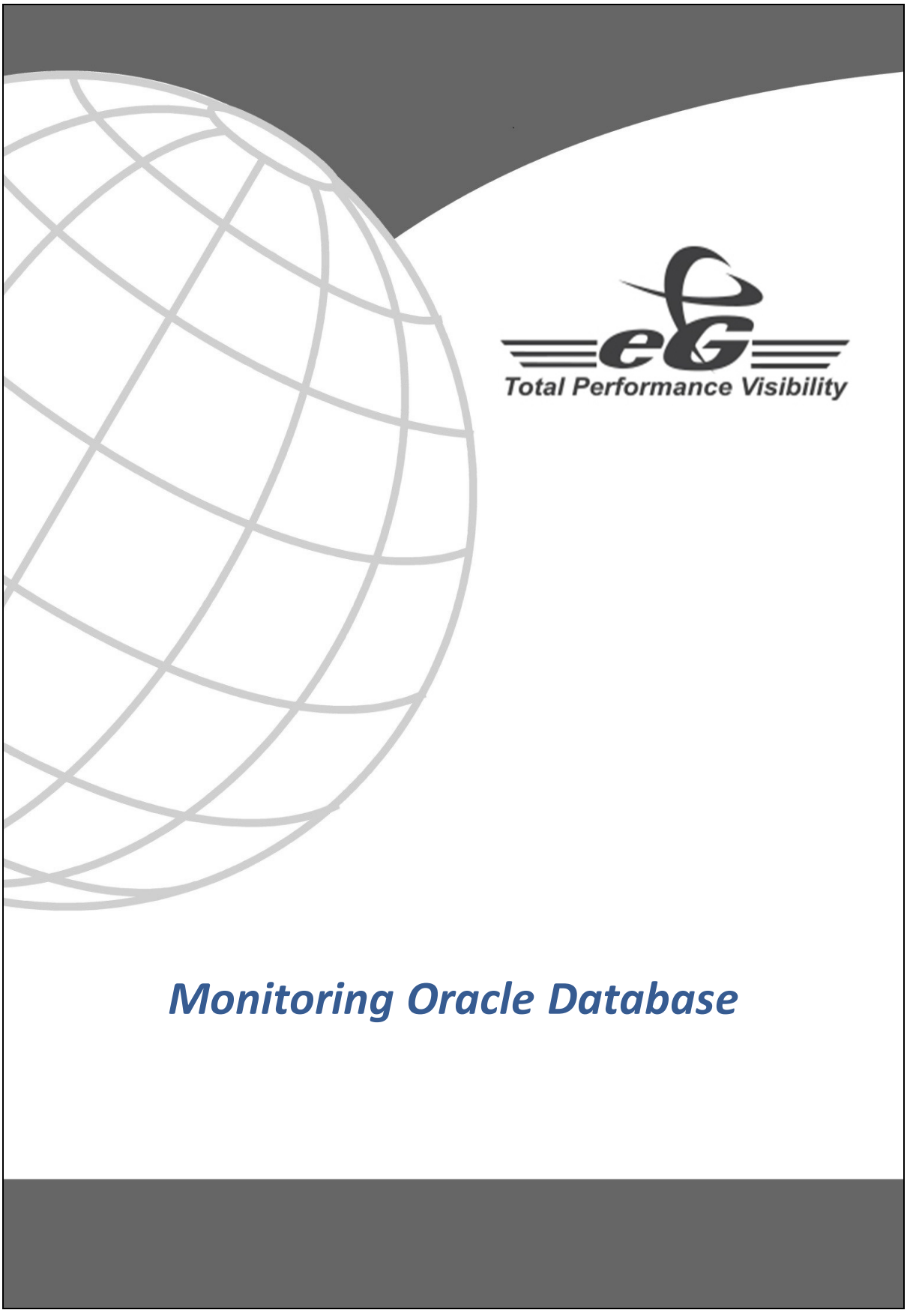


Monitoring Oracle Database

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.

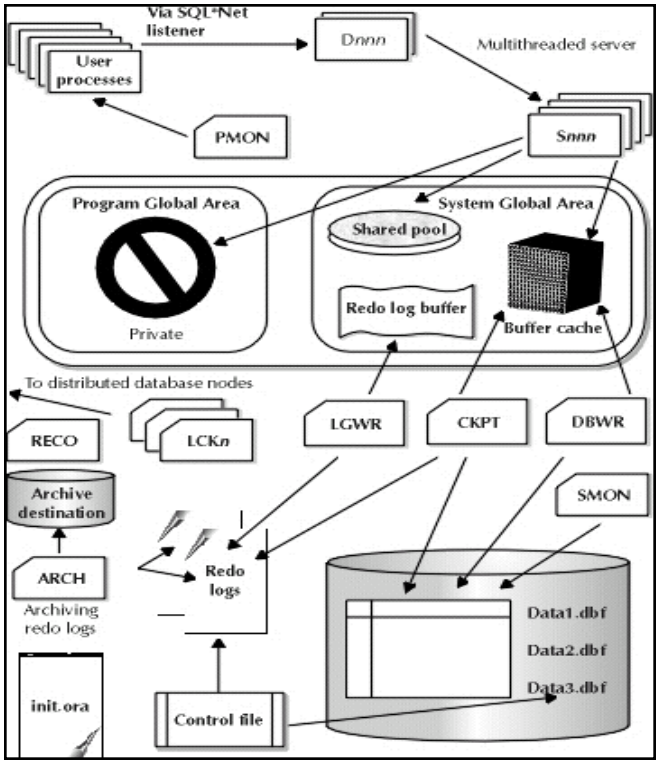


Figure 2.1: Architecture of an Oracle database server

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.

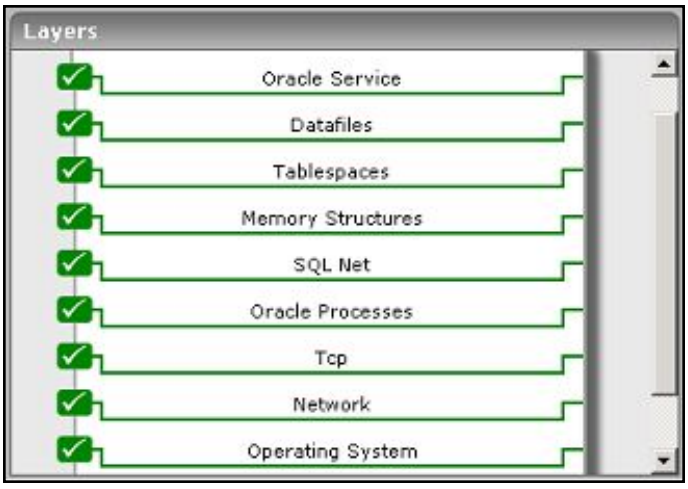


Figure 2.2: Layer model for Oracle database servers

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

This 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

Outputs of the test : One set of results for every device being accessed via the server being monitored

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. **CONTEXT** – This parameter appears only when v3 is selected as the **SNMPVERSION**. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the *SNMPEngineID* value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the

specific context (also called a *contextName*). If the **USERNAME** provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the **USERNAME** in the **CONTEXT** text box. By default, this parameter is set to *none*.

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 executed by this test should time out. The default is 10 seconds.
16. **DATA OVER TCP** - By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic - for instance, certain types of data traffic or traffic pertaining to specific components - to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the server over TCP (and not UDP). For this, set the **DATA OVER TCP** flag to **Yes**. By default, this flag is set to **No**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Current status:	This measure indicates the current status of a device that is accessible via the	Number	A value of 0 indicates that the device is operating normally. A value of 1 indicates that there is a warning

Measurement	Description	Measurement Unit	Interpretation
	server.		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.

Target of the test : A server that supports the Host Resources MIB

Agent deploying the test : A remote agent

Outputs of the test : One set of results for every storage area on the server being monitored

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. **CONTEXT** – This parameter appears only when v3 is selected as the **SNMPVERSION**. An SNMP context is a collection of management information accessible by an SNMP entity. An item of

management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the *SNMPEngineID* value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a *contextName*). If the **USERNAME** provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the **USERNAME** in the **CONTEXT** text box. By default, this parameter is set to *none*.

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 executed by this test should time out. The default is 10 seconds.
16. **DATA OVER TCP** - By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic - for instance, certain types of data traffic or traffic pertaining to specific components - to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the server over TCP (and not UDP). For this, set the **DATA OVER TCP** flag to **Yes**. By default, this flag is set to **No**.

Measurements made by the test

Measurement	Description	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.

Target of the test : A server that supports the Host Resources MIB

Agent deploying the test : A remote agent

Outputs of the test : One set of results for each server being monitored

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. **CONTEXT** – This parameter appears only when v3 is selected as the **SNMPVERSION**. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the *SNMPEngineID* value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a *contextName*). If the **USERNAME** provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the **USERNAME** in the **CONTEXT** text box. By default, this parameter is set to *none*.
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 executed by this test should time out. The default is 10 seconds.
16. **DATA OVER TCP** - By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic - for instance, certain types of data traffic or traffic pertaining to specific components - to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the server over TCP (and not UDP). For this, set the **DATA OVER TCP** flag to **Yes**. By default, this flag is set to **No**.

Measurements made by the test

Measurement	Description	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.

Target of the test : A server that supports the HOST-RESOURCES MIB

Agent deploying the test : A remote agent

Outputs of the test : One set of results for every processor on the server being monitored

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. **CONTEXT** – This parameter appears only when v3 is selected as the **SNMPVERSION**. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the *SNMPEngineID* value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a *contextName*). If the **USERNAME** provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the **USERNAME** in the **CONTEXT** text box. By default, this parameter is set to *none*.
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 executed by this test should time out. The default is 10 seconds.
16. **DATA OVER TCP** - By default, in an IT environment, all data transmission occurs over UDP. Some

environments however, may be specifically configured to offload a fraction of the data traffic - for instance, certain types of data traffic or traffic pertaining to specific components - to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the server over TCP (and not UDP). For this, set the **DATA OVER TCP** flag to **Yes**. By default, this flag is set to **No**.

Measurements made by the test

Measurement	Description	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.

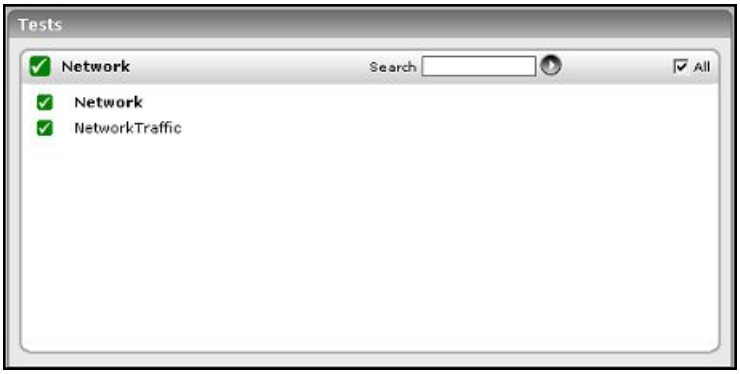


Figure 2.3: The tests associated with the Network layer

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.

Target of the test : An application being monitored

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every pattern

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. **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*:**

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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	Number	

Measurement	Description	Measurement Unit	Interpretation
	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.		
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 acknowledgment. (**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.	Bytes/sec	A high value of this measure indicates a poor network response.

Measurement	Description	Measurement Unit	Interpretation
	(**Available only for Solaris, Linux, HP-UX and AIX**)		

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Processes running:	Number of instances of a pmon, smon, lgwr, dbw, and reco processes currently executing.	Number	This value indicates if too many or too few processes corresponding to an application are executing on the host.
CPU utilization:	Total percentage CPU utilization of each process instance detected above.	Percent	A very high value could indicate that processes corresponding to the specified pattern are consuming excessive CPU resources.
Memory utilization:	The ratio of the resident set	Percent	A sudden increase in memory utilization

Measurement	Description	Measurement Unit	Interpretation
	size of a process to the physical memory of the host system on which it executes, expressed as a percentage.		for a process may be indicative of memory leaks in the application.

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**.

Target of the test : An Oracle server on Unix

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Oracle database server monitored.

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.

Measurements made by the test

Measurement	Description	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.

Measurement	Description	Measurement Unit	Interpretation
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

This 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.

Target of the test : A server that supports the Host Resources MIB

Agent deploying the test : A remote agent

Outputs of the test : One set of results for every configured process pattern

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. **CONTEXT** – This parameter appears only when v3 is selected as the **SNMPVERSION**. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the *SNMPEngineID* value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a *contextName*). If the **USERNAME** provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the **USERNAME** in the **CONTEXT** text box. By default, this parameter is set to *none*.
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 executed by this test should time out. The default is 10 seconds.
16. **DATA OVER TCP** - By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic - for instance, certain types of data traffic or traffic pertaining to specific components - to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the server over TCP (and not UDP). For this, set the **DATA OVER TCP** flag to **Yes**. By default, this flag is set to **No**.
17. **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*”.

18. **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.

Measurements made by the test

Measurement	Description	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.

Measurement	Description	Measurement Unit	Interpretation
			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.

Target of the test : An Oracle server (9i and 10g)

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the *session* and *process* resources allocated to the Oracle server being monitored

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Initial allocation:	Indicates the number of sessions/processes allocated at the time of creating the Oracle database instance.	Number	
Current utilization :	This measure indicates the number of sessions/processes currently active on the Oracle database server.	Number	If the value of the **Current 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

Measurement	Description	Measurement Unit	Interpretation
			be changed using the **SESSIONS** and **PROCESSES** parameters in the init database parameter file. The **Configured limit** measure of this test reports this new configuration only. For most resources, the **INITIAL_ ALLOCATION** value and the **Configured 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 signifies the maximum number of sessions and processes the database can handle, given its current memory capacity. The **Maximum utilization limit** measure reports this Oracle-recommended value. If the value of the **Current utilization** measure, exceeds the **Maximum utilization limit**, then the performance of your database is bound to deteriorate. Therefore, ensure that the Config_Limit is always well within the **Maximum 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	Number	

Measurement	Description	Measurement Unit	Interpretation
	configured to handle.		
Percentage utilized:	Indicates the percentage of the configured number of sessions/processes (i.e., the **Configured 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.

Target of the test : An Oracle server 10g

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Oracle server.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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;

Measurement	Description	Measurement Unit	Interpretation
			• 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.

Note:

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))
```

Target of the test : An Oracle 10g server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each instance of an Oracle server

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. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Oracle Instance availability:	This measure indicates whether this instance is available or not.	Percent	The value 100 denotes that the instance is available, and the value 0 indicates that the instance is unavailable. The detailed diagnosis of this measure displays information about successful connectivity attempts and error messages reported when the Oracle instances are unavailable. Use the detailed diagnosis of this measure to know number of successful and failed attempts
Uptime :	Indicates how long the instance has been up and running.	Seconds	
Uptime since last measure :	Indicates the duration for which the instance has been up since the last measurement period.	Seconds	If the value of this measure is lesser than the test frequency, it indicates that the instance was rebooted during the last measurement period.
Is rebooted?	Indicates whether this instance was rebooted or not.		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>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<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.

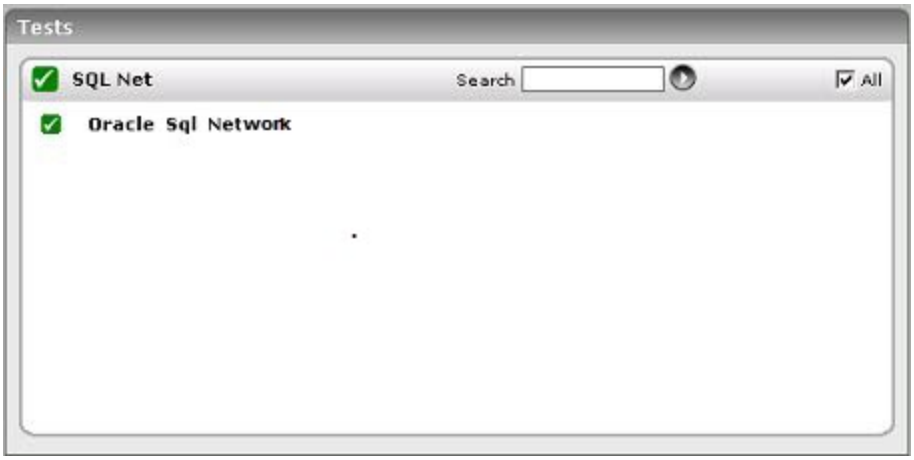


Figure 2.5: The tests associated with the SQL Net layer

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.

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.

Outputs of the test : One set of results for every SID (instance) monitored.

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 manually creating a user on an Oracle database server before version 10G, ensure that the *select_catalog_role* and *create session* privileges are granted to the user. When monitoring an Oracle database server 10G (and above), this test additionally requires *select* privileges to the *sys.user$* table.

   The sample script we recommend for user creation (in Oracle database server versions before 10G) 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 version 10G and 11G) 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 select sys.user$ 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. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Oracle server availability:	Whether the database instance is responding to requests.	Percent	The availability is 100% when the instance is responding to a request and 0% when it

Measurement	Description	Measurement Unit	Interpretation
			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 **Oracle SQL Network** test, this measure maps to the **Oracle Service** layer.
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.	Secs	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.
Data transmit rate:	The rate of data being transmitted by the server	KB/Sec	The data transmission rate reflects the workload on the server.

Measurement	Description	Measurement Unit	Interpretation
	in response to client requests.		
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

Measurement	Description	Measurement Unit	Interpretation
			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).

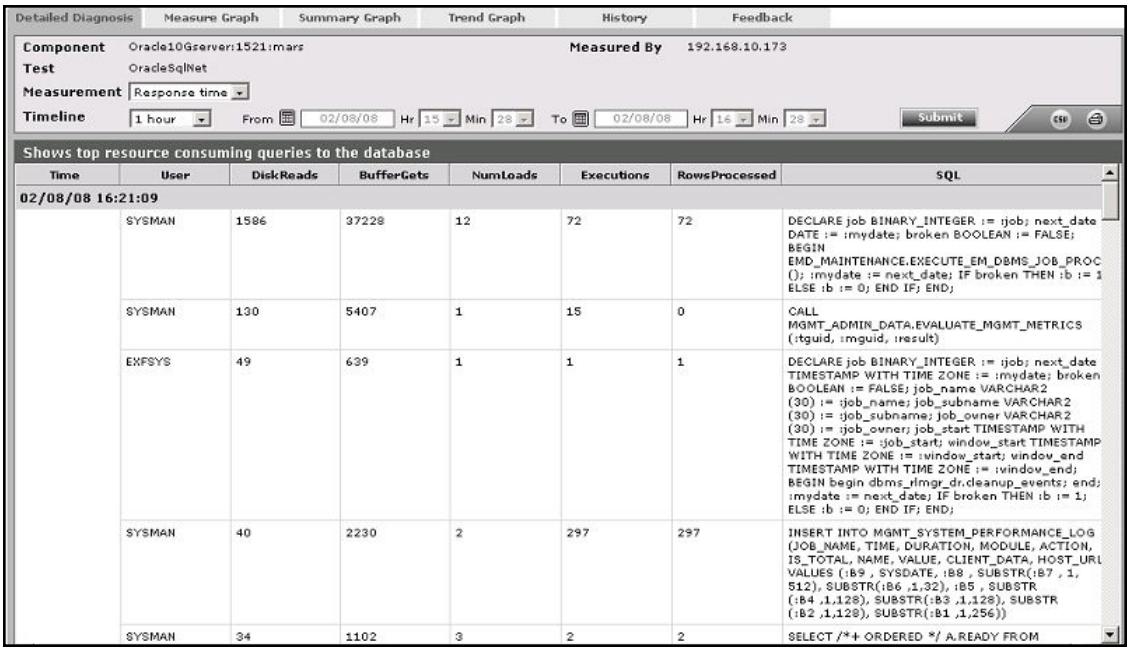


Figure 2.6: Detailed diagnosis of the Response time measure displaying the top 10 resource consuming queries

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).

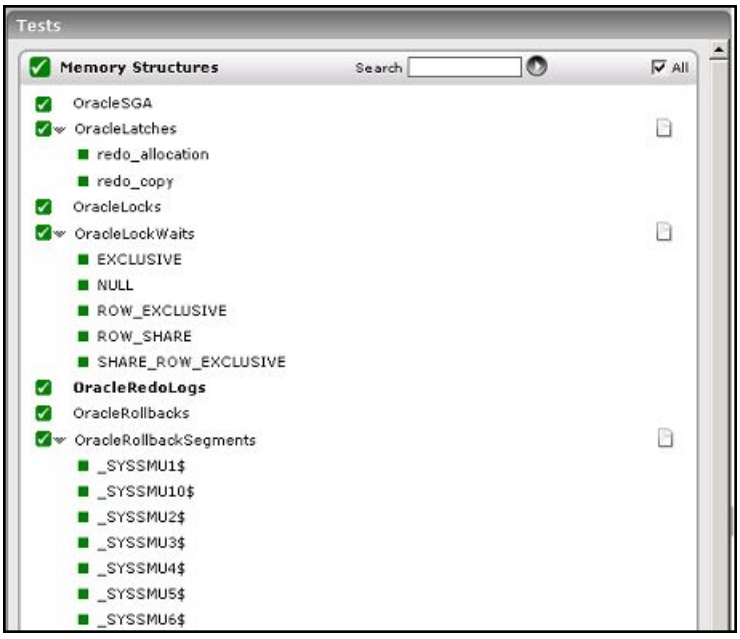


Figure 2.7: Tests mapping to the Memory Structures layer

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 Oracle Rollback Segments** test measures the load and the efficiency of the rollback segments.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Writes:	Indicates the total number of writes to a specific rollback segment during the last measurement period.	Number	This gives you an indication of the load on a specific rollback segment.

Measurement	Description	Measurement Unit	Interpretation
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 **Oracle Redo Logs** test monitors key performance metrics pertaining to the redo log buffer in an Oracle server instance.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored

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. **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.

Measurements made by the test

Measurement	Description	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	Number	

Measurement	Description	Measurement Unit	Interpretation
	copied into the redo log buffer. (ie. The number of attempts to allocate space in the redo)		
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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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:

a. Enough latches are not available

b. 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 Latches test is used to monitor latches in an Oracle database.

Target of the test : An Oracle server (9i and 10g)

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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

Measurement	Description	Measurement Unit	Interpretation
			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 **Oracle SGA** test collects a variety of statistics relating to the various SGA components.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Library cache hitratio:	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*

Measurement	Description	Measurement Unit	Interpretation
			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

Measurement	Description	Measurement Unit	Interpretation
			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

Measurement	Description	Measurement Unit	Interpretation
			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.

Measurement	Description	Measurement Unit	Interpretation
			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 calculated 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

Measurement	Description	Measurement Unit	Interpretation
			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 in 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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Current size:	Indicates the amount of PGA memory that is currently in use.	MB	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.
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 in*it 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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	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, 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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Locks:	Gives the total number of locks that are held.	Number	A high value may indicate one of the following: a. Too many transactions happening b. Locked resources not being released properly c. Locks are being held unnecessarily.
Avg lock time:	Indicates the time for which these locks are held.	Secs	A high value may indicate one of the following: a. Locked resources not being released properly b. Locks are being held unnecessarily.

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 **Oracle Lock Waits** test identifies the sessions that are waiting for acquiring a lock. The measures made by this test are as follows:

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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. **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. **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.

    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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Lock waits:	Indicates the number of lock waits.	Number	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.
Avg lock wait time:	Indicates the duration for which sessions were waiting for this lock.	Secs	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.

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.

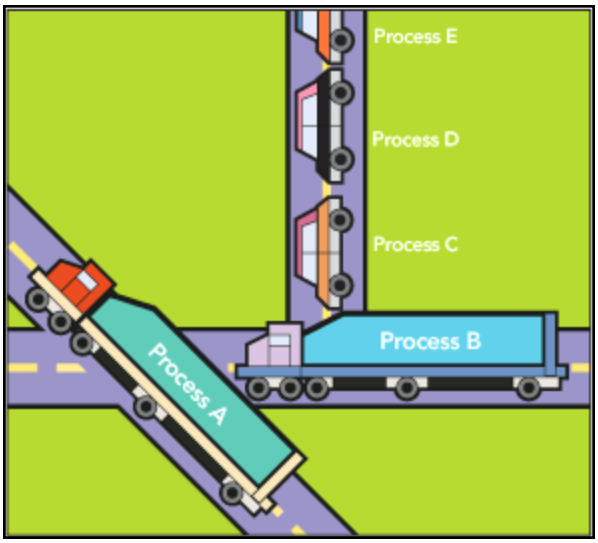


Figure 2.8: The traffic jam analogy representing blocking

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of blockers:	Indicates the number of blocker processes.	Number	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.

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 Oracle Table Spaces test.

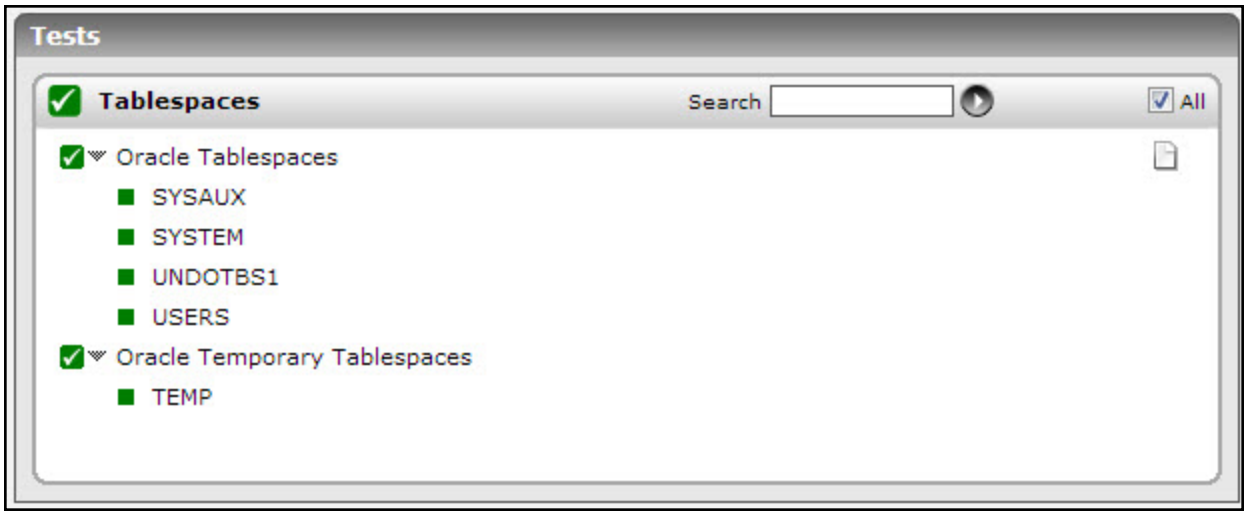


Figure 2.9: Tests mapping to the Tablespaces layer

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Current usage:	Indicates the current actual usage with respect to the	Percent	As a rule of thumb, at any time, about

Measurement	Description	Measurement Unit	Interpretation
	current allocated size (Current_size)		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.
Physical reads:	Indicates the rate of physical reads happening on a tablespace.	Reads/Sec	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.
Physical writes:	Indicates the rate of physical writes happening on a tablespace.	Writes/Sec	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.
Auto extensible:	Indicates whether the tablespace has the capability to grow automatically or not		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>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<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>
<tr><td>Max size:</td><td>Indicates the maximum extent (in MB) upto which a tablespace can grow</td><td>MB</td><td></td></tr>
<tr><td>Current size:</td><td>Indicates the current allocated size of the tablespace</td><td>MB</td><td>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.</td></tr>
<tr><td>Free space:</td><td>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</td><td>MB</td><td>If this value is very low, then it indicates over-utilization of the tablespace.</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
	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.		
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?

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	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.	Percentage	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**

Measurement	Description	Measurement Unit	Interpretation
			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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	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.

Measurement	Description	Measurement Unit	Interpretation
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 Growth** test 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 Tablespace Status 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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Online status:	Indicates the current state of this tablespace.	Percent	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.

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.

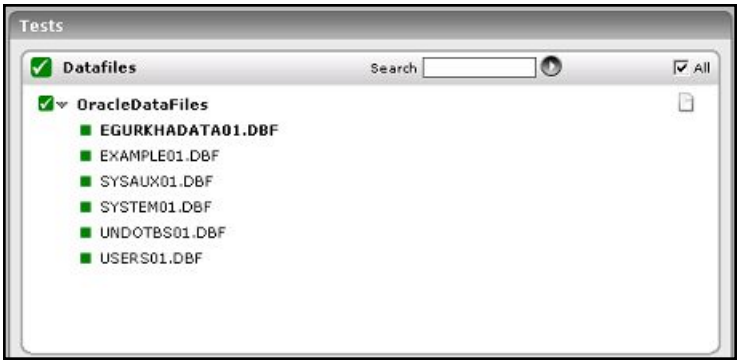


Figure 2.10: Tests mapping to the Datafiles layer

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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. **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**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Oracle server being monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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. This test 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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every datafile monitored.

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.

Measurements made by the test

Measurement	Description	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.

Target of the test : An Oracle 10g server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every datafile on the Oracle server.

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. **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**.

Measurements made by the test

Measurement	Description	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.

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every datafile on the Oracle server.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
File status:	Indicates the current status of this datafile.		The table below indicates the values that this measure can report and their corresponding numeric equivalents: **Numeric Value** / **Measure Value** 1 / System 2 / Online 3 / Recover 4 / Unknown 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 Value**s 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.

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td>File access mode:</td><td>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.

Target of the test : An Oracle 12c server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every datafile on the Oracle server.

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. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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.

Measurement	Description	Measurement Unit	Interpretation
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.

Measurement	Description	Measurement Unit	Interpretation
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.

Target of the test : An Oracle 12c server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for *deadprocessaddress_deadsessionaddress* on the Oracle instance monitored.

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. **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.

Measurements made by the test

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td>Process state:</td><td>Indicates the current cleanup state of this process.</td><td></td><td>The values that this measure can report and their corresponding numeric values have been discussed hereunder:
<table>
<tr><th>Measure Value</th><th>Description</th><th>Numeric Value</th></tr>
<tr><td>unsafe to attempt</td><td>Occurs for a killed session that has not been moved, so no cleanup can occur on it yet</td><td>1</td></tr>
<tr><td>cleanup pending</td><td>Occurs for a dead process / killed session that can be cleaned up, but PMON has not yet made an attempt</td><td>2</td></tr>
<tr><td>resources freed</td><td>Occurs for a</td><td>3</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Measure Value</th><th>Description</th><th>Numeric Value</th></tr>
<tr><td></td><td>dead process / killed session where all children have been freed, but the process / killed session itself is not yet freed</td><td></td></tr>
<tr><td>resources freed – pending ack</td><td>- Occurs for a killed session where all children have been freed, but the session itself cannot be freed until the owner has acknowledged it</td><td>4</td></tr>
<tr><td>partial cleanup</td><td>Occurs if some of the children have been cleaned up</td><td>5</td></tr>
</table>
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.
</td></tr>
<tr><td>Dead time:</td><td>Indicates how long it has</td><td>Secs</td><td>A consistent increase in the value of this</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
	been since this process was marked dead or this session was marked killed.		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.

Target of the test : An Oracle 12c server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every I/O function that is taking more than 500 milliseconds to complete.

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. **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.

Measurements made by the test

Measurement	Description	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.

Target of the test : An Oracle 12c server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every file other than datafiles and temp files on the Oracle server.

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. **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.

Measurements made by the test

Measurement	Description	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

Measurement	Description	Measurement Unit	Interpretation
			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

Measurement	Description	Measurement Unit	Interpretation
			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.

Target of the test : An Oracle 12c server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every PDB on the Oracle server.

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. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Status:	Indicates the current status of this PDB.		The values that this measure can report and their corresponding numeric values are discussed hereunder:

Measure Value	Description	Numeric Value
new	The PDB has never been opened since it was created. It must be opened inREAD	1

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Measure Value</th><th>Description</th><th>Numeric Value</th></tr>
<tr><td></td><td>WRITE mode for Oracle to perform processing needed to complete the integration of the PDB into the CDB and mark it NORMAL. An error will be thrown if an attempt is made to open the PDB read only</td><td></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 DROP PLUGGABLE DATABASE.</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</td><td>4</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Measure Value</th><th>Description</th><th>Numeric Value</th></tr>
<tr><td></td><td>plugged</td><td></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>
<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</td><td>6</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Measure Value</th><th>Description</th><th>Numeric Value</th></tr>
<tr><td></td><td>there was an error during PDB creation.</td><td></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>Mode:</td><td>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?

Target of the test : An Oracle 12c server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every temp file on the Oracle server.

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. **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.

Measurements made by the test

Measurement	Description	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.

Measurement	Description	Measurement Unit	Interpretation
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

Measurement	Description	Measurement Unit	Interpretation
			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).

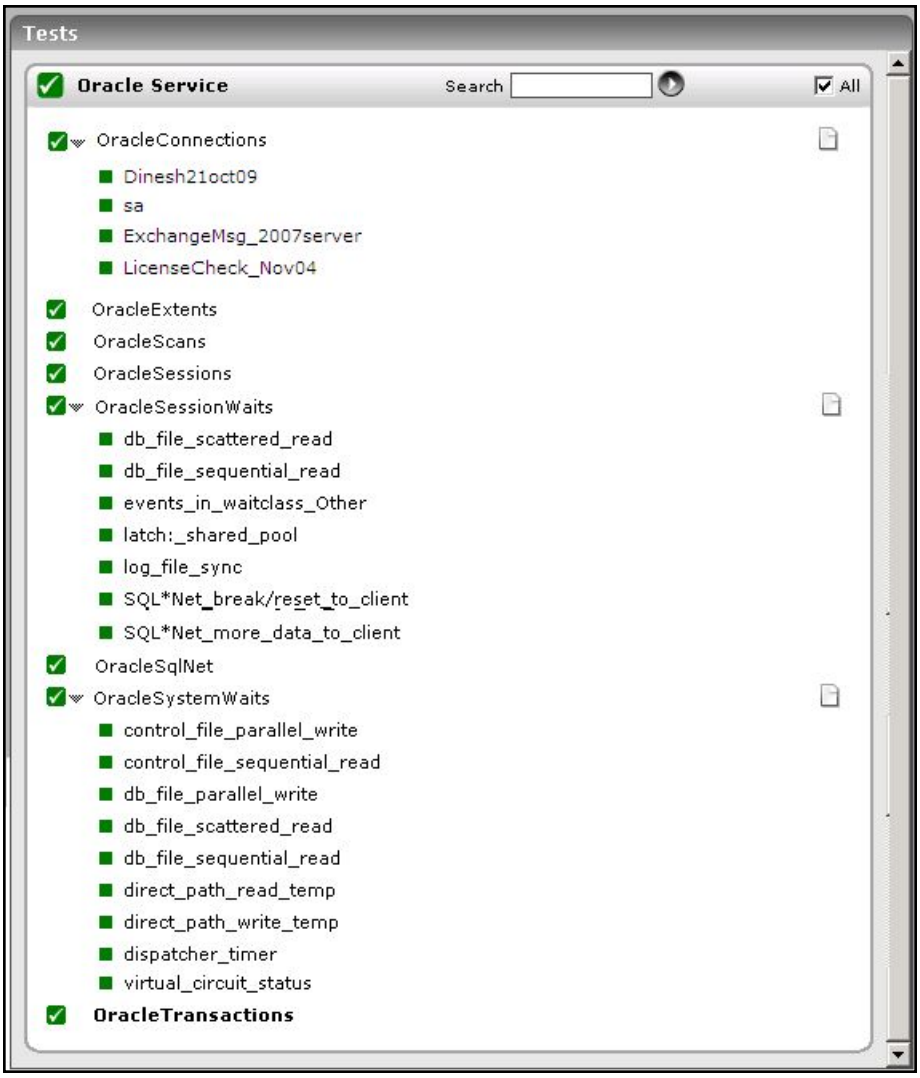


Figure 2.11: Tests mapping to the Oracle Service layer

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every user who is currently connected to the Oracle server.

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. **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.
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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total connections:	Indicates the total number of connections currently established by this user on the server.	Number	
Active connections:	Indicates the number of connections of this user that are currently active.	Number	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.
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,

Measurement	Description	Measurement Unit	Interpretation
			you can understand how each of the idle connections were made - i.e., using which program - and from where - i.e., from which host.
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

Measurement	Description	Measurement Unit	Interpretation
			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	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure,

Measurement	Description	Measurement Unit	Interpretation
	idle time while waiting for a client's response.		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 **Oracle Extents** 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. 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.

Note:

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.

Target of the test : An Oracle 12c server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Oracle server.

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. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Cannot extend objects:	This measure indicates the number of objects that cannot extend any further by acquiring new extents.	Number	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.
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.	Number	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.

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every session wait event monitored on the Oracle server.

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. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
New 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 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.
Total waits timedout:	Indicates the total number	Number	A large number of timed out wait events

Measurement	Description	Measurement Unit	Interpretation
	of waits on this event that timed out since the last measurement period.		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	

2.9.4 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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every system wait event monitored on the Oracle server.

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.

Measurements made by the test

Measurement	Description	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

Measurement	Description	Measurement Unit	Interpretation
			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 **Oracle Sessions** test is used by an eG agent to track user activity related to a database server instance.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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. **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.

Measurements made by the test

Measurement	Description	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

Measurement	Description	Measurement Unit	Interpretation
			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.

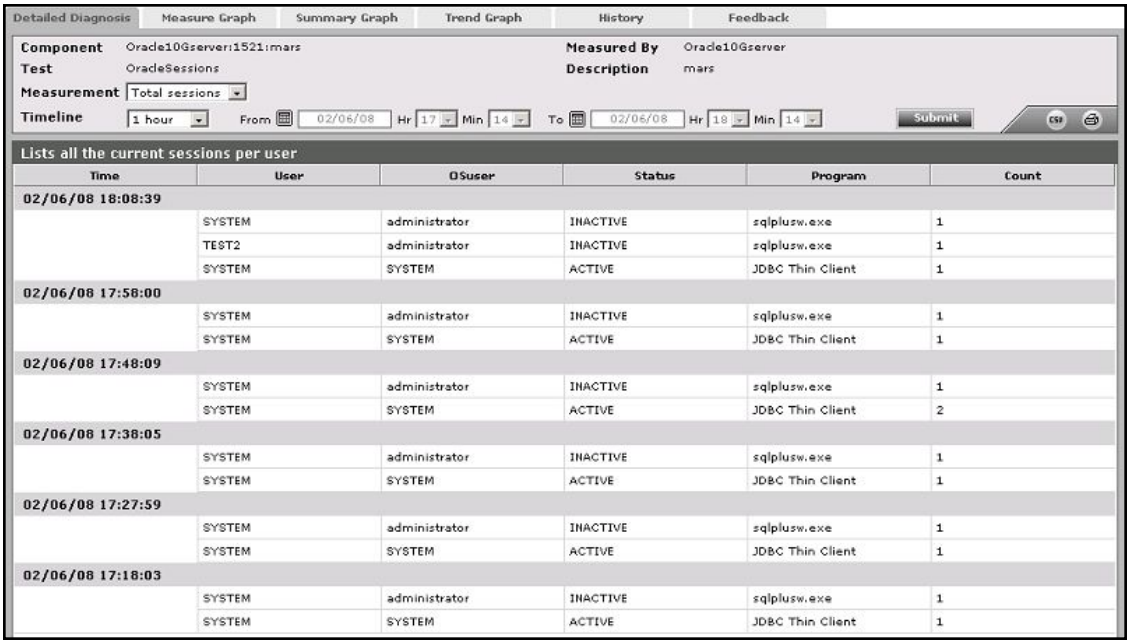


Figure 2.12: The detailed diagnosis of the Total sessions measure

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

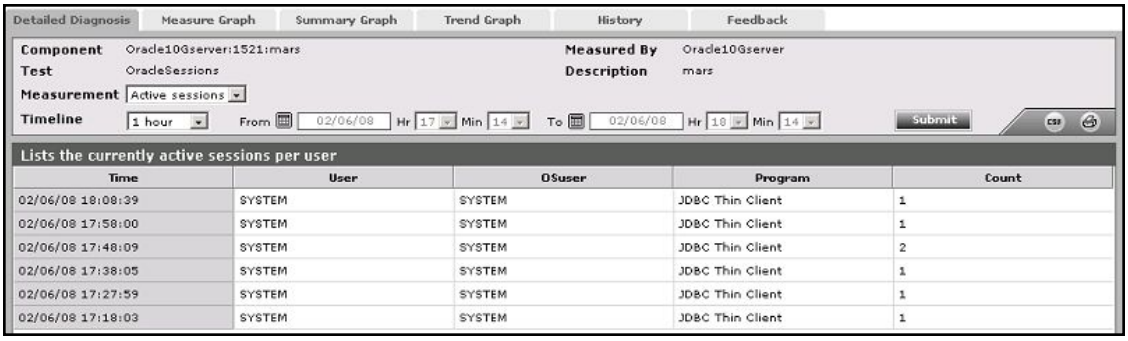


Figure 2.13: The detailed diagnosis of the Active sessions measure

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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

Measurement	Description	Measurement Unit	Interpretation
			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.

Measurement	Description	Measurement Unit	Interpretation
			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.

Note:

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

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every cluster-related session wait event monitored on the Oracle server.

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. **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.

Measurements made by the test

Measurement	Description	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.

Note:

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.

Target of the test : An Oracle server in an RAC environment

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every system wait event monitored on the Oracle server.

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 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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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	Indicates the total number	Number	A large number of timed out wait events

Measurement	Description	Measurement Unit	Interpretation
timedout:	of waits on this event that timed out since the last measurement period.		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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every user account monitored.

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. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Invalid objects:	The number of invalid	Number	The detailed diagnosis capability for this

Measurement	Description	Measurement Unit	Interpretation
	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.		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.

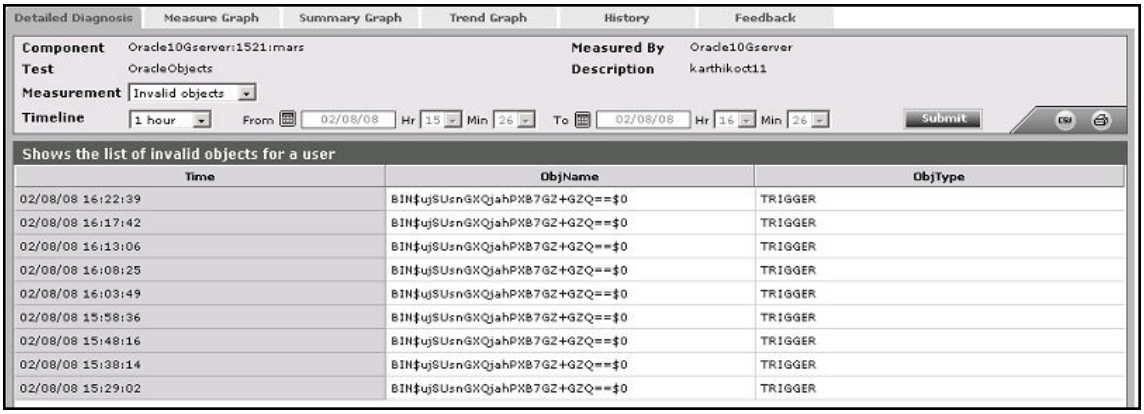


Figure 2.14: The detailed diagnosis of the Invalid objects measure

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every instance of an Oracle database.

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. **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.

Measurements made by the test

Measurement	Description	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.

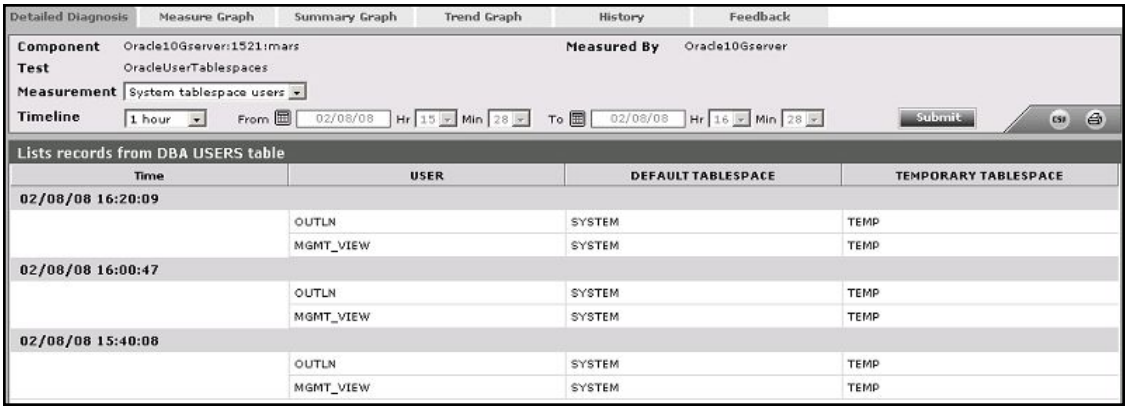


Figure 2.15: The detailed diagnosis of the System tablespace users measure

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.

Note:

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every instance of an Oracle database.

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. **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.

Measurements made by the test

Measurement	Description	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.

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
User commits:	The number of user commits that have happened during the last measurement period	Number	
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

Measurement	Description	Measurement Unit	Interpretation
			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, 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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every instance of an Oracle database.

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. **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.

Measurements made by the test

Measurement	Description	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.

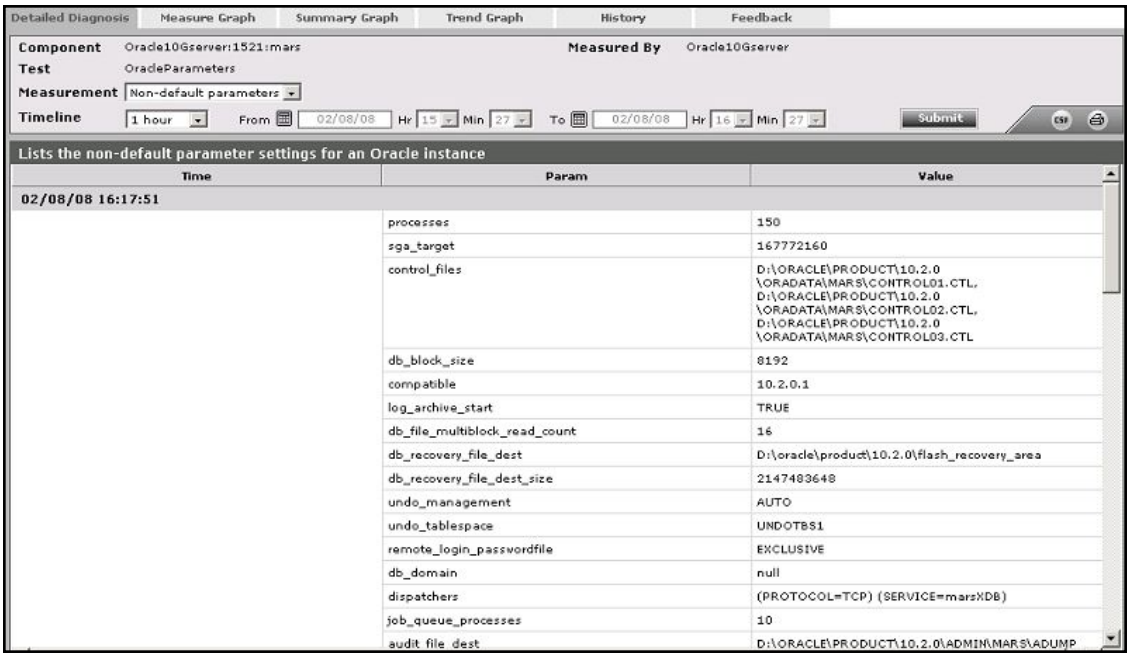


Figure 2.16: The detailed diagnosis of the Non-default parameters measure

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every instance of an Oracle database.

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, and select privilege to

   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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every alert log file.

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. **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.

   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. **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.
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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Recent errors:	Indicates the number of new errors that have been written by oracle to the alert log file.	Number	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.
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.**

Measurement	Description	Measurement Unit	Interpretation
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 **Idle Oracle Sessions** test 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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every instance of an Oracle database.

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 enable/disable the detailed diagnosis capability will be available only if the

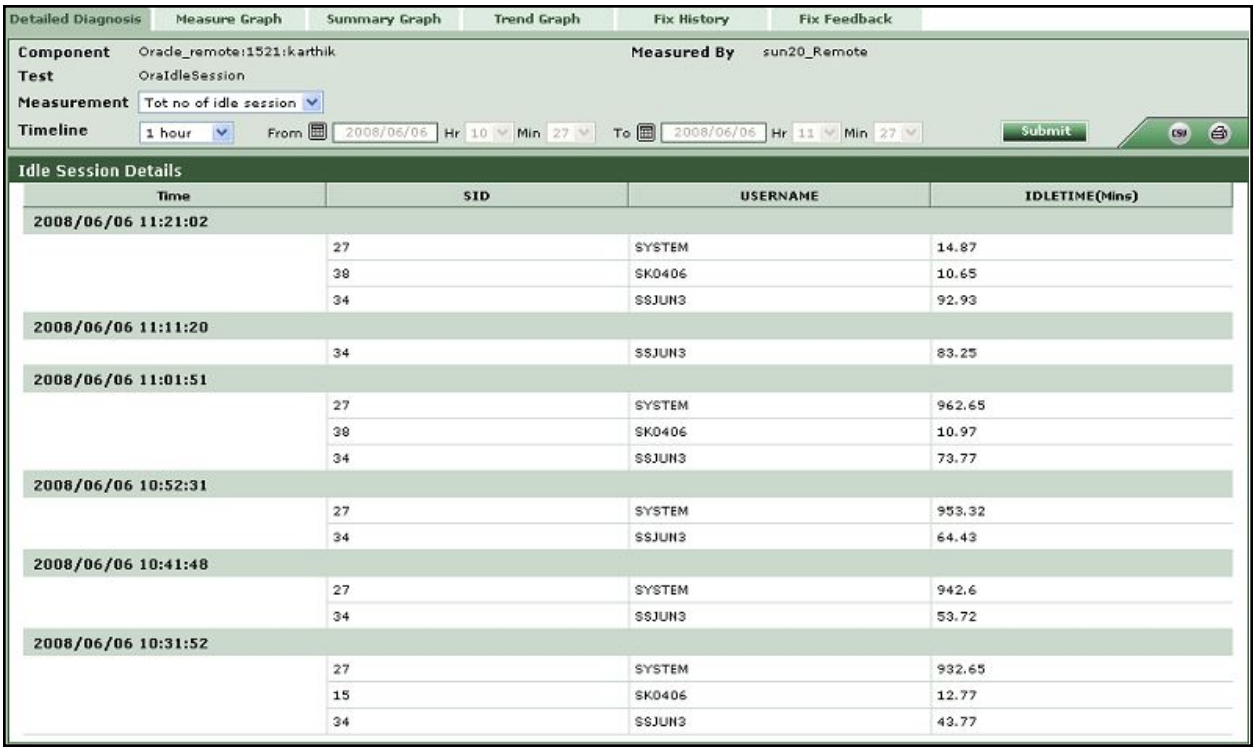
following conditions are fulfilled:

- 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.

Measurements made by the test

Measurement	Description	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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every instance of an Oracle database.

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.**

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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**.	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.

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 **Oracle Dump Area** 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.

Note:

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

Target of the test : An Oracle database server (9i and 10g)

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every **DUMPFILE** that is auto-discovered.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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	Percent	This measurement value should ideally

Measurement	Description	Measurement Unit	Interpretation
	is currently unused.		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 **Oracle FlashArea Usage** 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.

Note:

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

Target of the test : An Oracle database server 10g

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the flash recovery area on the Oracle server.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Flash area usage:	Indicates the space currently occupied by the flash recovery files.	MB	
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,

Measurement	Description	Measurement Unit	Interpretation
			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.

Note:

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.

Target of the test : An Oracle server (9i and 10g)

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every *username.objectname* configured.

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. **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.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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

Measurement	Description	Measurement Unit	Interpretation
			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	Number	Buffer busy waits occur within Oracle when a task goes to fetch a data block,

Measurement	Description	Measurement Unit	Interpretation
	that have occurred during the last measurement period.		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

Measurement	Description	Measurement Unit	Interpretation
			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.

Target of the test : An Oracle 10g database server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every wait event on the Oracle server being monitored.

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. **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.

Measurements made by the test

Measurement	Description	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 the query 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.

Target of the test : An Oracle 10g server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every wait class on the Oracle server monitored.

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. **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.

Measurements made by the test

Measurement	Description	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.

Target of the test : An Oracle 10g database server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every wait class generating one/more SQL wait events on the Oracle server.

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. **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.

Measurements made by the test

Measurement	Description	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.

Target of the test : An Oracle 10g server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every system wait class on the Oracle server.

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. **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.

Measurements made by the test

Measurement	Description	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 **Oracle Archive Area** 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.

Note:

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

Target of the test : An Oracle 10g database server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the **ARCHIVELOGFILE** configured/auto-discovered.

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. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Used archive area:	Indicates the space currently occupied by the archive files in the archive destination directory.	MB	

Measurement	Description	Measurement Unit	Interpretation
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

Measurement	Description	Measurement Unit	Interpretation
			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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Oracle server.

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. **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.

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.

Measurements made by the test

Measurement	Description	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	Number	Ideally, this value should be low. If the value is high, you may want to check

Measurement	Description	Measurement Unit	Interpretation
	last measurement period, but with warnings.		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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every *DisplayName* configured for the **OBJECT NAME** parameter of this test.

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* 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. **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. **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.

Measurements made by the test

Measurement	Description	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

Measurement	Description	Measurement Unit	Interpretation
			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 Oracle fragmented 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

Measurement	Description	Measurement Unit	Interpretation
			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.

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Oracle server.

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. **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.

Measurements made by the test

Measurement	Description	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

Measurement	Description	Measurement Unit	Interpretation
			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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every database on the Oracle server.

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. **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.

Measurements made by the test

Measurement	Description	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

Measurement	Description	Measurement Unit	Interpretation
			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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every database on the Oracle server.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle server being monitored.

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. **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.

Measurements made by the test

Measurement	Description	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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle server being monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle server being monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle server being monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Defer transaction count:	Indicates the number of deferred transactions in the Oracle database.	Number	

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle server being monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle server being monitored.

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. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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.

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>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.</td></tr>
<tr><td>Is listener logging enabled?:</td><td>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></td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>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>Listener status:</td><td>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 indicate 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>Uptime of the listener:</td><td>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.

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.

Target of the test : An Oracle 10g database server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each *DiskGroup:Disk* pair on the Oracle server being monitored.

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.

Measurements made by the test

Measurement	Description	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.	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.

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each *DiskGroup:Disk* pair on the Oracle server being monitored.

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. **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.

Measurements made by the test

Measurement	Description	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.

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.

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle database server being monitored.

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. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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.

Target of the test : An Oracle 10g database server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle database server being monitored.

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. **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.

Measurements made by the test

Measurement	Description	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	Percent	A high value indicates that one/more Oracle sessions are consuming memory excessively. Use the detailed

Measurement	Description	Measurement Unit	Interpretation
	sessions of this Oracle server.		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')

Wait Class	Description
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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each wait class active on the Oracle database server being monitored.

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. **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.

Measurements made by the test

Measurement	Description	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.

Measurement	Description	Measurement Unit	Interpretation
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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each user with one/more sessions that have been open for over the configured **USER LOGIN TIME**.

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**.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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

Measurement	Description	Measurement Unit	Interpretation
			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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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	

Measurement	Description	Measurement Unit	Interpretation
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

Measurement	Description	Measurement Unit	Interpretation
			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

Measurement	Description	Measurement Unit	Interpretation
			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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored..

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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 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/Trans	Parsing is one stage in the processing of a

Measurement	Description	Measurement Unit	Interpretation
	parses executed by the server per transaction.		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

Measurement	Description	Measurement Unit	Interpretation
			• 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

Measurement	Description	Measurement Unit	Interpretation
			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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every SID monitored.

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. **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**.

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.

Measurements made by the test

Measurement	Description	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.	Memorybuffergets / 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	Secs/execution	If the value of this measure crosses 10

Measurement	Description	Measurement Unit	Interpretation
	query.		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.

2.9.46 Oracle Large Table Test

Larger tables in an Oracle database server may be prone to fragmentation which may reduce the performance of the database as well as result in longer query times when a full scan is performed. To improve the overall performance of the database server and optimize the query times, it is essential to check whether the tables are indexed appropriately and if partitioning of the tables are necessary. This check when performed on the larger tables, will automatically improve the query response time thus providing a better overall performance. Prior to performing this check, the administrators should identify the tables that are large and are more resource intensive on their database. The **Oracle Large Table** test helps the administrators to determine such large tables.

This test tracks the tables in an Oracle database server and reports the number of tables that are larger than the configured limit.

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.

Target of the test : An Oracle server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every instance of an Oracle database.

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. **TABLE SIZE GB** - Specify the size limit (in GB) for the tables against the **TABLE SIZE GB** text box beyond which the tables are termed as large tables. The default value is 5.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Large Tables:	Indicates the number of tables in the database server that exceed the limit configured against the **TABLE SIZE GB** parameter	Number	The detailed diagnosis for this measure reveals the details of the tables that are large in size.

3

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 **Oracle 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.