contention for requests that were not willing to wait to acquire a latch. The value of this metric represents the percentage of "not willing to wait" latch requests that failed. In other words:
the number of "not willing to wait" request misses / the total number of "not willing to wait" requests</td><td>Percent</td></tr>
</table>

	Immediate misses: This metric measures the latch contention for requests that were not willing to wait to acquire a latch. The value of this metric represents the percentage of "not willing to wait" latch requests that failed. In other words: **the number of "not willing to wait" request misses / the total number of "not willing to wait" requests**	Percent	to the number of CPU's (assuming a multiple CPU node). Oracle recommends setting it as large as 2 times the number of CPU's on the particular node, although quite a bit of experimentation may be required to get the value adjusted in a suitable manner for any particular instance's workload. Depending on CPU capability and utilization, it may be beneficial to set this initialization parameter smaller or larger than 2 X #CPU's. **Note that the LOG_SIMULTANEOUS_COPIES parameter obsolete from Oracle 8i onwards. Hence, if you are monitoring Oracle 8i (or higher), use the hidden parameter _LOG_SIMULTAENOUS_COPIES instead.** Recall that the assignment of log buffer writes to either redo allocation latches or redo copy latches is controlled by the maximum log buffer write size allowed for a redo allocation latch, and is specified in the LOG_SMALL_ENTRY_MAX_SIZE initialization parameter. Recall also that redo copy latches apply only to multiple CPU hosts. **Note that the LOG_SMALL_ENTRY_MAX-SIZE parameter is not supported from Oracle 9i onwards.**

10.3.19 Oracle RAC Long Running Queries Test

This test tracks the currently executing queries on each node of an Oracle cluster and determines the number of queries that have been running for a long time and on which node.

Purpose	Tracks the currently executing queries on each node of an Oracle cluster and determines the number of queries that have been running for a long time and on which node.
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 10. **ELAPSED TIME** - In the **ELAPSED TIME** text box, specify the duration (in seconds) for which a query should have executed for it to be regarded as a long running query. The default value is 10. 11. **DISPLAYQUERY FULLTEXT** - The detailed diagnosis of this test lists the queries that have been running for a long time. In the **DETAILED DIAGNOSIS** page by default, query strings that are very long are truncated to display the first 1000 characters of the query alone. This is why, the **DISPLAYQUERY FULLTEXT** flag is set to **No** by default. To view the full query in the detailed diagnosis page, set this flag to **Yes**. **Note that setting this flag to 'Yes' may increase the size of your eG database**. 12. **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 enabled/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.
Outputs of the test	One set of results for every node in the cluster

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Long running queries: Indicates the number of queries currently executing on the this cluster node that have been running for more time than the configured **ELAPSED TIME**.	Number	The detailed diagnosis for this measure indicates the exact queries and which user is executing the queries. This information can be very useful in identifying queries that may be candidates for optimization.

10.3.20 Oracle RAC Redo Logs Test

Redo logs are applied during the roll forward phase of the recovery process. These logs hold information about the changes made to the database and whether they were committed. Each change is recorded in a redo record, which has information like the SCN of the change, changed data, commit flag, and information about which data block is changed. The Oracle RAC Redo Logs test monitors key performance metrics pertaining to the redo log buffer in each node of an Oracle cluster.

Purpose	Monitors the redo log buffer in each node of an Oracle cluster

Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running.

	Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every node in a cluster		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Redo buffer entries: This indicates the number of attempts to allocate space in the redo buffer of this node.	Number	A value other then 0 indicates that the redo writer is falling behind. This could be caused by log switches or checkpoints. By adjusting the LOG_CHECKPOINT_INTERVAL and LOG_CHECKPOINT_TIMEOUT parameters in the init.ora, you will be able to minimize the number of checkpoints. **From Oracle 9i onwards however, the LOG_CHECKPOINT_INTERVAL parameter is supported only for ensuring backward compatability with previous versions of Oracle. The recommended equivalent in case of Oracle 9i therefore is FAST_START_MTTR_TARGET.** You can also increase the number of LGWR writers. These parameters are new in Oracle 8 and are defined in the init.ora parameters LGWR_IO_SLAVES and ARCH_IO_SLAVES. However, **note that both these parameters are obsolete from Oracle 8i onwards.**
	Redo log space requests: The active log file is full and Oracle is waiting for disk space to be allocated for the redo log entries on this cluster node. Space is created by performing a log switch.	Number	Small Log files in relation to the size of the SGA or the commit rate of the work load can cause problems. When the log switch occurs, Oracle must ensure that all committed dirty buffers are written to disk before switching to a new log file. If you have a large SGA full of dirty buffers and small redo log files, a log switch must wait for DBWR to write dirty buffers to disk before continuing.
	Redo entries: This statistic increments each time redo entries are copied into the redo log buffer on this cluster node. (ie. The number of attempts to allocate space in the redo)	Number	

	Log space requests: This indicates the percentage of log space requests on this cluster node.	Percentage	If the number is greater than 1%, you should increase the size of the Redo Log buffer. I would also check the checkpoint and size of the online redo log file.
	Log space waits: This measure indicates the number of times wait has happened to acquire a log buffer on this node.	Number	If the Log Buffer space waits exist, consider increasing the size of the redo log. Also I would check the speed of the disk that the Online Redo Log files are in.

10.3.21 Oracle RAC SGA Test

The System Global Area (SGA) is the most important memory structure in Oracle. The SGA stores several different components of memory usage that are designed to execute processes to obtain data for user queries as quickly as possible while also maximizing the number of concurrent users that can access the Oracle instance. The main components of the SGA are

- **The buffer cache:** This area of memory allows for selective performance gains on obtaining and changing data. The buffer cache stores data blocks that contain row data that has been selected or updated recently. When the user wants to select data from a table, Oracle looks in the buffer cache to see if the data block that contains the row has already been loaded. If it has, then the buffer cache has achieved its selective performance improvement by not having to look for the data block on disk. If not, then Oracle must locate the data block that contains the row, load it into memory, and present the selected output to the user.
- **The shared pool:** The two main components of the shared pool are the shared SQL library cache and the data dictionary cache. The shared SQL library cache is designed to store parse information for SQL statements executing against the database. Parse information includes the set of database operations that the SQL execution mechanism will perform in order to obtain data requested by the user processes. This information is treated as a shared resource in the library cache. If another user process comes along wanting to run the same query that Oracle has already parsed for another user, the database will recognize the opportunity for reuse and let the user process utilize the parse information already available in the shared pool. The other component of the shared pool is the data dictionary cache, also referred to by many DBAs as the "row" cache. This memory structure is designed to store the data from the Oracle data dictionary in order to improve response time on data dictionary queries. Since all user processes and the Oracle database internal processes use the data dictionary, the database as a whole benefits in terms of performance from the presence of cached dictionary data in memory.

An Oracle database server brings in data into the SGA before doing any operation on it. So it is critical to monitor the various structures inside the SGA to ensure optimal database performance. The Oracle RAC SGA test collects a variety of statistics relating to the various SGA components on each node in an Oracle cluster.

Purpose	This test indicates the level of activity on the main components of System Global Area of every node in the cluster
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running.

	Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every node in the Oracle cluster		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Library cache hit ratio: The library cache is a buffer that contains the shared SQL and PL/SQL areas. The library cache hit ratio indicates the percentage of shared SQL statements being reparsed by this cluster node.	Percent	For a well-tuned database, this ratio is 90% or more. A lower hit ratio may indicate that the memory allocation to the library cache is insufficient. A low value can significantly degrade the database performance. Increasing the value of the SHARED_POOL_SIZE initialization parameter will help in improving the hit ratio.
	Data buffer cache hit ratio: Indicates the percentage of time that this cluster node is able to satisfy a request with information that is already available in the memory.	Percent	Physical I/O takes a significant amount of time, and also increases the CPU resources required. The database configuration should be tuned to ensure that a required block will most likely be in memory. The extent to which this is achieved is measured using the buffer cache hit ratio. For a well-tuned database, this ratio should be 80% or higher. A lower value indicates insufficient memory allocation to the database buffer cache. Increasing the value of the DB_BLOCK_BUFFERS initialization parameter will help in improving the hit ratio. If you are monitoring Oracle 9i or higher, then, **note that the DB_BLOCK_BUFFERS parameter is not supported in Oracle 9i or above. It is therefore recommended that you use the equivalent DB_CACHE_SIZE parameter instead.**
	Dictionary cache hit ratio: Indicates the percentage of data dictionary information pertaining to the database, file space availability and object privileges being readily available in this cluster node's memory.	Percent	As with the case of the library cache, the dictionary cache hit ratio should be at least 90%. A lower value may be due to the insufficient memory allocation to the dictionary cache. Increasing the value of the SHARED_POOL_SIZE parameter will help in improving the hit ratio.

	Redo log buffer misses: Indicates the percentage of requests to this cluster node that had to wait before the redolog buffer is allocated to it.	Percent	Before any transaction could occur, the before image of the data will be stored in the redo log buffer. It is crucial to make the redo log buffer available immediately to the transactions without any wait. The above is crucial to improve the overall performance. This measure indicates how many percentage of times it had to wait for a redo log buffer to be allocated. This can be improved by increasing the LOG_BUFFER parameter.
	Sorts on disk: Indicates the percentage of sorts that is happening on the secondary storage disk of this cluster node.	Percent	For best performance, most sorts should occur in memory; sorts written to disk adversely affect performance. If more than 10% of sorts happen on disk, the database performance could degrade. To improve the sorting performance of a database, consider tuning the parameters SORT_AREA_SIZE and SORT_AREA_RETAINED_SIZE. The dynamically modifiable initialization parameter called SORT_AREA_SIZE specifies the maximum amount of memory to use for each sort. If a significant number of sorts require disk I/O to temporary segments, an application's performance may benefit from increasing the size of the sort area. **Oracle 9i (or above) supports the SORT_AREA_SIZE and the SORT_AREA_RETAINED_SIZE parameters only to ensure backward compatibility with previous versions of Oracle. Therefore, while monitoring Oracle 9i or higher, it is recommended that you use the equivalent PGA_AGGREGATE_TARGET parameter instead.**

10.3.22 Oracle RAC Rollbacks Test

The immediate availability of rollback segments for the various activities that occur in a database server is very critical. Contention for rollback segments can adversely impact the performance of a database server and hence, needs to be detected and reported immediately. To detect contention for rollback segments on each node in an Oracle cluster, the Oracle RAC Rollbacks test monitors every cluster node for the degree of contention for buffers that contain rollback segment blocks.

Purpose	Monitors every cluster node for the degree of contention for buffers that contain rollback segment blocks
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here. 9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running.

	Measures will be reported as "Not applicable' by the agent if the server is not up.		
Outputs of the test	One set of results for every cluster node monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	System segment waits: Denotes the ratio of the number of waits for acquiring a header block or a block of the SYSTEM rollback segment to the total number of requests for data to this cluster node, measured over a period of time.	Percent	If the number of waits for any class of block exceeds 1% of the total number of requests, the size of the SYSTEM rollback segment needs to be increased.
	Non-system segment waits: Denotes the ratio of the number of waits for acquiring a header block or any other block of a non-SYSTEM rollback segment to the total number of requests for data to this cluster node, measured over a period of time.	Percent	If the number of waits for any class of block exceeds 1% of the total number of requests, the sizes of the existing rollback segments may need to be increased. Alternatively, additional rollback segments to may be created to reduce contention.

10.3.23 Oracle RAC SQL Network Test

Using the JDBC API, this test reports the availability and responsiveness of each node in the cluster, and collects statistics pertaining to the traffic into and out of every node.

Purpose	Reports the availability and responsiveness of each node in the cluster, and collects statistics pertaining to the traffic into and out of every node
Target of the test	An Oracle cluster
Agent deploying the test	An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the target Oracle server is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the target Oracle server is listening.

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USERNAME** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">9. INDIVIDUAL NODE – By default, this flag is set to Yes, indicating that this test will report metrics for every node in the cluster by default. You can set this flag to No to ensure that the test reports the availability and responsiveness of the cluster service as a whole, and not the individual cluster nodes.
10. 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.
11. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every node in the cluster</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Oracle cluster node availability:
Whether the cluster node is responding to requests.</td><td>Percent</td><td>The availability is 100% when a cluster node is responding to a request and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the node, or because the node is using an invalid user account. Besides the above, this measure will report that the server is unavailable even if a connection to the node is unavailable, or if a query to the node fails. In this case, you can check the values of the DB connection availability and Query processor availability measures to know what is exactly causing the node to not respond to requests - is it owing to a connection unavailability? or is it due to a query failure?</td></tr>
<tr><td>Total response time:
The time taken by this node to respond to a user query. This is the sum total of the connection time and query execution time.</td><td>Secs</td><td>A sudden increase in response time is indicative of a bottleneck at the node. This could even be owing to a connection delay and/or long running queries to the node. Whenever the value of this measure is high, it would be good practice to compare the values of the Connection time and Query execution time measures for a node to zero-in on the root-cause of the poor responsiveness of the server - is it because of connectivity issues? or is it because of inefficient queries?</td></tr>
</table>

	Data transmit rate: The rate of data being transmitted by this node in response to client requests.	KB/Sec	The data transmission rate reflects the workload on the server.
	Data receive rate: The rate of data received by this node from clients over SQL*Net.	KB/Sec	This measure also characterizes the workload on a node. As the data rate to a node increases, consider tuning the Service Layer Data Buffer (SDU) and the Transport Layer Data Buffer (BDU) in the `TNSNames.ora` and `Listener.ora` files to optimize packet transfers across the network.
	Cluster node connection availability: Indicates whether the database connection to this node is available or not.	Percent	If this measure reports the value 100, it indicates that the database connection is available. The value 0 on the other hand indicates that the database connection is unavailable. A connection to the database may be unavailable if the database is down or if the database is listening on a port other than the one configured for it in the eG manager or owing to a poor network link. If the *Oracle server availability* measure reports the value 0, then, you can check the value of this measure to determine whether/not it is due to the unavailability of a connection to the server.
	Connection time to cluster node: Indicates the time taken to connect to the cluster node.	Secs	A high value could indicate a connection bottleneck. Whenever the *Total response time* of the measure soars, you may want to check the value of this measure to determine whether a connection latency is causing the poor responsiveness of the node.
	Query processor availability: Indicates whether the query to this node is executed successfully or not.	Percent	If this measure reports the value 100, it indicates that the query executed successfully. The value 0 on the other hand indicates that the query failed. In the event that the *Oracle server availability* measure reports the value *0*, check the value of this measure to figure out whether the failed query is the reason why that measure reported a node unavailability.
	Query execution time: Indicates the time taken for query execution.	Secs	A high value could indicate that one/more queries to the node are taking too long to execute. Inefficient/badly designed queries to the database often take too long to execute. If the value of this measure is higher than that of the *Connection time* measure, you can be rest assured that long running queries are causing the node to respond slowly to requests.

10.3.24 Oracle RAC User Waits Test

When Oracle executes an SQL statement, it is not constantly executing. Sometimes it has to wait for a specific event to happen before it can proceed. For example, if Oracle (or the SQL statement) wants to modify data, and the corresponding database block is not currently in the SGA, Oracle waits for this block to be available for modification. Every such wait event belongs to a class of wait events. The following list describes each of the wait classes.

Wait Class	Description
Administrative	Waits resulting from DBA commands that cause users to wait (for example, an index rebuild)
Application	Waits resulting from user application code (for example, lock waits caused by row level locking or explicit lock commands)
Cluster	Waits related to Real Application Cluster resources (for example, global cache resources such as 'gc cr block busy'
Commit	This wait class only comprises one wait event - wait for redo log write confirmation after a commit (that is, 'log file sync')
Concurrency	Waits for internal database resources (for example, latches)
Configuration	Waits caused by inadequate configuration of database or instance resources (for example, undersized log file sizes, shared pool size)
Idle	Waits that signify the session is inactive, waiting for work (for example, 'SQL*Net message from client')
Network	Waits related to network messaging (for example, 'SQL*Net more data to dblink')
Other	Waits which should not typically occur on a system (for example, 'wait for EMON to spawn')
Scheduler	Resource Manager related waits (for example, 'resmgr: become active')
System I/O	Waits for background process IO (for example, DBWR wait for 'db file parallel write')
User I/O	Waits for user IO (for example 'db file sequential read')

Since wait events are resource-drains and serious performance degraders, administrators need to keep a close eye on these wait classes, figure out how much time the Oracle cluster actually spends waiting for each class, and rapidly decipher why, so that measures can be initiated to minimize these events. To achieve this, you can use the **Oracle RAC User Waits** test. This test reports the time spent by the nodes in the cluster waiting for events of each wait class, helps identify those wait classes with wait events that have remained active for a long time, and also reveals the number of sessions that have been impacted by the waiting. With the help of the detailed diagnostics of this test, you can also zoom into these sessions and identify the queries that they executed that may have caused wait events to occur; this way, inefficient queries can be isolated.

Purpose	Reports the time spent by the nodes in the cluster waiting for events of each wait class, helps identify those wait classes with wait events that have remained active for a long time, and also reveals the number of sessions that have been impacted by the waiting
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">9. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not unning. Measures will be reported as "Not applicable' by the agent if the server is not up.

10. 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 enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:
o The eG manager license should allow the detailed diagnosis capability
o Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each wait class active on the cluster nodes of the Oracle cluster being monitored</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Active sessions:
Indicates the current number of sessions in which events of this wait class are currently active.</td><td>Number</td><td>A high value indicates that too many sessions are waiting owing to the events of a particular wait class. To know more about these sessions, the wait events that each session triggered, and which query triggered the events, use the detailed diagnosis of this measure. With the help of the detailed metrics, you can quickly isolate the queries that require optimization.</td></tr>
</table>

	Max wait time: Indicates the maximum time for which the Oracle server has waited for events of this wait class.	Secs	A high value is indicative of the following: • An increase in load (either more users, more calls, or larger transactions) • I/O performance degradation (I/O time increases and wait time increases, so DB time increases) • Application performance degradation • CPU-bound host (foregrounds accumulate active run-queue time, wait event times are artificially inflated) Compare the value of this measure across wait classes to identify which wait class has caused the Oracle database server to wait for the maximum time. You can then use the detailed diagnostics reported by the *Active sessions* measure to identify which sessions were impacted, and what queries were executed by those sessions to increase wait time. Inefficient queries can thus be identified and optimized to ensure that waiting is eliminated or at least minimized.

10.3.25 Oracle RAC SQL Workload Test

Nothing can degrade the performance of an Oracle cluster like a resource-hungry or a long-running query! When such queries execute on the cluster, they either hog almost all the available CPU, memory, and disk resources of the cluster or keep the resources locked for long time periods, thus leaving little to no resources for carrying out other critical cluster operations. This can significantly slowdown the cluster and adversely impact user experience with the cluster. To ensure peak performance of the Oracle cluster at all times, such queries should be rapidly identified and quickly optimized to minimize resource usage. This is where the **Oracle SQL Workload** test helps. At configured intervals, this test compares the usage levels and execution times of all queries that started running on the cluster in the last measurement period and identifies a 'top query' in each of the following categories - CPU usage, memory usage, disk activity, and execution time. The test then reports the resource usage and execution time of the top queries and promptly alerts administrators if any query consumes more resources or takes more time to execute than it should. In such a scenario, administrators can use the detailed diagnosis of this test to view the inefficient queries and proceed to optimize them to enhance cluster performance.

Purpose	At configured intervals, this test compares the usage levels and execution times of all queries that started running on the cluster in the last measurement period and identifies a 'top query' in each of the following categories - CPU usage, memory usage, disk activity, and execution time. The test then reports the resource usage and execution time of the top queries and promptly alerts administrators if any query consumes more resources or takes more time to execute than it should. In such a scenario, administrators can use the detailed diagnosis of this test to view the inefficient queries and proceed to optimize them to enhance cluster performance.
Target of the test	An Oracle cluster
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

<table>
<tr><td></td><td colspan="3">9. DDCOUNT – By default, the detailed diagnosis of this test reports the top-5 queries in resource usage and execution time. This is why, the DDCOUNT parameter is set to 5 by default. If you want detailed diagnosis to display less or more number of top queries, then change the DDCOUNT.
10. ISPASSIVE – If the value chosen is YES, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
11. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the Oracle cluster monitored.</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Top most query physical reads:
Indicates the number of physical disk reads performed by the top query per execution.</td><td>Reads/execution</td><td>If the value of this measure is abnormally high, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries generating maximum physical disk activity. From this, you can identify the top query in terms of number of physical disk reads. You may then want to optimize the query to reduce the disk reads.</td></tr>
<tr><td>Top most buffer gets:
Indicates the number of memory buffers used by the top query per execution.</td><td>Memorybuffer gets/execution</td><td>If the value of this measure is abnormally high, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries consuming memory excessively. From this, you can easily pick that query which is consuming the maximum memory. You may then want to optimize the query to minimize memory usage.</td></tr>
<tr><td>Top most query CPU time:
Indicates the duration for which each execution of the top query was hogging the CPU resources.</td><td>Secs/execution</td><td>If the value of this measure is over 30 seconds, you can use the detailed diagnosis of this measure to the top-5 (by default) queries hogging the CPU resources. From this, you can easily pick that query which is consuming the maximum CPU. You may then want to optimize the query to minimize CPU usage.</td></tr>
</table>

	Top most query elapsed time: Indicates the running time of each execution of the top query.	Secs/execution	If the value of this measure crosses 10 seconds, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries that are taking too long to execute. . From this, you can easily pick that query with the maximum execution time. You may then want to optimize the query to minimize execution time.

10.3.26 Oracle RAC Uptime Test

In most production environments, it is essential to monitor the uptime of critical servers in the infrastructure. By tracking the uptime of each of the servers, administrators can determine what percentage of time a server has been up. Comparing this value with service level targets, administrators can determine the most trouble-prone areas of the infrastructure.

In some environments, administrators may schedule periodic reboots of their servers. By knowing that a specific server has been up for an unusually long time, an administrator may come to know that the scheduled reboot task is not working on a server.

The Oracle RAC Uptime Test monitors the uptime of every node in an Oracle cluster.

Purpose	To monitor the uptime of every node in an Oracle cluster
Target of the test	A Windows or Unix server
Agent deploying the test	An internal agent

Configurable parameters for 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 on which the server is listening 4. **ORASID** - The variable name of the oracle instance 5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster. To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster: *select name, value from v$parameter where name ='service_names'* 6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges. The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is: *create user oraeg identified by oraeg* *create role oratest;* *grant create session to oratest;* *grant select_catalog_role to oratest;* *grant oratest to oraeg;* The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is: *alter session set container=**<Oracle_service_name>**;* *create user **<user_name>** identified by **<user_password>** container=current default tablespace **<name_of_default_tablespace>** temporary tablespace **<name_of_temporary_tablespace>**;* *Grant create session to <user_name>;* *Grant select_catalog_role to **<user_name>**;* The name of this user has to be specified here. 7. **PASSWORD** – Password of the specified database user 8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

	9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 10. **REPORTMANAGERTIME** – By default, this flag is set to **Yes**, indicating that, by default, the detailed diagnosis of this test, if enabled, will report the shutdown and reboot times of the device in the manager's time zone. If this flag is set to **No**, then the shutdown and reboot times are shown in the time zone of the system where the agent is running (i.e., the system being managed for agent-based monitoring, and the system on which the remote agent is running - for agentless monitoring).		
Outputs of the test	One set of results for every node in the Oracle cluster being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Has Oracle server been restarted?: Indicates whether this node has been rebooted during the last measurement period or not.	Boolean	If this measure shows 1, it means that the node was rebooted during the last measurement period. By checking the time periods when this metric changes from 0 to 1, an administrator can determine the times when this node was rebooted. The detailed diagnosis of this measure, if enabled, indicates the date/time at which the node was shutdown, the date on which it was restarted, the duration of the shutdown, and whether the node was shutdown as part of a maintenance outine.
	Uptime since the last measurement: Indicates the time period that this node has been up since the last time this test ran.	Secs	If the node has not been rebooted during the last measurement period and the agent has been running continuously, this value will be equal to the measurement period. If the node was rebooted during the last measurement period, this value will be less than the measurement period of the test. For example, if the measurement period is 300 secs, and if the node was rebooted 120 secs back, this metric will report a value of 120 seconds. The accuracy of this metric is dependent on the measurement period – the smaller the measurement period, greater the accuracy.

	Uptime: Indicates the total time that the node has been up since its last reboot.	Mins	Administrators may wish to be alerted if a node has been running without a reboot for a very long period. Setting a threshold for this metric allows administrators to determine such conditions.

11

Monitoring the Microsoft SQL Cluster Server

When two or more MS SQL servers exist in an environment, they can be grouped together to form a SQL cluster. Requests to a cluster are routed through a virtual cluster server that is assigned a cluster IP address and TCP port. Requests to this server can be handled by any of the individual nodes in the cluster at any given point in time, depending on which node is active at that time.

Since clusters are deployed in environments where 24*7 availability and responsiveness are critical, it is imperative that the performance of the clusters is monitored all the time.

To monitor an MS SQL server cluster, the eG agent (internal/remote) connects to the virtual cluster IP address and port, periodically checks cluster availability, responsiveness and uptime, and also promptly alerts on a fail-over when it occurs. If the cluster service is up and running, the same agent also collects from the cluster, all performance statistics that are typically collected from a stand-alone SQL server – this includes, metrics on the size and usage of databases, the health of the database engine, locking and blocking activity, queries/transactions to the database, SQL errors and wait events, and many more.

Figure 11.1 depicts the *Microsoft SQL Cluster Server* monitoring model.

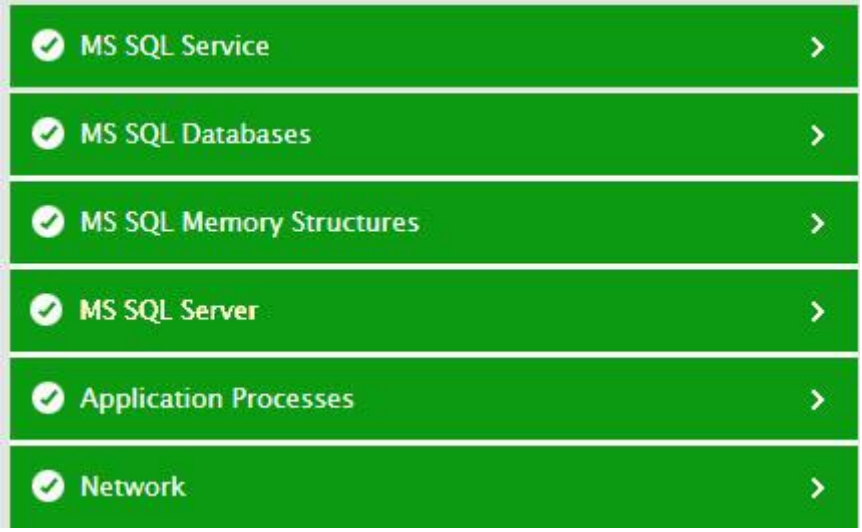


Figure 11.1: The layer model of a SQL cluster service

This section will be discussing the top 3 layers of Figure 11.1 only, as the **Network** layer has already been extensively discussed in the *Monitoring Unix and Windows Servers* document.

11.1 The Application Processes Layer

The tests mapped to this layer report on the availability of the cluster server port and the cluster process.

Figure 11. 2: The tests mapped to the Application Processes layer

11.1.1 SQL Cluster Process Test

This test reports the current state and resource usage of the cluster process, and promptly alerts administrators if the cluster goes down or is up and consuming more resources than it should.

Purpose	Reports the current state and resource usage of the cluster process, and promptly alerts administrators if the cluster goes down or is up and consuming more resources than it should.		
Target of the test	A SQL Cluster		
Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the SQL cluster 3. **PORT** – The port on which the cluster is listening 4. **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.		
Outputs of the test	One set of results for the cluster being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**

	Service availability: Indicates the availability of the cluster service.	Percent	The availability is 100% when the cluster process is running. This measure will report the value 0, if the cluster process is not running.
	Processes running: Indicates the number of instances of the cluster process that are currently running.	Number	The value 0 for this measure is indicative of the non-availability of the cluster service. The detailed diagnosis of this measure reveals the process ID of each running instance of the cluster process and the resource and I/O usage of each instance, so that you can quickly identify which instance is resource-intensive.
	CPU utilization: Indicates the percentage of CPU resources used by the cluster process.	Percent	A value close to 100% indicates that the cluster process is hogging the CPU resources of the 'active' node in the cluster.
	Handle count: Indicates the number of handles opened by the process.	Number	An increasing trend in this measure is indicative of a memory leak in the process.
	Number of threads: Indicates the number of threads that are used by the process.	Number	
	Virtual memory used: Indicates the amount of virtual memory that is being used by the process.	MB	
	Private memory used: Indicates the amount of memory that this process has been allotted, that cannot be shared with other processes.	MB	
	Pool paged: Indicates the amount of memory currently in the pool paged area of system memory.	MB	
	Pool non paged: Indicates the amount of memory currently in the pool non-paged area of system memory.	MB	

	I/O data rate: Indicates the rate at which processes are reading and writing bytes in I/O operations.	Kbytes/Sec	This value counts all I/O activity generated by each process and includes file, network and device I/Os.
	I/O data operations: Indicates the rate at which the process is issuing read and write data to file, network and device I/O operations.	Operations/Sec	
	I/O read data rate: Indicates the rate at which the process is reading data from file, network and device I/O operations.	Kbytes/Sec	
	I/O write data rate: Indicates the rate at which the process is writing data to file, network and device I/O operations.	Kbytes/Sec	
	Page fault rate: Indicates the total rate at which page faults are occurring for the threads of all matching processes.	Faults/Sec	A page fault occurs when a thread refers to a virtual memory page that is not in its working set in main memory. This may not cause the page to be fetched from disk if it is on the standby list and hence already in main memory, or if it is in use by another process with whom the page is shared.
	Memory working set: Indicates the current size of the working set of a process.	MB	The Working Set is the set of memory pages touched recently by the threads in the process. If free memory in the computer is above a threshold, pages are left in the Working Set of a process even if they are not in use. When free memory falls below a threshold, pages are trimmed from Working Sets. If they are needed they will then be soft-faulted back into the Working Set before leaving main memory. The detailed diagnosis for this test provides details of the individual process instances and their individual working sets. Comparing the working set across process instances indicates which instances taking up excessive memory.

11.2 The MS SQL Server Layer

The tests associated with this layer monitor the health of the SQL server engine and the number and type of system processes executing on the MS SQL server. In addition, the test also reports the count of blocker processes executing on the MS SQL server. As these tests have already been discussed in Chapter 3 of this document, this section will not once again indulge in a detailed discussion of the same.

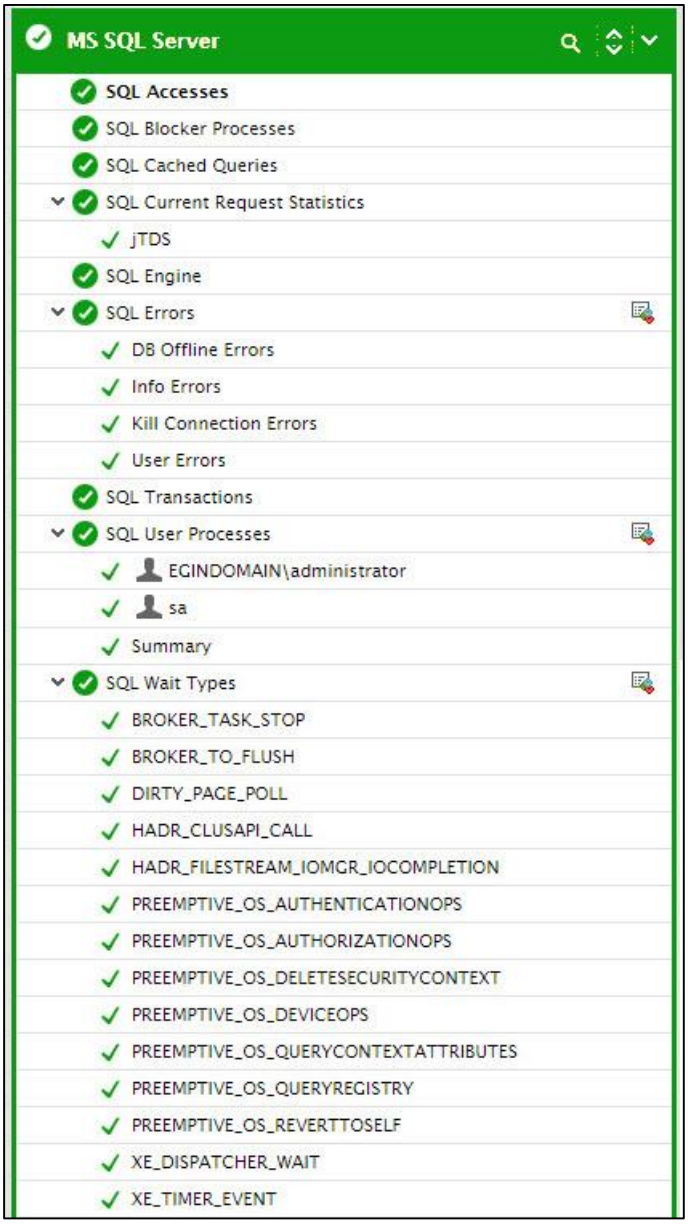


Figure 11.3: The MS SQL Server layer

11.3 The MS SQL Memory Structures Layer

This layer tracks the health of the memory and buffer structures of an MS SQL server. The details of the tests are available in Chapter 3 of this document.

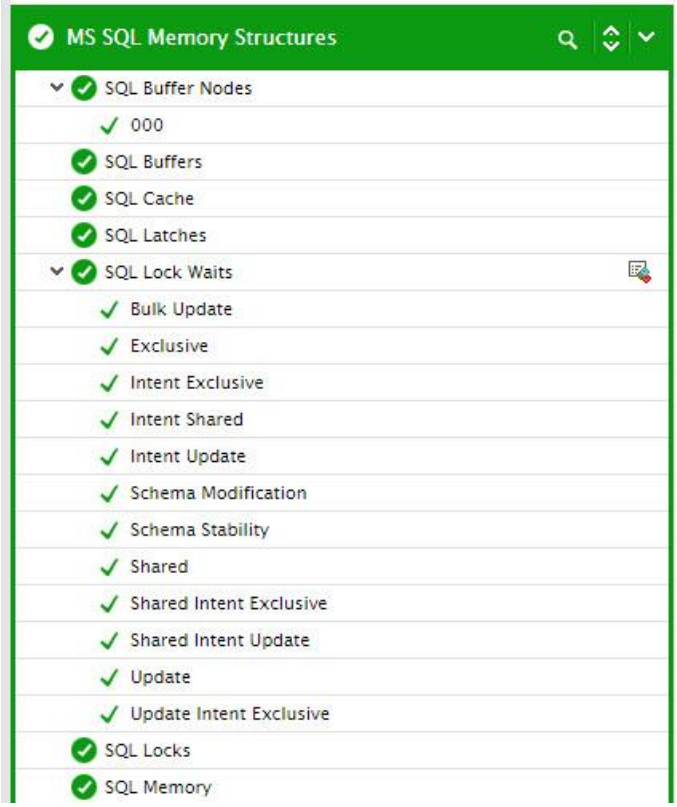


Figure 11. 4: The tests mapped to the MS SQL Memory Structures layer

11.4 The MS SQL Databases Layer

The space usage on the MS SQL server databases and the transaction log space usage can be tracked using the tests associated with this layer. These tests have also been discussed in Chapter 3 of this document.

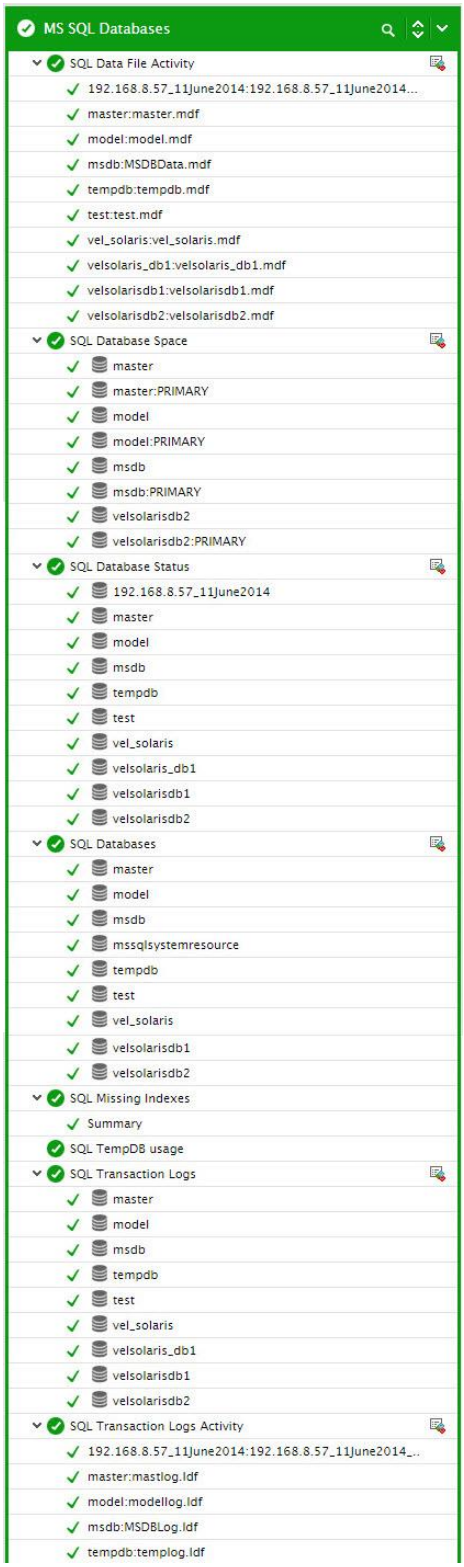


Figure 11.5: The tests mapped to the MS SQL Databases layer

11.5 The MS SQL Service Layer

Figure 11.6 depicts the tests associated with this layer.

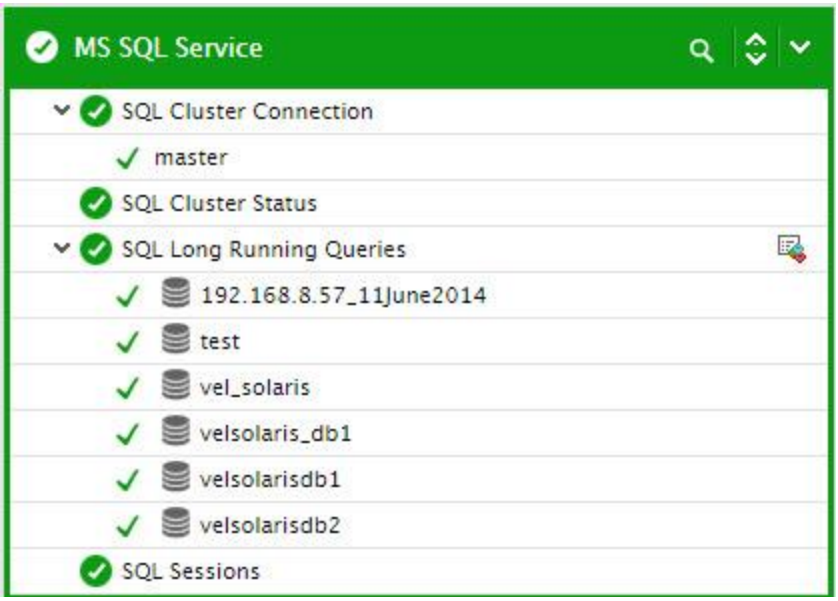


Figure 11.6: The tests associated with the MS SQL Service layer

The SQL Sessions and SQL Long Running Queries test has already been discussed in Chapter 3 of this document. Therefore, the sub-sections that follow will elaborate on the cluster-specific tests mapped to this layer only.

11.5.1 SQL Cluster Connection Test

This test emulates a user executing a query on the cluster, and in the process, captures the availability of the cluster service and the responsiveness of the cluster.

Purpose	Emulates a user executing a query on the cluster, and in the process, captures the availability of the cluster service and the responsiveness of the cluster.
Target of the test	A SQL Cluster
Agent deploying the test	An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the virtual cluster server is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the virtual cluster server is listening.

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the SQL cluster
3. PORT – The port on which the cluster is listening
4. USER – A database user name.
5. PASSWORD- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password.
6. CONFIRM PASSWORD – Confirm the PASSWORD (if any) by retyping it here.
7. DATABASE - The name of the database to connect to. The default is "master".
8. QUERY – The select query to execute. The default is "select * from master.dbo.spt_monitor".
9. CASE – Takes the value "upper" or "lower" depending upon the case-sensitivity of the SQL server installation.
10. INSTANCE – The name of a specific MS SQL instance to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the "instance" parameter.
11. DOMAIN - By default, none is displayed in the DOMAIN text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the DOMAIN can continue to be none. On the other hand, if 'Windows only' authentication has been enabled, then, in the DOMAIN text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the USER name and PASSWORD that you provide should be that of a user authorized to access the monitored SQL server.
12. SSL - If the MS SQL server being monitored is an SSL-enabled server, then set the SSL flag to Yes. If not, then set the SSL flag to No.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the cluster being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>SQL availability:
Indicates the availability of the cluster service</td><td>Percent</td><td>The availability is 100% when the cluster is able to respond to a request. This can happen if any one server in the cluster is currently 'active' and is responding to client requests.
This measure will report the value 0, if the cluster service is not up and running. Such an eventuality can be caused by the non-availability of active nodes in the cluster to handle the emulated query.</td></tr>
<tr><td>SQL response time:
Indicates the time taken by the cluster to respond to a user query</td><td>Seconds</td><td>A sudden increase in response time is indicative of a bottleneck in query processing on the 'active' server of the cluster.</td></tr>
</table>

	Database connection availability: Indicates whether the database connection is available or not.	Percent	If this measure reports the value 100 , it indicates that the database connection is available. The value 0 on the other hand indicates that the database connection is unavailable. A connection to the database may be unavailable if the database is down or if the database is listening on a port other than the one configured for it in the eG manager or owing to a poor network link. If the *SQL availability* measure reports the value 0, then, you can check the value of this measure to determine whether/not it is due to the unavailability of a connection to the server.
	Query processor availability: Indicates whether the database query is executed successfully or not.	Percent	If this measure reports the value 100, it indicates that the query executed successfully. The value 0 on the other hand indicates that the query failed. In the event that the *SQL availability* measure reports the value *0*, check the value of this measure to figure out whether the failed query is the reason why that measure reported a server unavailability.
	Database connection time: Indicates the time taken by the database connection.	Secs	A high value could indicate a connection bottleneck. Whenever the *SQL response time* of the measure soars, you may want to check the value of this measure to determine whether a connection latency is causing the poor responsiveness of the cluster.
	Query execution time: Indicates the time taken for query execution.	Secs	A high value could indicate that one/more queries to the cluster are taking too long to execute. Inefficient/badly designed queries often run for long periods. If the value of this measure is higher than that of the *Connection time* measure, you can be rest assured that long running queries are the ones causing the responsiveness of the cluster to suffer.
	Records fetched: Indicates the number of records fetched from the database.	Number	The value 0 indicates that no records are fetched from the database.

11.5.2 SQL Cluster Status Test

This test reports the current status (whether running or not) and uptime of the cluster service. This way, administrators can quickly find out if the cluster service was restarted recently. In addition, the test also indicates whether/not fail-over occurred recently. The IP address of the 'active' server in the cluster is also revealed as part of detailed diagnosis.

This test will report metrics only on Microsoft SQL Server 2008 (and above).

Purpose	Reports the current status (whether running or not) and uptime of the cluster service. This way, administrators can quickly find out if the cluster service was restarted recently. In addition, the test also indicates whether/not fail-over occurred recently. The IP address of the 'active' server in the cluster is also revealed as part of detailed diagnosis.
Target of the test	A SQL Cluster
Agent deploying the test	An internal/remote agent

Configurable parameters for 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. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**. 4. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter. 5. **USER** – Provide the name of a SQL user with the **VIEW SERVER STATE** role. 6. **PASSWORD** - The password of the specified **USER** 7. **CONFIRM PASSWORD** - Confirm the password by retyping it 8. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server. 9. **ISNTLMV2** - In some Windows networks, **NTLM** (**NT LAN Manager**) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host. 10. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up. 11. **REPORTMANAGERTIME** – By default, this flag is set to **Yes**, indicating that, by default, the detailed diagnosis of this test, if enabled, will report the shutdown and reboot times of the device in the manager's time zone. If this flag is set to **No**, then the shutdown and reboot times are shown in the time zone of the system where the agent is running (i.e., the system being managed for agent-based monitoring, and the system on which the remote agent is running - for agentless monitoring). 12. **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.
Outputs of the test	One set of results for the cluster being monitored

<table>
<tr><th>Measurements made by the test</th><th>Measurement</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td rowspan="3"></td><td>Is cluster running ?:
Indicates whether/not the cluster is currently running.</td><td></td><td>The values that this measure can report and their corresponding numeric values are listed in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Yes</td><td>0</td></tr><tr><td>No</td><td>1</td></tr></table>
Note:
By default, the test reports the Measure Values in the table above to indicate whether/not the cluster is running currently. In the graph of this measure however, the same is represented using the numeric equivalents only.
The detailed diagnosis of the Is cluster running? measure reveals the IP address of the currently ‘active’ node in the cluster and the date/time at which the active node was last started.</td></tr>
<tr><td>Has SQL cluster switched?:
Indicates whether/not fail-over occurred in the last measurement period.</td><td></td><td>The values that this measure can report and their corresponding numeric values are listed in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>0</td></tr></table>
Note:
By default, the test reports the Measure Values in the table above to indicate whether/not fail-over occurred. In the graph of this measure however, the same is represented using the numeric equivalents only.</td></tr>
<tr><td>Uptime of the SQL cluster:
Indicates the total time the SQL cluster has been up since its last reboot.</td><td>Secs</td><td>Administrators may wish to be alerted if a cluster has been running without a reboot for a very long period. Setting a threshold for this metric allows administrators to determine such conditions.</td></tr>
</table>

<table>
<tr>
<td></td>
<td>SQL cluster uptime since last measure:
Indicates how long the SQL cluster has been up since the last measurement period.</td>
<td>Secs</td>
<td>If the cluster has not been rebooted during the last measurement period and the agent has been running continuously, this value will be equal to the measurement period. If the cluster was rebooted during the last measurement period, this value will be less than the measurement period of the test. For example, if the measurement period is 300 secs, and if the cluster was rebooted 120 secs back, this metric will report a value of 120 seconds. The accuracy of this metric is dependent on the measurement period – the smaller the measurement period, greater the accuracy.</td>
</tr>
<tr>
<td></td>
<td>Has the SQL cluster been restarted?:
Indicates whether the SQL cluster server has been rebooted during the last measurement period or not.</td>
<td></td>
<td>The values that this measure can report and their corresponding numeric values are listed in the table below:
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>Yes</td><td>1</td></tr>
<tr><td>No</td><td>0</td></tr>
</table>
Note:
By default, the test reports the Measure Values in the table above to indicate whether/not the cluster was restarted in the last measure period. In the graph of this measure however, the same is represented using the numeric equivalents only.</td>
</tr>
</table>

The detailed diagnosis of the *Is cluster running?* measure reveals the IP address of the currently 'active' node in the cluster and the date/time at which the active node was last started.

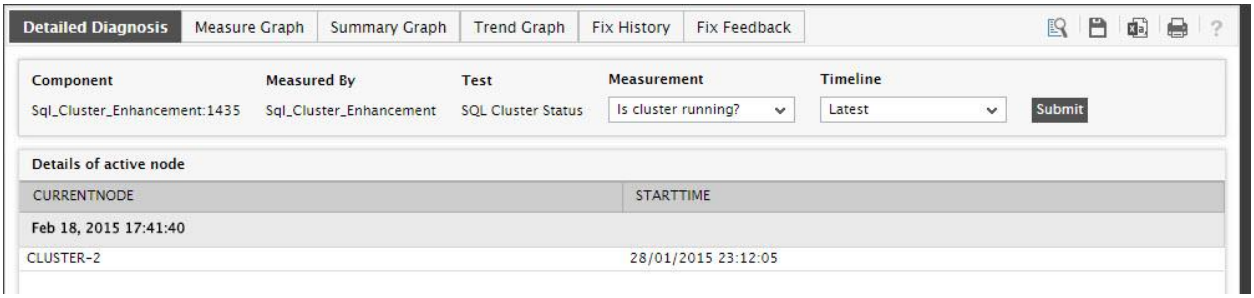


Figure 11.7: The detailed diagnosis of the Is cluster running? measure of the SQL Cluster Status test

12

Monitoring Backup SQL Servers

Backup SQL servers are those SQL servers in a cluster that serve as backups to a primary SQL server during its downtime. Just like the primary SQL server, it is also necessary to monitor the Backup SQL server, because if both the primary and the backup server are unavailable for use at the same time, then this would bring down the cluster as well as the business service dependent on it.

eG Enterprise provides a specialized *Backup SQL* monitoring model (see Figure 12.1) that periodically checks if the Backup SQL server is available, and in the process, reveals whether critical SQL health parameters are stable.

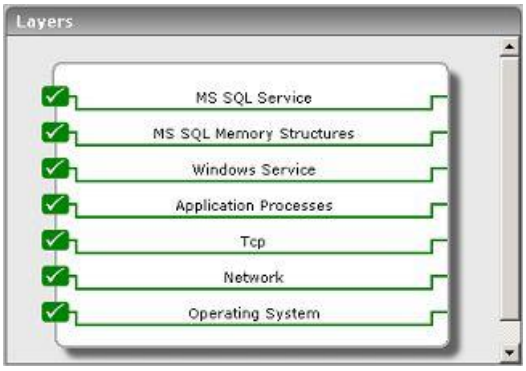


Figure 12.1: The layer model of a Backup SQL server

This section will deal with all other layers except the bottom 3 layers, as these layers have already been discussed in the *Monitoring Unix and Windows Servers* document.

12.1 The Application Processes Layer

This layer checks whether the Backup SQL server process is running or not.

Figure 12.2: The tests associated with the Application Processes layer

12.1.1 Backup Processes Test

For every process pattern configured for a Backup SQL server, the process test reports a variety of CPU and memory statistics. By default, the test reveals the current status and resource usage of the critical SQL server process.

Purpose	To measure statistics pertaining to one or more processes executing on a Backup SQL server		
Target of the test	A Backup SQL server		
Agent deploying the test	An internal agent		
Configurable parameters for 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 to which the specified **HOST** listens 4. **PROCESS** - In the **PROCESS** text box, enter a comma separated list of names:pattern pairs which identify the process(es) associated with the server being considered. processName is a string that will be used for display purposes only. processPattern is an expression of the form - **expr** or expr or **expr or expr** or *expr1*expr2*... or expr1*expr2, etc. A leading '*' signifies any number of leading characters, while a trailing '*' signifies any number of trailing characters. The pattern(s) used vary from one application to another and must be configured per application. For example, for an iPlanet application server (Nas_server), there are three processes named kcs, kjs, and kxs associated with the application server. For this server type, in the **PROCESS** text box, enter "kcsProcess:*kcs*, kjsProcess:*kjs*, kxsProcess:*kxs*, where * denotes zero or more characters. Other special characters such as slashes (\) can also be used while defining the process pattern. For example, if a server's root directory is /home/egurkha/apache and the server executable named httpd exists in the bin directory, then, the process pattern is "*/home/egurkha/apache/bin/httpd*".		
Outputs of the test	One set of results per process pattern specified		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Processes running: Number of instances of a process(es) currently executing on a host.	Number	This value indicates if too many or too few processes corresponding to an application are executing on the host.

	CPU utilization: Percentage of CPU used by executing process(es) corresponding to the pattern specified.	Percent	A very high value could indicate that processes corresponding to the specified pattern are consuming excessive CPU resources.
	Memory utilization: For one or more processes corresponding to a specified set of patterns, this value represents the ratio of the resident set size of the processes to the physical memory of the host system, expressed as a percentage.	Percent	A sudden increase in memory utilization for a process(es) may be indicative of memory leaks in the application.

12.2 The Windows Service Layer

The **BackupSrvc** test mapped to this layer, by default, periodically monitors the availability of the critical MS SQL service on the Backup SQL server.

Figure 12.3: The tests associated with the Windows Service layer

12.2.1 Backup Service Test

This test checks the availability of the service that corresponds to the Backup SQL server.

Purpose	To check the availability of a service that corresponds to the Backup SQL server
Target of the test	A Backup SQL server
Agent deploying the test	An internal agent

Configurable parameters for 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 to which the specified **HOST** listens 4. **SERVICENAME** - Name of the service that is to be checked. More than one service name can also be provided with comma as the separator.		
Outputs of the test	One set of results for every ServiceName that has been configured.		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Availability: Indicates the availability of the service.	Percent	A value of 100 indicates that the specified service has been configured and is currently executing. A value of 0 for this measure indicates that the specified service has been configured on the server but is not running at this time. A value of –1 indicates that the service has not been configured on the target system.

12.3 The MS SQL Memory Structures Layer

This layer tracks the health of the memory and buffer structures of a Backup SQL server.

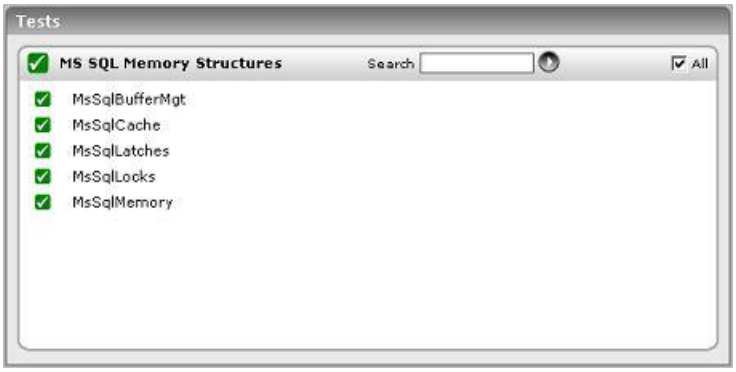


Figure 12.4: The tests associated with the MS SQL Memory Structures layer

All these tests have already been discussed elaborately in Chapter 0 of this document.

12.4 The MS SQL Service Layer

The tests associated with this layer track the health of the services associated with a Backup SQL server.

Figure 12.5: The tests associated with the MS SQL Service layer

Except the BackupSql test in Figure 12.5 all other tests have been discussed in Chapter 0 of this document.

12.4.1 Backup SQL Test

This test monitors the availability and response time from clients by the Backup MS Sql database server in a cluster.

Purpose	Monitors the availability and response time of a Backup MS SQL server in a cluster		
Target of the test	A Backup MS SQL server		
Agent deploying the test	An internal agent		
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the server 3. **PORT** – The port on which the server is listening 4. **USER –** A database user name. 5. **PASSWORD**- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** (if any) by retyping it here. 7. **DATABASE** - The name of the database to connect to. The default is "master". 8. **QUERY** – The select query to execute. The default is "select * from master.dbo.spt_monitor". 9. **CASE** – Takes the value "upper" or "lower" depending upon the case-sensitivity of the SQL server installation. 10. **CLUSTERNAME** – The IP/hostname of the primary MS SQL server in a cluster 11. **CLUSTERPORT** – The port number at which the primary MS SQL server listens		
Outputs of the test	One set of results for the Backup SQL server being monitored		
	Measurement	**Measurement Unit**	**Interpretation**

Measurements made by the test	**Service availability:** Indicates the availability of the server	Percent	The availability is 100% when the server is responding to a request and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the database server, or if the server has not been started.
	Response time: Indicates the time taken by the database to respond to a user query	Seconds	A sudden increase in response time is indicative of a bottleneck at the database server.

b.

13

Monitoring the PostgreSQL Server

PostgreSQL, often simply **Postgres**, is an object-relational database management system (ORDBMS) available for many platforms including Linux, FreeBSD, Solaris, Microsoft Windows and Mac OS X. It implements the majority of the SQL:2008 standard, is ACID-compliant, is fully transactional (including all DDL statements), has extensible data types, operators, and indexes, and has a large number of extensions written by third parties.

Owing to its ability to operate on heterogeneous platforms, the PostgreSQL has of late become the preferred backend for many mission-critical service offerings. A second's non-availability of the server, a sudden or steady erosion of free space in one/more of its tablespaces, ineffective caching by the server, and intense locking can cause serious harm to not only the performance of the PostgreSQL server in question, but also the services that rely on it. Continuous monitoring of the database server and prompt detection and resolution of anomalies is hence imperative.

eG Enterprise offers a 100%, web-based *PostgreSQL* monitoring model (see Figure 13.1) that provides indepth insights into the performance and problems related to the PostgreSQL database server. **This model can be used for monitoring PostgreSQL version 9.0 onwards**.

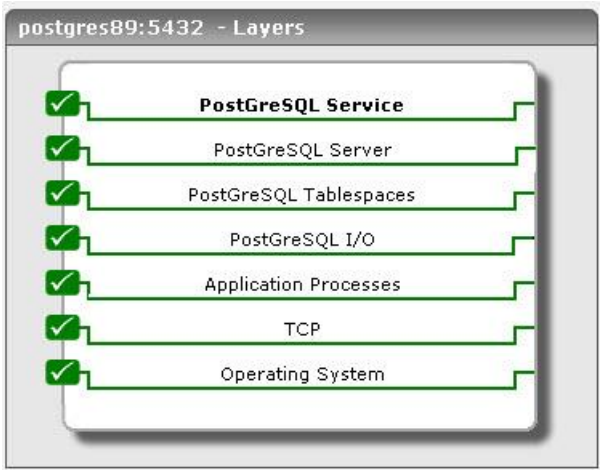


Figure 13.1: Layermodel of the PostgresSQL database server

This model can be configured to employ *agent-based* or *agentless* techniques to periodically check the status of critical database operations and proactively report problems. These metrics enable database administrators to find quick and accurate answers to the following performance queries:

- Is the database server available? If so, how quickly does it repond to client queries?
- Is the buffer cache utilized optimally, or are requests for heap blocks and index blocks being increasingly serviced by direct disk accesses?
- Is any tablespace running low on free space? If so, which one?

- How well does the background writer perform checkpointing? Is too much I/O load being imposed by the writer in the process of checkpointing?
- Are too many rollbacks occurring on any database? If so, which one?
- Are indexes used effectively?
- Are there any useless/unused indexes on the server? Which ones are these?
- Have too many sequential scans occurred on any table?
- Are inserts, updates, and deletes happening too slowly on any table?
- Is any table experiencing extreme or major issues while querying data from the server?
- Does any user have too many idle connections on the server?
- Is any user's connection waiting for a locked resource to be released?
- Are too many locks being currently held on the server? Which lock mode is the maximum?
- Are any queries running for too long a time on the server? If so, which ones are these?

The sections that follow will deal with the top four layers of Figure 13.1 as the other layers have already been dealt with in the *Monitoring Unix and Windows* servers document.

13.1 PostGreSQL I/O

Use the tests mapped to this layer to figure out how the server performs caching and how well the buffer cache is utilized. Inadequacies in the cache size are thus revealed.

Figure 13.2: The tests mapped to the PostGreSQL I/O

13.1.1 PostgreSQL Table I/O Test

In PostgreSQL, data is stored in tables, and tables are grouped into databases. Each table is stored in its own disk file. The contents of a table are stored in *pages*. A table can span many pages, depending upon the length of the row data in the table. A page that contains row data is called a *heap block*. As indexes are also stored in page files, a page that contains index data is called an *index block*.

Typically, in PostgreSQL, most disk I/O is performed on a page-by-page basis. To minimize disk I/O, PostgreSQL creates an in-memory data structure known as the *buffer cache* to which the frequently accessed data is stored. The buffer cache is organized as a collection of 8K pages—each page in the buffer cache corresponds to a page in some page file. The buffer cache is shared between all processes servicing a given database.

When you select a row from a table, PostgreSQL will read the heap block that contains the row into the buffer cache. If there is not enough free space in the cache, PostgreSQL will move some other block out of the cache. If a block being removed from the cache has been modified, it will be written back out to disk; otherwise, it will simply be discarded. Index blocks are also buffered in a similar manner.

If the buffer cache is not sized right, it may not be able to hold enough heap or index blocks to serve subsequent requests. If queries do not find the heap/index blocks they need in the buffer cache, they will be forced to access the disk directly to retrieve data. As direct disk accesses are I/O-intensive operations, they may cause serious performance degradations if not nipped in the bud!

Using the **PostgreSQL Table I/O** test, you can continuously monitor the heap blocks read from the tables in configured databases and index blocks read from the indexes that correspond to those tables. In the process, you can understand how the buffer cache serviced these read requests and learn of ineffective cache usage early, so that you can investigate the reasons for the same (whether/not it is owing to an under-sized cache) and initiate appropriate remedial action.

Purpose	You can continuously monitor the heap blocks read from configured tables and index blocks read from the indexes that correspond to those tables. In the process, you can understand how the buffer cache serviced these read requests and learn of ineffective cache usage early, so that you can investigate the reasons for the same (whether/not it is owing to an under-sized cache) and initiate appropriate remedial action
Target of the test	PostgreSQL server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the server
3. PORT – The port on which the server is listening. The default port is 5432.
4. USER – In order to monitor a PostgreSQL server, you need to manually create a special database user account in every PostgreSQL database instance that requires monitoring. When doing so, ensure that this user is vested with the superuser privileges. The sample script we recommend for user creation for eG monitoring is:
CREATE ROLE eguser LOGIN
ENCRYPTED PASSWORD {'eguser password'}
SUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;
The name of this user has to be specified in the USERNAME text box.
5. PASSWORD- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password.
6. CONFIRM PASSWORD – Confirm the PASSWORD (if any) by retyping it here.
7. DBNAME - The name of the database to connect to. The default is "postgres".
8. INCLUDE DB - Specify a comma-separated list of databases that you wish to monitor.
9. EXCLUDE DB - Specify a comma-separated list of databases that need to be excluded from monitoring.
10. SSL - The name of this user has to be specified in the USERNAME text box.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every table (and corresponding index) in every database that is configured for monitoring in the target PostgreSQL server</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Heap blocks read:
Indicates the rate at which the heap blocks are read from this table.</td><td>Reads/Sec</td><td></td></tr>
<tr><td>Heap blocks hit:
Indicates the number of heap block requests to this table that were serviced by the buffer cache during the last measurement period.</td><td>Number</td><td>Ideally, the value of this measure should be high.</td></tr>
</table>

	Heap hit ratio: Indicates the ratio of the heap block read requests to this table to the heap block requests found in the buffer cache.	Percent	Ideally, the value of this measure should be high. A low value is indicative ineffective cache usage, which in turn can increase disk I/O and degrade server performance. One of the most common reasons for a low cache hit ratio is small cache size. In such a case, you can consider increasing the cache size. There are two ways that you can adjust the size of the cache. You could edit PostgreSQL's configuration file (**$PGDATA/postgresql.conf**) and change the **shared_buffers** variable therein. Alternatively, you can override the **shared_buffers** configuration variable when you start the **postmaster**. A sample command for implementing a **shared_buffers** override while starting the **postmaster** is given below: *pg_start -o "-B 65" -l /tmp/pg.log* If increasing the cache size also does not help, then, you can include a **LIMIT** clause in your queries to select a sub-set of the queried tables and add them to the cache.
	Index block reads: Indicates the rate at which the index blocks were read from the indexes of this table.	Reads/Sec	
	Blocks hit: Indicates the number of read requests to the indexes of this table that were found in the buffer cache during the last measurement period.	Number	Ideally, the value of this measure should be high.

	Block hit ratio: Indicates the percentage of index block requests to the indexes of this table that were served by the buffer cache.	Percent	Ideally, the value of this measure should be high. A low value is indicative of ineffective cache usage, which in turn can increase disk I/O and degrade server performance. One of the most common reasons for a low cache hit ratio is small cache size. In such a case, you can consider increasing the cache size. There are two ways that you can adjust the size of the cache. You could edit PostgreSQL's configuration file (**$PGDATA/postgresql.conf**) and change the **shared_buffers** variable therein. Alternatively, you can override the **shared_buffers** configuration variable when you start the **postmaster**. A sample command for implementing a **shared_buffers** override while starting the **postmaster** is given below: *pg_start -o "-B 65" -l /tmp/pg.log* If increasing the cache size also does not help, then, you can include a **LIMIT** clause in your queries to select a sub-set of the queried tables and add them to the cache.

13.1.2 PostgreSQL Index I/O Test

Indexes are buffered in the same way as tables are. Therefore, if too many index blocks are not found in the buffer cache, disk I/O increases, causing the overall performance of the PostgreSQL server to suffer. It is hence imperative to monitor how the cache services index block read requests.

The **PostgreSQL Index I/O** test helps monitor each index in a database for read requests. In the process, the test reveals how the buffer cache serviced these read requests and provides early pointers to ineffective cache usage, so that you can investigate the reasons for the same (whether/not it is owing to an under-sized cache) and initiate appropriate remedial action.

Purpose	Helps monitor each index in a database for read requests. In the process, the test reveals how the buffer cache serviced these read requests and provides early pointers to ineffective cache usage, so that you can investigate the reasons for the same (whether/not it is owing to an under-sized cache) and initiate appropriate remedial action
Target of the test	A PostgreSQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the server 3. **PORT** – The port on which the server is listening. The default port is 5432. 4. **USER –** In order to monitor a PostgreSQL server, you need to manually create a special database user account in every PostgreSQL database instance that requires monitoring. When doing so, ensure that this user is vested with the *superuser* privileges. The sample script we recommend for user creation for eG monitoring is: *CREATE ROLE eguser LOGIN* *ENCRYPTED PASSWORD {'eguser password'}* *SUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;* The name of this user has to be specified in the **USERNAME** text box. 5. **PASSWORD**- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** (if any) by retyping it here. 7. **DBNAME** - The name of the database to connect to. The default is "postgres". 8. **INCLUDE DB** - Specify a comma-separated list of databases that you wish to monitor. 9. **EXCLUDE DB** - Specify a comma-separated list of databases that need to be excluded from monitoring. 10. **SSL** - The name of this user has to be specified in the **USERNAME** text box.		
Outputs of the test	One set of results for every in index in every database that is configured for monitoring in the target PostgreSQL server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Index block reads: Indicates the rate at which the index blocks were read from this index during the last measurement period.	Reads/Sec	
	Blocks hit: Indicates the number of read requests to this index that were found in the buffer cache during the last measurement period.	Number	Ideally, the value of this measure should be high.

	Hit ratio: Indicates the percentage of index block requests to this index that were served by the buffer cache.	Percent	Ideally, the value of this measure should be high. A low value is indicative ineffective cache usage, which in turn can increase disk I/O and degrade server performance. One of the most common reasons for a low cache hit ratio is small cache size. In such a case, you can consider increasing the cache size. There are two ways that you can adjust the size of the cache. You could edit PostgreSQL's configuration file (**$PGDATA/postgresql.conf**) and change the **shared_buffers** variable therein. Alternatively, you can override the **shared_buffers** configuration variable when you start the **postmaster**. A sample command for implementing a **shared_buffers** override while starting the **postmaster** is given below: *pg_start -o "-B 65" -l /tmp/pg.log* If increasing the cache size also does not help, then, you can include a **LIMIT** clause in your queries to select a sub-set of the queried tables and add them to the cache.

13.2 PostGreSQL Tablespaces

To know whether any tablespace has been excessively utilized, use the test mapped to this layer.

Figure 13.3: The test mapped to the PostGreSQL Tablespaces layer

13.2.1 PostgreSQL Tablespaces Test

Tablespaces in PostgreSQL allow database administrators to define locations in the file system where the files representing database objects can be stored. Once created, a tablespace can be referred to by name when creating database objects.

By using tablespaces, an administrator can control the disk layout of a PostgreSQL installation. This is useful in at least two ways. First, if the partition or volume on which the cluster was initialized runs out of space and cannot be extended, a tablespace can be created on a different partition and used until the system can be reconfigured.

Second, tablespaces allow an administrator to use knowledge of the usage pattern of database objects to optimize performance. For example, an index which is very heavily used can be placed on a very fast, highly available disk, such as an expensive solid state device. At the same time a table storing archived data which is rarely used or not performance critical could be stored on a less expensive, slower disk system.

Tablespaces should be adequately sized. If not, the tablespaces may not be able to accomodate many critical database objects, thereby causing the performance of the database to suffer. Continuous monitoring of tablespace size and usage is hence important. The **PostgreSQL Tablespaces** test does just that. This test auto-discovers tablespaces managed by this PostgreSQL server and reports how well the tablespace has been utilized.

Purpose	Auto-discovers tablespaces managed by this PostgreSQL server and reports how well the tablespace has been utilized
Target of the test	A PostgreSQL server
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the server 3. **PORT** – The port on which the server is listening. The default port is 5432. 4. **USER –** In order to monitor a PostgreSQL server, you need to manually create a special database user account in every PostgreSQL database instance that requires monitoring. When doing so, ensure that this user is vested with the *superuser* privileges. The sample script we recommend for user creation for eG monitoring is: *CREATE ROLE eguser LOGIN* *ENCRYPTED PASSWORD {'eguser password'}* *SUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;* The name of this user has to be specified in the **USERNAME** text box. 5. **PASSWORD**- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** (if any) by retyping it here. 7. **DBNAME** - The name of the database to connect to. The default is "postgres". 8. **INCLUDE DB** - Specify a comma-separated list of databases that you wish to monitor. 9. **EXCLUDE DB** - Specify a comma-separated list of databases that need to be excluded from monitoring. 10. **SSL** - The name of this user has to be specified in the **USERNAME** text box.

Outputs of the test	One set of results for every tablespace in every database that is configured for monitoring in the target PostgreSQL server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Tablespace size: Indicates the amount of space currently used in this tablespace.	MB	A high value of this measure indicates that the table consumes a large chunk of space in the tablespace which may cause serious performance issues ranging from slowdown to shutdowns of this database.

13.3 PostGreSQL Server

Using the tests mapped to this layer, you can determine:

uu. Whether/not the background writer minimizes the I/O load on the server;

vv. How well the server handles the transaction load to it, and whether any processing pain-points can be noticed;

ww. Whether/ not the indexes are properly used;

xx. The number and names of unused indexes (if any);

yy. The count of sequential scans and index scans that occurred per table and the rows that were returned in the process.

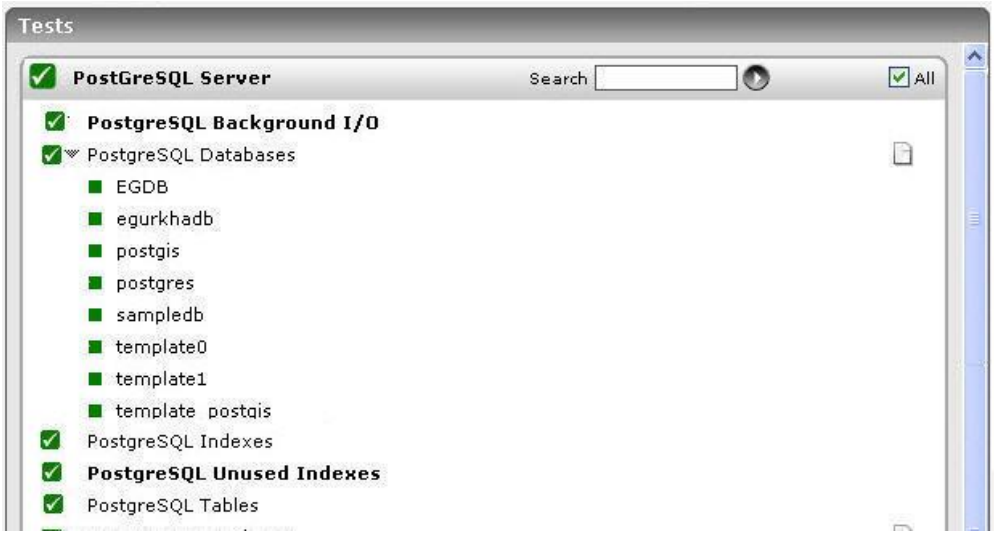


Figure 13.4: The tests mapped to the PostGreSQL Server layer

13.3.1 PostgreSQL Background I/O Test

Checkpoints are points in the sequence of transactions at which it is guaranteed that the heap and index data files have been updated with all information written before the checkpoint. At checkpoint time, all dirty data pages are flushed to disk and a special checkpoint record is written to the log file. In the event of a crash, the crash recovery

procedure looks at the latest checkpoint record to determine the point in the log (known as the redo record) from which it should start the REDO operation. Any changes made to data files before that point are guaranteed to be already on disk.

The checkpoint requirement of flushing all dirty data pages to disk can cause a significant I/O load. To minimize this I/O, there is a separate server process called the background writer in PostgreSQL, whose sole function is to issue writes of "dirty" shared buffers. The background writer will continuously trickle out dirty pages to disk, so that only a few pages will need to be forced out when checkpoint time arrives, instead of the storm of dirty-buffer writes that formerly occurred at each checkpoint. However, there is a net overall increase in I/O load, because where a repeatedly-dirtied page might before have been written only once per checkpoint interval, the background writer might write it several times in the same interval.

You hence need to continuously track how often the background writer performs checkpointing and how much I/O load it imposes on the server, so that you can proactively detect potential overload conditions, appropriately fine-tune the checkpointing activity performed by the background writer to minimize the I/O, and thus prevent the performance degradation that may otherwise occur on the server. The **PostgreSQL Background I/O** test helps achieve all of the above. In the process, the test also reports useful statistics related to shared buffers.

Purpose	Continuously tracks how often the background writer performs checkpointing and how much I/O load it imposes on the server, so that you can proactively detect potential overload conditions, appropriately fine-tune the checkpointing activity performed by the background writer to minimize the I/O, and thus prevent the performance degradation that may otherwise occur on the server
Target of the test	A PostgreSQL server
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the server 3. **PORT** – The port on which the server is listening. The default port is 5432. 4. **USER –** In order to monitor a PostgreSQL server, you need to manually create a special database user account in every PostgreSQL database instance that requires monitoring. When doing so, ensure that this user is vested with the *superuser* privileges. The sample script we recommend for user creation for eG monitoring is: *CREATE ROLE eguser LOGIN* *ENCRYPTED PASSWORD {'eguser password'}* *SUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;* The name of this user has to be specified in the **USERNAME** text box. 5. **PASSWORD**- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** (if any) by retyping it here. 7. **DBNAME** - The name of the database to connect to. The default is "postgres". 8. **SSL** - The name of this user has to be specified in the **USERNAME** text box.
Outputs of the test	One set of results for the target PostgreSQL server

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Checkpoint requests: Indicates the number of checkpoint requests received by the server during the last measurement period.	Number	A checkpoint request is generated every *checkpoint_segments* log segments, or every *checkpoint_timeout* seconds, whichever comes first. While *checkpoint_segments* denotes the maximum number of log file segments between automatic WAL checkpoints, the *checkpoint_timeout* indicates the maximum time between WAL checkpoints. The default settings are 3 segments and 300 seconds (5 minutes), respectively. Reducing checkpoint_segments and/or checkpoint_timeout causes checkpoints to occur more often. This allows faster after-crash recovery (since less work will need to be redone). However, one must balance this against the increased cost of flushing dirty data pages more often. If full_page_writes is set (as is the default), there is another factor to consider. To ensure data page consistency, the first modification of a data page after each checkpoint results in logging the entire page content. In that case, a smaller checkpoint interval increases the volume of output to the WAL log, partially negating the goal of using a smaller interval, and in any case causing more disk I/O. Checkpoints are fairly expensive, first because they require writing out all currently dirty buffers, and second because they result in extra subsequent WAL traffic as discussed above. It is therefore wise to set the checkpointing parameters high enough that checkpoints don't happen too often.
	Check point time outs: Indicates the number of scheduled checkpoints that did not occur even after the *checkpoint_timeout* setting was violated during the last measurement period.	Number	Ideally, the value of this measure should be low. A consistent increase in this value is a cause of concern, as it indicates that checkpoints are not occurring in the desired frequency. This in turn will significantly slowdown after-crash recovery, as more work will have to be redone.

	Buffers freed: Indicates the total number of buffers that were released for re-use from the buffer cache during the last measurement period, when the *checkpoint_segments* setting was violated; this typically causes the background writer to automatically write dirty buffers to the disk.	Number	A high value is desired for this measure. A low value could indicate a checkpointing bottleneck, owing to which the background writer is unable to write updated index and heap files to the disk at an optimal rate. In such cases, the buffer cache may not have adequate free buffers to service subsequent write requests. This is a cause for concern in write-intensive database environments.
	Buffers cleaned: Indicates the number of buffer that were written to the disk during the last measurement period in anticipation of being allocated in the future.	Number	The background writer typically stalls some other process for a moment while it writes out dirty data. To keep that from happening as often, the background writer process scans forward looking for blocks that might be allocated in the near future that are dirty and that have a low usage count (alternatively called the Least Recently Used or LRUblocks). When it finds them, it writes some of them out pre-emptively, based on historical allocation rates.
	Max written: Indicates the maximum number of dirty buffers that can be written into the buffer cache during the last measurement period.	Number	If this measure indicates a high value it indicates that adequate buffers are not free in the cache. To optimize the value of this measure, you can increase the value of the *bgwriter_lru_maxpages* parameter.
	Buffers freed by connections: Indicates the number of buffers that were released from the cache for re-use during the last measurement period, when users wrote data directly to the disk.	Number	A high value is desired for this measure, as it reduces the need for an I/O-intensive operation such as 'checkpointing'.

	Buffers allocated: Indicates the total number of calls to allocate a new buffer for a page (whether or not it was already cached) during the last measurement period.	Number	

13.3.2 PostgreSQL Databases Test

For each database on the PostgreSQL server, this test reports the transaction load on the database and reveals how well the database processes the transaction requests to it and how well it utilizes its cache. Overload conditions and processing bottlenecks are thus revealed.

Purpose	Reports the transaction load on the database and reveals how well the database processes the transaction requests to it and how well it utilizes its cache
Target of the test	A PostgreSQL server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the server
3. PORT – The port on which the server is listening. The default port is 5432.
4. USER – In order to monitor a PostgreSQL server, you need to manually create a special database user account in every PostgreSQL database instance that requires monitoring. When doing so, ensure that this user is vested with the superuser privileges. The sample script we recommend for user creation for eG monitoring is:
CREATE ROLE eguser LOGIN
ENCRYPTED PASSWORD {'eguser password'}
SUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;
The name of this user has to be specified in the USERNAME text box.
5. PASSWORD- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password.
6. CONFIRM PASSWORD – Confirm the PASSWORD (if any) by retyping it here.
7. DBNAME - The name of the database to connect to. The default is "postgres".
8. SSL - The name of this user has to be specified in the USERNAME text box.
9. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for every database on the target PostgreSQL server</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td></td><td>Database size:
Indicates the current size of this database.</td><td>KB</td><td></td></tr>
</table>

	Cache hit ratio: Indicates the percentage of requests to this database that were serviced by the cache, without having to read from disk.	Percent	Because reading from the cache is less expensive than reading from disk, you want the ratio to be high. The higher this value is, the better. Generally, you can increase the cache hit ratio by increasing the amount of memory available to the database server. The detailed diagnosis of this measure provides you with the complete details of the database such as the number of server processes running on it, the number of transactions committed and rolled back, and the number of rows inserted, updated, and deleted.
	Commit ratio: Indicates the rate at which live rows are fetched while this index is scanned.	Percent	
	Server process: Indicates the number of processes that are currently running on this database.	Number	
	Inserts: Indicates the rate at which the records are inserted into this database.	Inserts/Sec	
	Deletes: Indicates the rate at which the records are deleted from this database.	Deletes/Sec	
	Updates: Indicates the rate at which records are updated into this database.	Updates/Sec	
	Commits: Indicates the transaction throughput.	Commits/Sec	A decrease in this measure during the monitoring period may indicate that the applications are not doing frequent commits. This may lead to problems with logging and data concurrency. The cause has to be probed in the application.

	Rollbacks: Indicates the rate at which rollbacks occurred on this database.	Rollbacks/Sec	A high rollback rate is an indicator of bad performance, since work performed up to the rollback point is wasted. The cause of the rollbacks has to be probed in the application.
	Rows fetched: Indicates the rate at which the rows that were read from this database based on a user query are stored in the buffer.	Fetches/Sec	
	Rows returned: Indicates the rate at which the rows are fetched from the buffer and sent to the client application.	Returns/Sec	If the size of the rows that are fetched from the buffer is too large, then the rows are fragmented and transferred to the client which is time consuming. This may in turn affect the performance of the database to some extent.
	Blocks read: Indicates the rate at which the blocks are read from this database.	Fetches/Sec	
	Block hits: Indicates the rate at which the blocks are fetched after a read is performed in this database.	Hits/Sec	

13.3.3 PostgreSQL Indexes Test

An *index* is a data structure that a database uses to reduce the amount of time it takes to perform certain operations. An index can also be used to ensure that duplicate values don't appear where they are not needed.

This test monitors the indexes on the PostgreSQL server and helps administrators quickly and accurately assess the effectiveness of these indexes.

Purpose	Monitors the indexes on the PostgreSQL server and helps administrators quickly and accurately assess the effectiveness of the database indexes
Target of the test	A PostgreSQL server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the server 3. **PORT** – The port on which the server is listening. The default port is 5432. 4. **USER –** In order to monitor a PostgreSQL server, you need to manually create a special database user account in every PostgreSQL database instance that requires monitoring. When doing so, ensure that this user is vested with the *superuser* privileges. The sample script we recommend for user creation for eG monitoring is: *CREATE ROLE eguser LOGIN* *ENCRYPTED PASSWORD {'eguser password'}* *SUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;* The name of this user has to be specified in the **USERNAME** text box. 5. **PASSWORD**- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** (if any) by retyping it here. 7. **DBNAME** - The name of the database to connect to. The default is "postgres". 8. **INCLUDE DB** - Specify a comma-separated list of databases that you wish to monitor. 9. **EXCLUDE DB** - Specify a comma-separated list of databases that need to be excluded from monitoring. 10. **SSL** - The name of this user has to be specified in the **USERNAME** text box.		
Outputs of the test	One set of results for every index for every table in each database that is configured for monitoring on the target PostgreSQL server		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Index scans: Indicates the rate at which the index scans are initiated on this index in this database .	Scans/Sec	
	Rows read: Indicates the rate at which the index entries (rows) are read during the index scans on this index.	Reads/Sec	
	Rows fetched: Indicates the rate at which the rows are fetched from this index upon execution of a query.	Fetches/Sec	If the value of this measure is greater than the value of the *Rows read* measure, it indicates a possibility of index fragmentation or that the executed query is inefficient.

13.3.4 PostgreSQL Unused Indexes Test

While at one end indexes greatly enhance database performance, at the other they also add significant overhead to table change operations. Useless/unused indices can therefore be unnecessary resource hogs. Such indexes are typically not used by any regular query and may not enforce a constraint. However, these unneeded indexes cost you in several ways: they slow updates, inserts and deletes; they may keep HOT from updating the row in-place, requiring more VACUUMs; they take time to VACUUM; they add to query planning time; they take time to backup and restore. Administrators hence need to identify such indexes and eliminate them. The **PostgreSQL Unused Indexes** test helps administrators achieve the same. This test reports the number and names of unused/useless indexes, and thus prompts administrators to remove them so as to save the server from unnecessary performance degradations.

Purpose	Reports the number and names of unused/useless indexes, and thus prompts administrators to remove them so as to save the server from unnecessary performance degradations
Target of the test	A PostgreSQL server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the server
3. PORT – The port on which the server is listening. The default port is 5432.
4. USER – In order to monitor a PostgreSQL server, you need to manually create a special database user account in every PostgreSQL database instance that requires monitoring. When doing so, ensure that this user is vested with the superuser privileges. The sample script we recommend for user creation for eG monitoring is:
CREATE ROLE eguser LOGIN
ENCRYPTED PASSWORD {'eguser password'}
SUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;
The name of this user has to be specified in the USERNAME text box.
5. PASSWORD- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password.
6. CONFIRM PASSWORD – Confirm the PASSWORD (if any) by retyping it here.
7. DBNAME - The name of the database to connect to. The default is "postgres".
8. INCLUDE DB - Specify a comma-separated list of databases that you wish to monitor.
9. EXCLUDE DB - Specify a comma-separated list of databases that need to be excluded from monitoring.
10. SSL - The name of this user has to be specified in the USERNAME text box.
11. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the target PostgreSQL server</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td></td><td>Number of indexes:
Indicates the number of indexes that are currently unused/useless on the server.</td><td>Number</td><td>A high value of this measure is a cause for concern. Use the detailed diagnosis of this measure to identify the unused indexes and take measures to get rid of them.</td></tr>
</table>

13.3.5 PostgreSQL Tables Test

The real test of the performance of a database server lies in how quickly the database responds to queries. Whenever users complaint of slow execution of their queries, administrators need to know the reason for the delay - is it because the queries themselves are badly designed? or is it due to how the database server performs table scans and returns the requested result set to the queries? The **PostgreSQL Tables** test helps with this root-cause analysis. This test auto-discovers the tables in the configured databases and reports the number of times every table was scanned, the type of scanning (sequential or index) that was performed, and the rate at which the server reads data (via index and sequential scans) from each table. On the basis of this data, the test also indicates if any table is experiencing any query processing bottlenecks, and if so, how severe is the problem. In addition, the test also reveals how quickly critical database operations such as inserts, deletes, and updates, are performed on every table. Using this information, administrator can figure out whether/not the number and nature of scans performed on the tables are causing queries to the corresponding database to slowdown.

Purpose	Auto-discovers the tables in the configured databases and reports the number of times every table was scanned, the type of scanning (sequential or index) that was performed, and the rate at which the server reads data (via index and sequential scans) from each table
Target of the test	A PostgreSQL server
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the server 3. **PORT** – The port on which the server is listening. The default port is 5432. 4. **USER –** In order to monitor a PostgreSQL server, you need to manually create a special database user account in every PostgreSQL database instance that requires monitoring. When doing so, ensure that this user is vested with the *superuser* privileges. The sample script we recommend for user creation for eG monitoring is: *CREATE ROLE eguser LOGIN* *ENCRYPTED PASSWORD {'eguser password'}* *SUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;* The name of this user has to be specified in the **USERNAME** text box. 5. **PASSWORD**- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** (if any) by retyping it here. 7. **DBNAME** - The name of the database to connect to. The default is "postgres". 8. **INCLUDE DB** - Specify a comma-separated list of databases that you wish to monitor. 9. **EXCLUDE DB** - Specify a comma-separated list of databases that need to be excluded from monitoring. 10. **SSL** - The name of this user has to be specified in the **USERNAME** text box.
Outputs of the test	One set of results for each table on every database configured for monitoring on the target PostgreSQL server

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Sequence scans count: Indicates the number of sequential scans initiated on this table during the last measurement period.	Number	**Sequential or Full table scan** is a scan made on the database where each row of the table under scan is read in a sequential (serial) order and the columns encountered are checked for the validity of a condition. Full table scans are usually the slowest method of scanning a table due to the heavy amount of I/O reads and writes required from the disk which consists of multiple seeks as well as costly disk to memory transfers. Typically therefore, a low value is desired for this measure. However, if a query returns more than approximately 5-10% of all rows in the table, then PostgreSQL prefers the sequential scan over the index scan. This is because an index scan requires *several* I/O operations for each row (look up the row in the index, then retrieve the row from the heap). Whereas a sequential scan only requires a single I/O for each row - or even less because a block (page) on the disk contains more than one row, so more than one row can be fetched with a single I/O operation.
	Sequence reads row count: Indicates the number of rows that are processed through sequential scan from this table during the last measurement period.	Number	
	Average reads per scan: Indicates the rate at which rows from this table were processed through a sequential scan.	Fetches/Sec	A high value is desired for this measure. If the value is low or falls consistently, it indicates bottlenecks while performing sequential scans on the table.

	Index scans: Indicates the number of index scans initiated over all the indexes belonging to this table during the last measurement period.	Number	An *index scan* occurs when the database manager accesses an index for any of the following reasons: • To narrow the set of qualifying rows (by scanning the rows in a certain range of the index) before accessing the base table. • To order the output. • To retrieve the requested column data directly. If all of the requested data is in the index, the indexed table does not need to be accessed. This is known as an *index-only access*. Typically, a high value of this measure is desired, as index scans are I/O-friendly operations. However, if a query returns more than approximately 5-10% of all rows in the table, then PostgreSQL prefers the sequential scan over the index scan. This is because an index scan requires *several* I/O operations for each row (look up the row in the index, then retrieve the row from the heap). Whereas a sequential scan only requires a single I/O for each row - or even less because a block (page) on the disk contains more than one row, so more than one row can be fetched with a single I/O operation.
	Average fetch per index: Indicates the rate at which the rows are processed through an index scan on this table.	Fetches/Sec	A high value is desired for this measure. If the value is low or falls consistently, it indicates bottlenecks while performing index scans on the table.
	Table scans: Indicates the number of times this table was scanned during the last measurement period.	Number	A high value indicates that there are no proper indexes for this table. This may cause delays in query execution.
	Inserts: Indicates the rate at which the rows are inserted into this table.	Inserts/Sec	
	Deletes: Indicates the rate at which the rows are deleted from this table.	Deletes/Sec	

<table>
<tr><td></td><td>Updates:
Indicates the rate at which the rows are updated in this table.</td><td>Updates/Sec</td><td></td></tr>
<tr><td></td><td>Priority:
Indicates the type of problem that is currently experienced by this table while processing a query.</td><td></td><td>The difference between the Sequence scan count and the Index scan count measures determines the Priority of the problem experienced by a table. The various Priorities this measure reports and their numeric equivalents as shown in the table:
<table><tr><th>Numeric Value</th><th>Priority</th></tr><tr><td>1</td><td>Minor Problem</td></tr><tr><td>2</td><td>Major Problem</td></tr><tr><td>3</td><td>Extreme Problem</td></tr></table>
Note:
By default, this measure reports the above-mentioned States while indicating the type of problem that is experienced while querying this database. However, the graph of this measure will be represented using the corresponding numeric equivalents of the states as mentioned in the table above.
If the severity of this measure is high, it indicates that the query used may be inefficient or there may be a problem with the indexing of the column or there may be a possibility of fragmentation of the table or index of this database.</td></tr>
</table>

13.4 PostGreSQL Service

Besides revealing the availability and responsiveness of the database server, the tests mapped to this layer also sheds light on the idle and waiting user connections on the server, the level of locking activity on the server, and the number and details of queries to the server that have been running for an unreasonably long time.

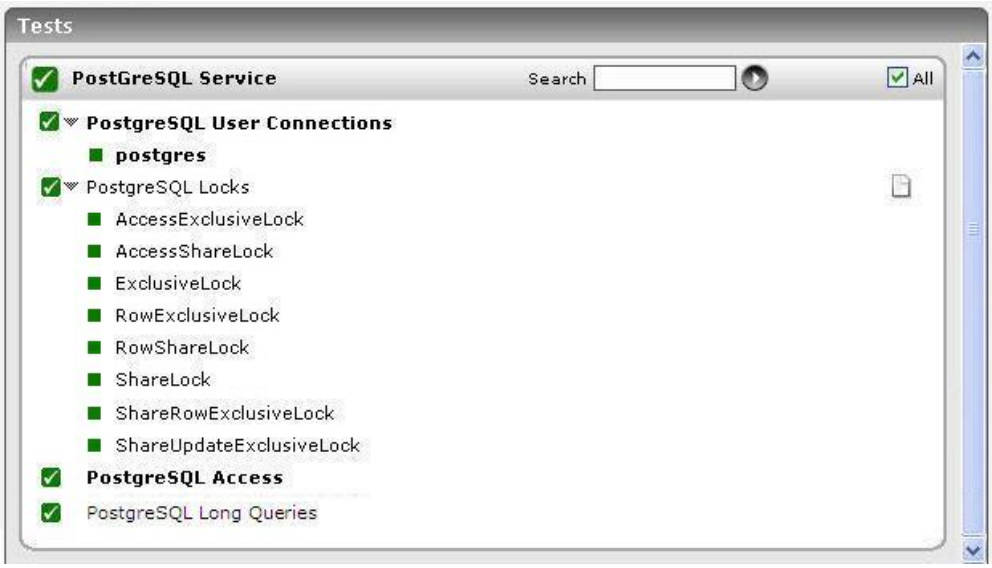


Figure 13.5: The tests mapped to the PostgreSQL Service layer

13.4.1 PostgreSQL User Connections Test

This test monitors the users who are currently connected to the server and reports the number and state of each user connection. Using the metrics reported by this test, administrators can promptly isolate idle and waiting connections, which are a drain on a server's resources.

Purpose	Monitors the users who are currently connected to the server and reports the number and state of each user connection
Target of the test	A PostgreSQL server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the server
3. PORT – The port on which the server is listening. The default port is 5432.
4. USER – In order to monitor a PostgreSQL server, you need to manually create a special database user account in every PostgreSQL database instance that requires monitoring. When doing so, ensure that this user is vested with the superuser privileges. The sample script we recommend for user creation for eG monitoring is:
CREATE ROLE eguser LOGIN
ENCRYPTED PASSWORD {'eguser password'}
SUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;
The name of this user has to be specified in the USERNAME text box.
5. PASSWORD- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password.
6. CONFIRM PASSWORD – Confirm the PASSWORD (if any) by retyping it here.
7. DBNAME - The name of the database to connect to. The default is "postgres".
8. SSL - The name of this user has to be specified in the USERNAME text box.
9. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each user currently connected to the target PostgreSQL server</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td></td><td>Total connections:
Indicates the total number of connections that are currently established by this user on the server.</td><td>Number</td><td></td></tr>
</table>

	Idle connections: Indicates the number of connections of this user that are currently idle on the server.	Number	Ideally, the value of this measure should be low. A high value is indicative of a large number of idle connections, which in turn causes unnecessary consumption of critical server resources. Idle connections also unnecessarily lock new connections from the connection pool, thereby denying other users access to the server for performing important tasks. Use the detailed diagnosis of this measure to view the details of the idle connections.
	Active connections: Indicates the number of connections of this user that are currently active.	Number	Use the detailed diagnosis of this measure to view the details of the active connections.
	Waiting connections: Indicates the number of connections of this user that are currently waiting for a resource/database object/ lock to be released.	Number	The value of this measure should be kept at a minimum, as waiting connections also cause a resource drain. Use the detailed diagnosis of this measure to view the details of the waiting connections.

13.4.2 PostgreSQL Locks Test

PostgreSQL provides various lock modes to control concurrent access to data in tables. These modes can be used for application-controlled locking in situations where MVCC does not give the desired behavior. Also, most PostgreSQL commands automatically acquire locks of appropriate modes to ensure that referenced tables are not dropped or modified in incompatible ways while the command executes. The common lock modes are as follows:

- **ACCESS SHARE**

  Conflicts with the ACCESS EXCLUSIVE lock mode only.

  The SELECT command acquires a lock of this mode on referenced tables. In general, any query that only reads a table and does not modify it will acquire this lock mode.

- **ROW SHARE**

  Conflicts with the EXCLUSIVE and ACCESS EXCLUSIVE lock modes.

  The SELECT FOR UPDATE and SELECT FOR SHARE commands acquire a lock of this mode on the target table(s) (in addition to ACCESS SHARE locks on any other tables that are referenced but not selected FOR UPDATE/FOR SHARE).

- **ROW EXCLUSIVE**

  Conflicts with the SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE, and ACCESS EXCLUSIVE lock modes.

  The commands UPDATE, DELETE, and INSERT acquire this lock mode on the target table (in addition to ACCESS SHARE locks on any other referenced tables). In general, this lock mode will be acquired by any

command that modifies the data in a table.

- **SHARE UPDATE EXCLUSIVE**

  Conflicts with the SHARE UPDATE EXCLUSIVE, SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE, and ACCESS EXCLUSIVE lock modes. This mode protects a table against concurrent schema changes and VACUUM runs.

  Acquired by VACUUM (without FULL), ANALYZE, and CREATE INDEX CONCURRENTLY.

- **SHARE**

  Conflicts with the ROW EXCLUSIVE, SHARE UPDATE EXCLUSIVE, SHARE ROW EXCLUSIVE, EXCLUSIVE, and ACCESS EXCLUSIVE lock modes. This mode protects a table against concurrent data changes.

  Acquired by CREATE INDEX (without CONCURRENTLY).

- **SHARE ROW EXCLUSIVE**

  Conflicts with the ROW EXCLUSIVE, SHARE UPDATE EXCLUSIVE, SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE, and ACCESS EXCLUSIVE lock modes.

  This lock mode is not automatically acquired by any PostgreSQL command.

- **EXCLUSIVE**

  Conflicts with the ROW SHARE, ROW EXCLUSIVE, SHARE UPDATE EXCLUSIVE, SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE, and ACCESS EXCLUSIVE lock modes. This mode allows only concurrent ACCESS SHARE locks, i.e., only reads from the table can proceed in parallel with a transaction holding this lock mode.

  This lock mode is not automatically acquired on user tables by any PostgreSQL command. However it is acquired on certain system catalogs in some operations.

- **ACCESS EXCLUSIVE**

  Conflicts with locks of all modes (ACCESS SHARE, ROW SHARE, ROW EXCLUSIVE, SHARE UPDATE EXCLUSIVE, SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE, and ACCESS EXCLUSIVE). This mode guarantees that the holder is the only transaction accessing the table in any way.

  Acquired by the ALTER TABLE, DROP TABLE, TRUNCATE, REINDEX, CLUSTER, and VACUUM FULL commands. This is also the default lock mode for LOCK TABLE statements that do not specify a mode explicitly.

The locking activity of a database server must be monitored carefully because an application holding a specific lock for a long time could cause a number of other transactions relying on the same lock to fail. The **PostgreSQL Locks** test does just that. For every lock mode that is currently active on the database server, this test reports the total number of locks that are in that mode.

Purpose	For every lock mode that is currently active on the database server, this test reports the total number of locks that are in that mode.
Target of the test	A PostgreSQL server
Agent deploying the test	An internal agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the server
3. PORT – The port on which the server is listening. The default port is 5432.
4. USER – In order to monitor a PostgreSQL server, you need to manually create a special database user account in every PostgreSQL database instance that requires monitoring. When doing so, ensure that this user is vested with the superuser privileges. The sample script we recommend for user creation for eG monitoring is:
CREATE ROLE eguser LOGIN
ENCRYPTED PASSWORD {'eguser password'}
SUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;
The name of this user has to be specified in the USERNAME text box.
5. PASSWORD- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password.
6. CONFIRM PASSWORD – Confirm the PASSWORD (if any) by retyping it here.
7. DBNAME - The name of the database to connect to. The default is "postgres".
8. SSL - The name of this user has to be specified in the USERNAME text box.
9. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each lock mode currently held on the target PostgreSQL server</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

	Number of locks: Indicates the total number of locks that are currently held on the database server.	Number	A high value may indicate one of the following: zz. Too many transactions happening aaa. Locked resources not being released properly bbb. Locks are being held unnecessarily. With the help of the detailed diagnosis of this measure, you can determine the query that is waiting for the lock, the user who executed that query, the query that is blocking, and the user who is executing the blocking query. Once the blocked and blocking queries are isolated, you can then proceed to do what's required to release unnecessary locks.

13.4.3 PostgreSQL Access Test

This test emulates a client executing a configured query on the database server, and in the process reports whether the server is available, and if so, how quickly it responds to the client queries. The unavailability of a network connection to the server and bottlenecks to responsiveness can thus be promptly isolated.

Purpose	Emulates a client executing a configured query on the database server, and in the process reports whether the server is available, and if so, how quickly it responds to the client queries
Target of the test	A PostgreSQL server
Agent deploying the test	An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the target PostgreSQL server is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the target PostgreSQL server is listening.

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD – How often should the test be executed
2. HOST – The IP address of the server
3. PORT – The port on which the server is listening. The default port is 5432.
4. USER – In order to monitor a PostgreSQL server, you need to manually create a special database user account in every PostgreSQL database instance that requires monitoring. When doing so, ensure that this user is vested with the superuser privileges. The sample script we recommend for user creation for eG monitoring is:
CREATE ROLE eguser LOGIN
ENCRYPTED PASSWORD {'eguser password'}
SUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;
The name of this user has to be specified in the USERNAME text box.
5. PASSWORD- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password.
6. CONFIRM PASSWORD – Confirm the PASSWORD (if any) by retyping it here.
7. DBNAME - The name of the database to connect to. The default is "postgres".
8. SSL - The name of this user has to be specified in the USERNAME text box.
9. QUERY - Specify the select query to execute. The default is "select * from pg_tables". Every DATABASE being monitored, should have a corresponding QUERY specification</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the target PostgreSQL server</td></tr>
<tr><td>Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td></td><td>Availability:
Indicates whether the database server is currently available or not.</td><td>Percent</td><td>The availability is 100% when the server is responding to a request and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the database server, or because the server has not been started.</td></tr>
<tr><td></td><td>Response time:
Indicates the time taken by this database to respond to a user query during the last measurement period.</td><td>Secs</td><td>A sudden increase in response time is indicative of a performance bottleneck at the database server.</td></tr>
</table>

13.4.4 PostgreSQL Long Queries Test

This test reports the number of queries that are executing for each database. Using this test, you can identify the query that takes too long to execute and thus the resource-intensive queries to a database can be isolated quickly.

Purpose	Reports the number of queries that are executing for each database. Using this test, you can identify the query that takes too long to execute and thus the resource-intensive queries to a database can be isolated quickly
Target of the test	A PostgreSQL server
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **TEST PERIOD** – How often should the test be executed 2. **HOST** – The IP address of the server 3. **PORT** – The port on which the server is listening. The default port is 5432. 4. **USER –** In order to monitor a PostgreSQL server, you need to manually create a special database user account in every PostgreSQL database instance that requires monitoring. When doing so, ensure that this user is vested with the *superuser* privileges. The sample script we recommend for user creation for eG monitoring is: *CREATE ROLE eguser LOGIN ENCRYPTED PASSWORD {'eguser password'} SUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;* The name of this user has to be specified in the **USERNAME** text box. 5. **PASSWORD**- The password associated with the above user name (can be 'NULL'). Here, 'NULL' means that the user does not have any password. 6. **CONFIRM PASSWORD** – Confirm the **PASSWORD** (if any) by retyping it here. 7. **DBNAME** - The name of the database to connect to. The default is "postgres". 8. **SSL** - The name of this user has to be specified in the **USERNAME** text box. 9. **ELAPSED TIME** - Specify the duration (in seconds) for which a query should have executed for it to be regarded as a long running query. The default value is *10*. 10. **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.
Outputs of the test	One set of results for the target PostgreSQL server

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Number of queries: Indicates the number of queries currently executing on the database server that have been running for more time than the configured **ELAPSED TIME**.	Number	The detailed diagnosis for this measure indicates the exact queries and which user is executing the queries. This information can be very useful in identifying queries that may be candidates for optimization.

14

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 **database servers**. 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.