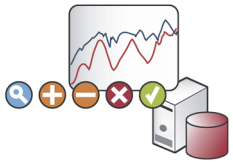
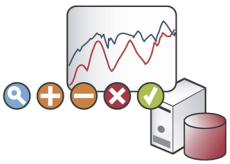
Vitness User's Guide

Witness



VERSANT

Witness



Vitness User's Guide:

VERSANT

Versant Object Database™

Copyright © 2001-2007 Versant Corporation. All rights reserved.

The software described in this document is subject to change without notice. This document does not represent a commitment on the part of Versant. The software is furnished under a license agreement or nondisclosure agreement and may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license or nondisclosure agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or for any purpose without the express written permission of Versant.

Versant and FastObjects are either registered trademarks or trademarks of Versant Corporation in the United States and/or other countries.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Versant is independent of Sun Microsystems, Inc.

Microsoft, Windows, Visual C#, Visual Basic, Visual J#, and ActiveX are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

All other products are a registered trademark or trademark of their respective company in the United States and/or other countries.

Table of Contents

List of Figures

List of Tables

List of Examples

Contact Information for Versant

You can obtain the latest information about Versant products by contacting either of our main office locations, visiting our web sites, or sending us email.

Versant Office Locations

Versant Corporation is headquartered in Redwood City, California. Versant GmbH is responsible for European operations and has headquarters in Hamburg, Germany.

Versant Corporation

255 Shoreline Drive
Redwood City, CA 94065
USA

1-800-VERSANT
+1.510.789.1500
+1.510.789.1515 (FAX)

Versant GmbH

Wiesenkamp 22b
D-22359 Hamburg
Germany

+49 (0)40 609 90 0
+49 (0)40 609 90 113 (FAX)

Versant Web Sites

For the latest corporate and product news, please visit the Versant web site at http://www.versant.com/.

The Versant Developer Center provides all the essential information and valuable resources needed for developers using Versant Object Database or Versant FastObjects including trial software and the Versant Developer forums. Visit the Versant Developer Center at http://developer.versant.com/.

Versant Email Addresses

For inquiries about about Versant products and services, send email to:

<mailto:info@versant.com>

For help in using Versant products, contact Technical Support at:

<mailto:support@versant.com> (Customer Services US)
or
<mailto:support@versant.net> (Customer Services Europe)

Please send feedback regarding this guide to:

<mailto:documentation@versant.com>

Chapter 1. Introduction

Enterprise production systems need to maintain a guaranteed service level with availability requirements of 24x7 and 99.999% being more and more common. In such environments, a database typically serves as the backbone of the deployed applications and that means that database availability is the key to the success of the whole system. System monitoring and management is critical in such situations. Vitness is a complete administration environment for monitoring and managing Versant Object Database databases, whose aim is to fulfill such needs.

Vitness provides detailed monitoring capabilities that can be used to understand what is going on “under the hood” of a production system. Discovering a potential problem before it becomes a real one is crucial to preventing faults that could be unexpectedly generated by an unobserved system. And regular monitoring of database systems can suggest actions that help improve overall system performance. It is also important in determining future needs and that helps the support and management decision teams.

In fact, the observation of a system is a cornerstone to the proper establishment of an adequate System & Network Management (S&NM) policy as it is intended and needed in large enterprises. In this context, Vitness provides remote management capabilities that make it a fundamental tool for all database administration operations and decisions.

1.1. Product Overview

At its core, Vitness is a tool for the monitoring of Versant Object Database databases. But Vitness is also an expandable framework for additional administrative and management components such as license management, database compaction and more.

And Vitness can keep itself up-to-date. The built-in update mechanism will download and install any improvements or new components as they are introduced.

1.1.1. Database Monitoring

The Vitness Database Monitor component is designed following the standard agent/monitor paradigm. The remote agent resides on the Versant Object Database server system, while the monitor may run anywhere to present a graphical display of the ongoing activity of the monitored database.

The architecture of Vitness is summarized in the following figure.

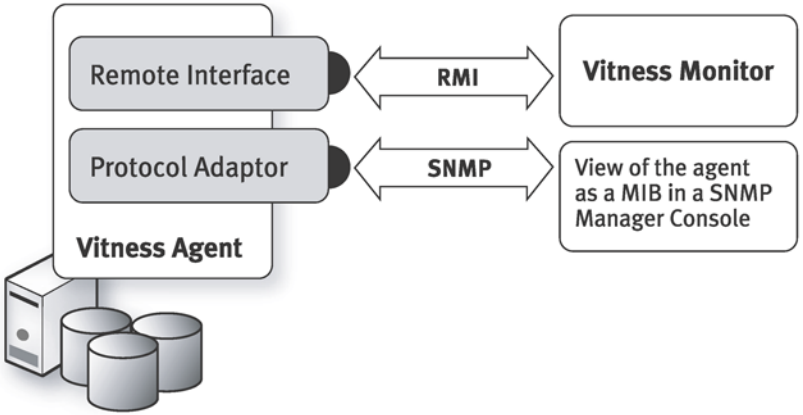


Figure 1.1. VOD Agent Architecture

The VOD Agent is deployed on the server where the databases to be monitored are deployed. The VOD Agent does not open any connections with the monitored databases so no overhead is imposed by monitoring the databases.

Vitness uses an RMI distributed architecture for the communication between agents and monitors. The VOD Agent is a pure Java RMI server. The Vitness application (and the Vitness Database Monitor) is an Eclipse-based RCP application that acts as a Versant Object Database JVI client and makes a limited use of Versant Object Database APIs.

The VOD Agent supports multiple protocol adaptors. It can be deployed as an SNMP standard agent allowing for a straightforward integration in managed environments. Access to the VOD Agent is thus available through the standard SNMP protocol and event notification is handled through the trap mechanism. With SNMP support enabled the VOD Agent can be run standalone with no need of a Vitness Database Monitor.

The main features of the Vitness monitoring component are listed below.

- Multiple simultaneous database monitoring
- Dynamic discovery of agents and databases
- Display of database structure and configuration information
- Dynamic sampling of physical and logical log volumes

- Dynamic sampling of volumes usage
- Dynamic statistics collection on multiple detachable graphs and information on sessions, transactions and locks
- Customizable statistics and logging
- Dynamic sampling and parsing of LOGFILE messages and alarms triggering
- Configurable threshold and event triggered alarms with acknowledge
- Configurable alarm logging and notification
- Standalone multithreaded agent
- Multiple protocol adaptors
- SNMP v3 compliant agent - MIB II implementation
- Versant Enterprise MIB implementation
- Versant enterprise specific traps with configurable resend timeout
- Multi-protocol dynamic agent announcement and discovery
- Integration as a subagent with standard master agents

1.1.2. Database Administration and Management

Also provided with Vitness are a number of tools for database administration including license viewer.

And you add Vitness Vorkout, a tool for compacting Versant Object Database databases to reclaim space resulting from fragmentation. (Vorkout is licensed separately. Contact your Versant sales representative for details.)

1.2. Installation and Start-up

Vitness is supplied as two separate components and installations, one for the VOD Agent and one for the Vitness application which includes the Vitness Database Monitor. The VOD Agent is installed on the machine hosting the Versant Object Database for the databases you wish to monitor. The Vitness application may be installed on any machine and is independent of any Versant Object Database installation. The following sections describe the installation procedures and how to run the Vitness components.

1.2.1. The VOD Agent

The VOD Agent must be installed on the same machine as the Versant Object Database server. To install the VOD Agent run the install program provided. The installer will prompt you for the location of your Versant Object Database installation and check that the installation is present.

To start the VOD Agent run the **agent.sh** (**agent.bat** for Windows) script with the command line option **-start** (i.e., **agent.sh -start**). This file is in the `bin` directory of your VOD Agent installation.

If you are planning to use the Vitness SNMP Agent interface note that SNMP is disabled by default and must be explicitly enabled. Refer to Chapter 4, *VOD Agent SNMP Support* [p. 75] for details. For information about SNMP in the Vitness refer to Section 4.1, “Introduction to SNMP” [p. 76].

1.2.2. The Vitness Application

To install the Vitness application simply run the installer file provided. Vitness is installed separately from and does not require a Versant Object Database installation.

To run Vitness on a UNIX or Linux machine, execute the vitness.sh file. On Windows, run the vitness.bat file.

Vitness connects to the VOD Agent as the DBA. Therefore you must have a DBA account (an account with the same name as the DBA) on the machine where you run Vitness. You must use that account when starting Vitness.

1.3. Licensing

The VOD Agent requires a properly configured license in your Versant Object Database root directory. The license file for your Versant Object Database installation should be replaced with a new license file that supports the VOD Agent.

Chapter 2. The Vitness Database Monitor

The Vitness Database Monitor is the client counterpart to the VOD Agent and provides a graphical interface for monitoring all the relevant aspects of a Versant Object Database system.

The Vitness Database Monitor can be used to discover and configure VOD Agent instances and to discover and monitor Versant Object Database databases. The SLP protocol is used to discover the VOD Agent and Versant Object Database. The Java RMI protocol is used to communicate with the VOD Agent instances.

When you start Vitness for the first time, you will see a welcome screen. When you close this welcome screen, you see that the Vitness Database Monitor screen is divided into four main areas.

- Menu and Toolbar
- Service Browser View
- Statistic View and Generic Property View
- Alarms, License, Log, Progress, and Error Log Views

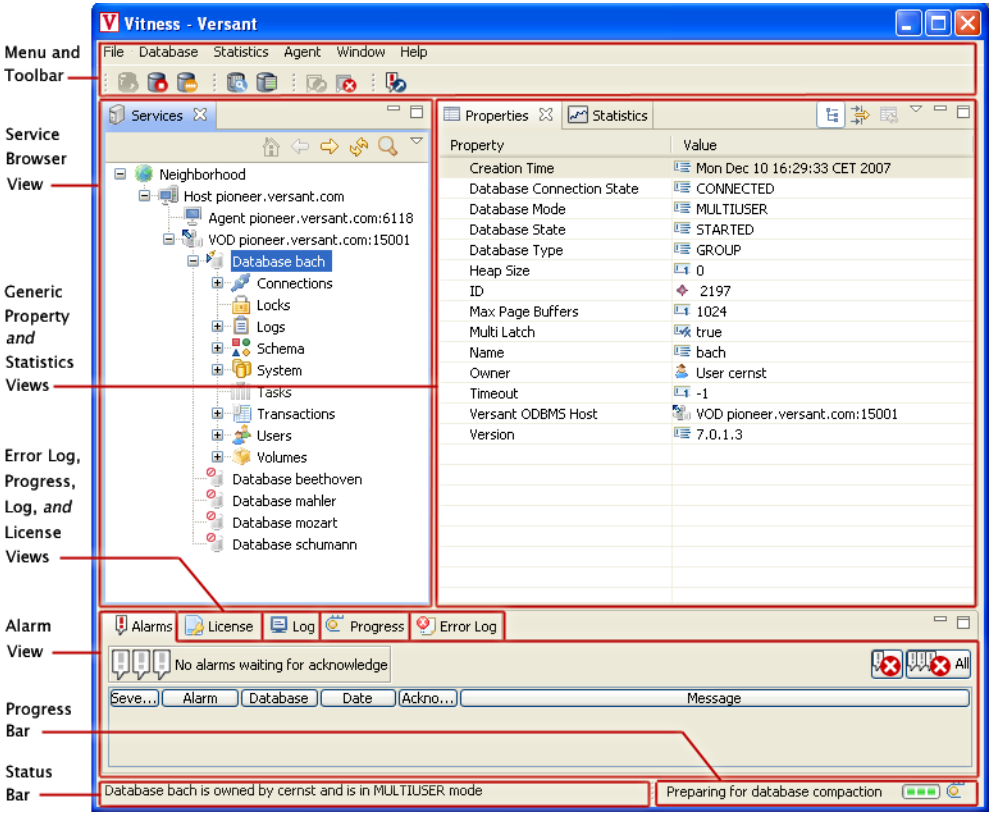


Figure 2.1. The Vitness Database Monitor Screen

Each of the views can be expanded or collapsed by clicking on the buttons in the upper right corner of the view (**Minimize**, **Maximize**, **Restore**). You can also resize the views by dragging their borders.

The toolbar at the top of the main Vitness Database Monitor window displays a series of buttons that can be used as shortcuts to various menu items. Each button has a tool tip that shows its function. To view the tool tip hold the mouse cursor over the button. The toolbar buttons and their use are shown in the following figure.

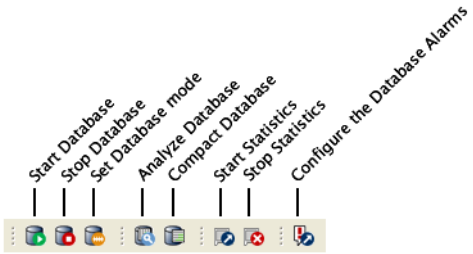


Figure 2.2. The Vitness Database Monitor toolbar

2.1. Vitness Database Monitor Configuration

The configuration parameters of a Vitness Database Monitor can be set using the **Preferences** window. These settings are related to communication parameters, logging activity and security policy. Selecting **Preferences...** from the **Window** menu opens the window. The settings are arranged in a tree view under the heading **Monitoring**. The following settings are provided.

- **Agent** (refer to Section 2.1.1, “VOD Agent and Versant Object Database Discovery” [p. 8] and Section 2.1.2, “Vitness Database Monitor Alarms Communication Parameters” [p. 9])
- **Logging, Alarms** (refer to Section 2.14, “Alarms” [p. 51])
- **Logging, Console** (refer to Section 2.1.3, “Logging Parameters” [p. 10])
- **Logging, Statistics** (refer to Section 2.6, “Statistics” [p. 20])
- **Update**
- **Network**

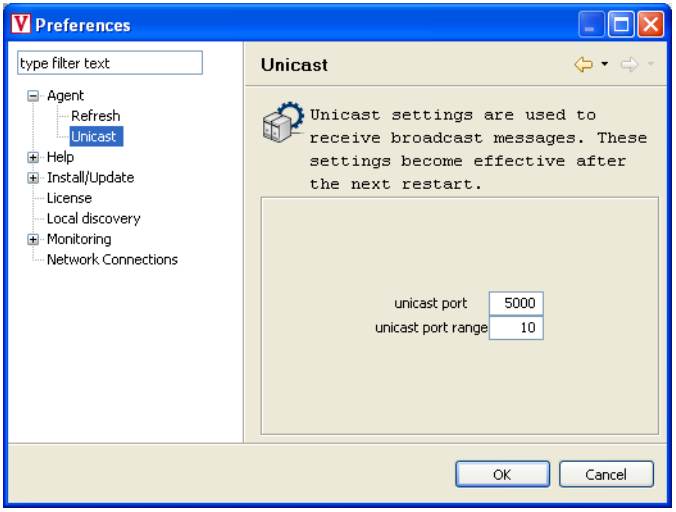


Figure 2.3. Preferences window

The configuration parameters for the Vitness Database Monitor are stored in an XML file named `/vitness/plugins/com.versant.tools.monitor.core_1.3.0.200703131036/console.xml` (the full name of the `com.versant.tools.monitor.core_...` directory will vary depending on the version and build date). Even though it is possible to edit this file directly it is strongly recommended that you make configuration changes using the **Preferences** window as described above. Manually editing the `console.xml` file could result in Vitness becoming inoperable.

2.1.1. VOD Agent and Versant Object Database Discovery

By default, in a standard installation of Vitness, when a Vitness Database Monitor is started an automatic discovery process is started on the local subnet to find running VOD Agent instances and Versant Object Database servers.

The Vitness components announce themselves when started using the SLP. Although the discovery mechanism is of little use on an isolated standalone machine, it can still be used by configuring a loopback adaptor. However, this approach does not work on Windows-based machines

If the multicast protocol cannot be used (for example, because of the presence of a firewall) the manual discovery can be used instead to connect to the VOD Agent instances and Versant Object Database servers.

Both methods can be used at the same time. The only requirement is that the VOD Agent and Vitness Database Monitor communication parameter configurations match.

2.1.1.1. Versant Object Database Manual Discovery

A Versant Object Database can be located manually. This is useful if SLP discovery protocol cannot be used (e.g., because of a firewall).

From the Service Browser View toolbar click the button. A pop-up window, **Connect to Service** will appear on the screen. It will ask for specification of the scheme, host address and port. The format of the address is also specified in the address bar of the pop-up window, “scheme://host.domain.tld:port”.

Scheme

The scheme component has two options, “vod” and “vodagent”. To connect to VOD replace the scheme in the address bar of the window with “vod” and to connect to the Agent on the host replace scheme with “vodagent”.

Host Address

Enter the host name (host.domain.tld) or IP address of the host that you desire to be connected to.

Port

Specification of the port is optional. The default for "vod" is 5019 and that for "vodagent" is 6118.

After specifying the address to locate press the **Ok** button.

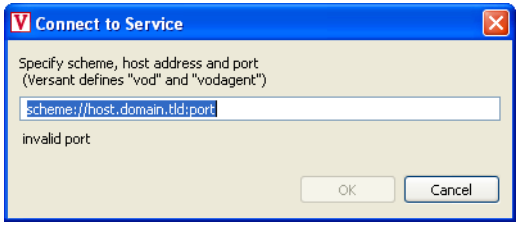
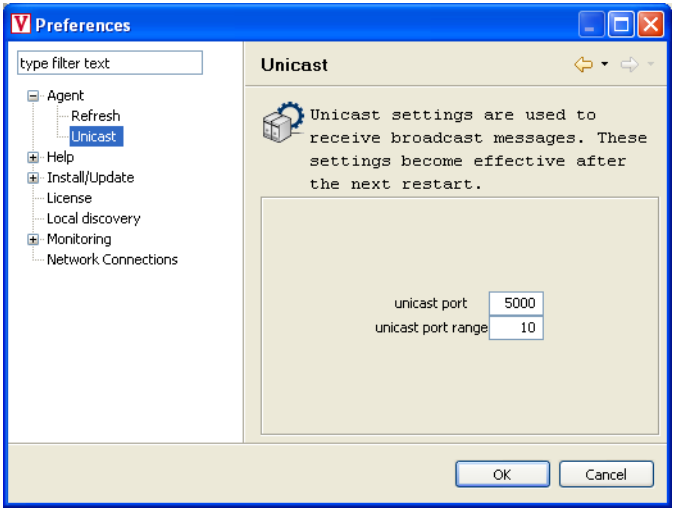


Figure 2.4. Agent lookup dialog

2.1.2. Vitness Database Monitor Alarms Communication Parameters

The alarms configuration parameters for the Vitness Database Monitor are specified in the **Agent Unicast** panel under the **Agent** tree in the **Preferences** window.



The parameters are described in Section 3.1.4, “VOD Agent Communication Parameters” [p. 61]. The configuration must match the corresponding parameters specified in the VOD Agent configuration.

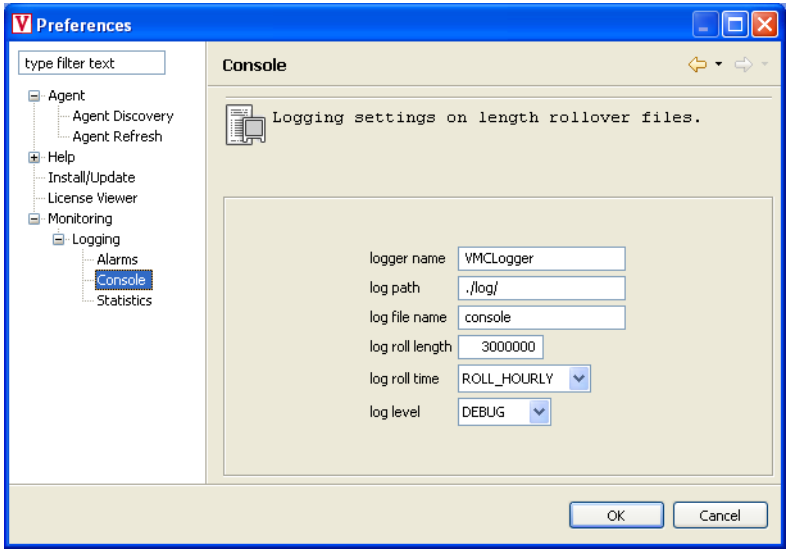
The following example shows the parameters in the `console.xml` configuration file.

Example 2.1. Vitness Database Monitor alarms communication parameters

```
<rmi>
<rmi-registry external="false" port="6118"/>
      <rmi-discovery>
       <multicast ip-address="230.0.0.1" port="4446"/>
            <unicast port="5000" portRange="10"/>
      </rmi-discovery>
</rmi>
```

2.1.3. Logging Parameters

The Vitness Database Monitor logs its activity to a file in the same way as the VOD Agent does. The related configuration parameters are specified in **Console** panel in the **Logging** tree of the **Preferences** window.



These parameters are specified in the `<sys-logger>` element of the `console.xml` file. Refer to Section 3.1.6, “Logging Parameters” [p. 69].

Other logging parameters can be specified to log alarms and statistics data to a file. Such parameters are documented in Section 2.14, “Alarms” [p. 51] and in Section 2.6, “Statistics” [p. 20], respectively.

2.1.4. Vitness Database Monitor and Java Security

The same considerations related to Java security made for the VOD Agent apply to the Vitness Database Monitor. For these configuration parameters, refer to Section 3.2, “VOD Agent and Java Security” [p. 72].

2.2. Database Monitoring

Information about the Versant Object Database system and the static database parameters and structure information include data such as the database name, path, type, etc. This structural and parameter information is made available to the Vitness Database Monitor and is displayed in the **Common Property Page** for each selected element from the Service Browser View.

The information is presented in a tree view. If you select the Versant Object Database system (the server) the related information is displayed as shown in the following figure.

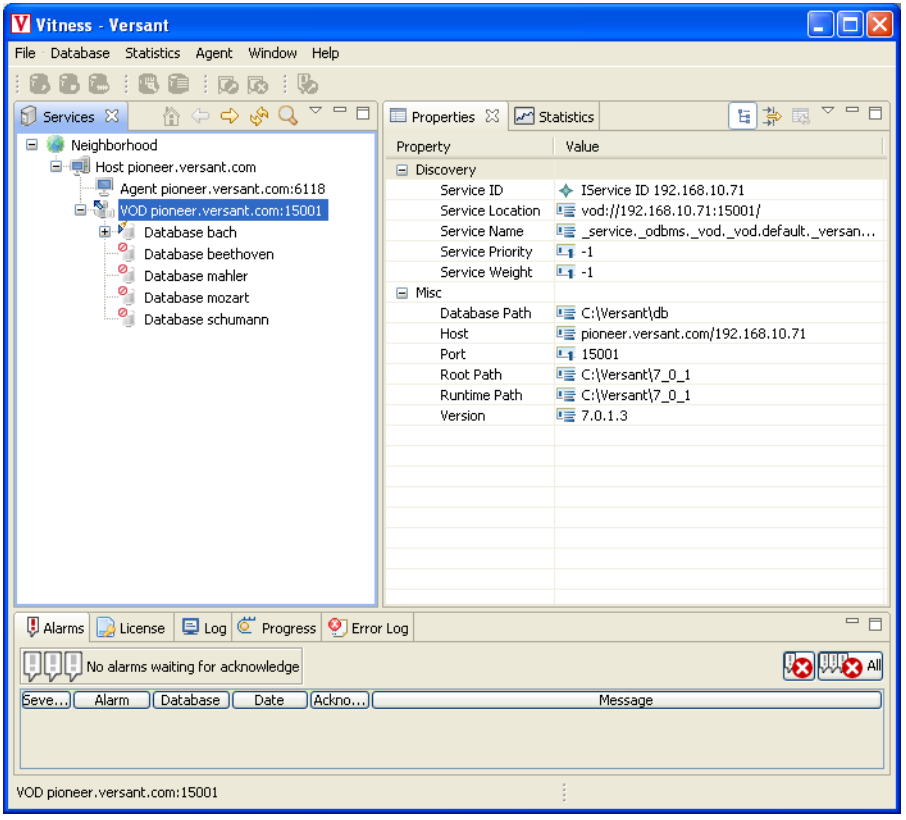


Figure 2.5. Versant Object Database System Properties View

The information for a specific database is displayed by selecting the database in the tree as shown in the next figure.

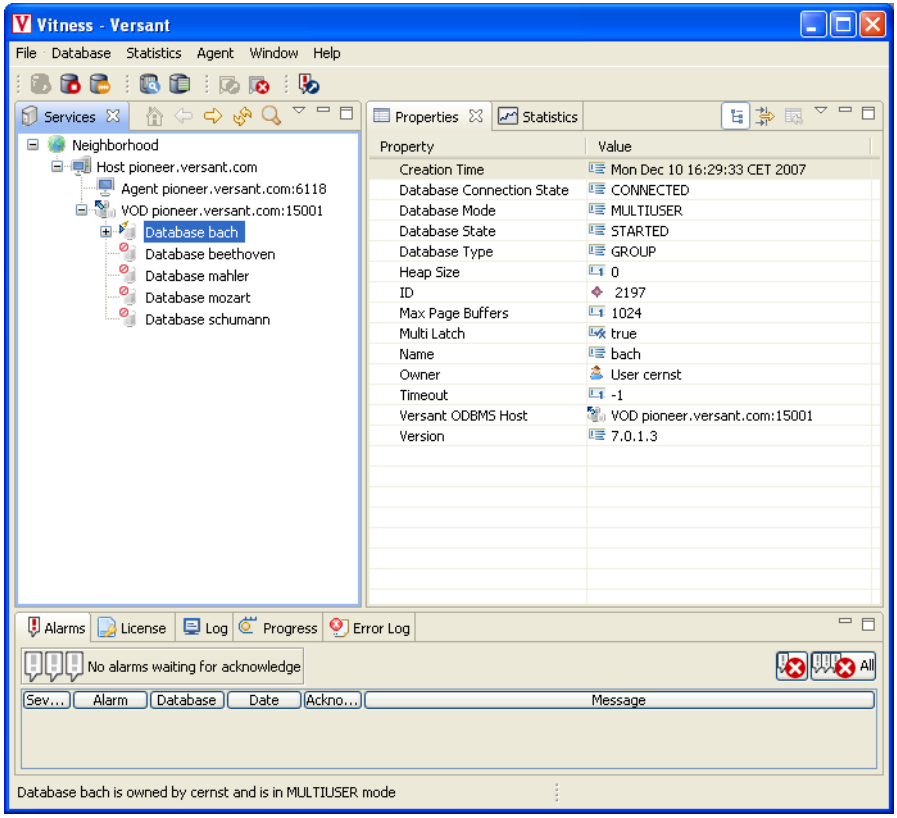
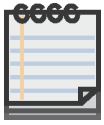


Figure 2.6. Database Structure Information in the Vitness Database Monitor



This cascading tree view model is used for other properties views such as connections, schema, locks, etc.

The system and database information is also made available, through the appropriate protocol adaptor, to a SNMP management station as a MIB.

All databases deployed on a host where a VOD Agent has been discovered are displayed in the Service Browser View of the Vitness Database Monitor. All displayed databases are made available for monitoring. To start monitoring a database, click on its icon to select it and then either click on the button in the toolbar, select **Connect** from the **Monitor** menu, or right-click on the database icon and click on the **connect** item in the pop-up menu.

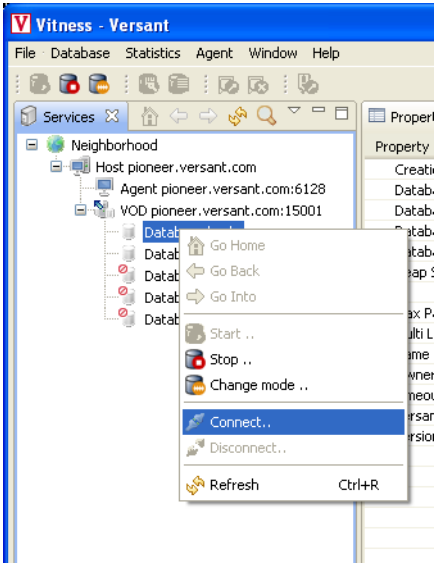


Figure 2.7. Database monitoring

If the database requires authentication before allowing the connection a login window is displayed. You must supply a valid user name and password.

Figure 2.8. Login window

As soon as the connection is established, the database being monitored starts and the individual database parts appear as subnodes in the Service Browser Tree.

It is possible to monitor multiple databases at the same time. Connect to them and then switch from one to another by selecting the desired database icon from the Service Browser View. The Statistic View and Property View will show the monitored components of the selected database.

To stop monitoring a database, click on its icon to select it, then either click on the button in the toolbar or select **Disconnect** from the **Monitor** menu. You can also right-click on the database icon and click the **disconnect** item in the pop-up menu.

2.2.1. Standalone Database Monitoring

It is possible to continue monitoring a database using a VOD Agent without a connected Vitness Database Monitor. This allows a standalone VOD Agent to be used by other management tools. This is the case, for example, with pure SNMP monitoring.

Standalone monitoring can be activated for the selected database by selecting **Enable Persistent Monitoring** in the **Agent** menu.

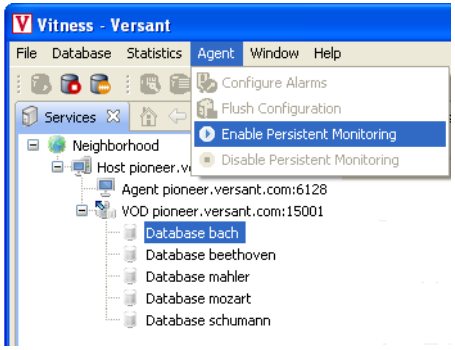


Figure 2.9. Standalone database monitoring

From that point, it is possible to stop the monitoring activity of the Vitness Database Monitor but the VOD Agentwill continue monitoring the database, even if the Vitness Database Monitor is closed.

When a database is being monitored by a Vitness Database Monitor, a database session is kept open. This is necessary for the Vitness Database Monitor to gather information related to the database activity. This also means that a database cannot be shutdown gracefully unless all monitoring by any Vitness Database Monitor is stopped.

Terminate persistent monitoring by selecting the appropriate database and clicking **Disable Persistent Monitoring** in the **Agent** menu.

2.3. Log Volumes

The physical and logical log volumes can be displayed in graphical form in the Vitness Database Monitor in the Volume Property View or retrieved from the MIB through a protocol adaptor by a SNMP management station. For each log volume the following information is displayed.

Static information

- Volume Name
- Volume Path
- Initial Size
- The log volume is on raw device

Dynamic information

- Actual size
- Free bytes

If the log volumes are expanded beyond their configured size an alarm is sent to the Vitness Database Monitor. If the SNMP adaptor is enabled a specific trap is triggered.

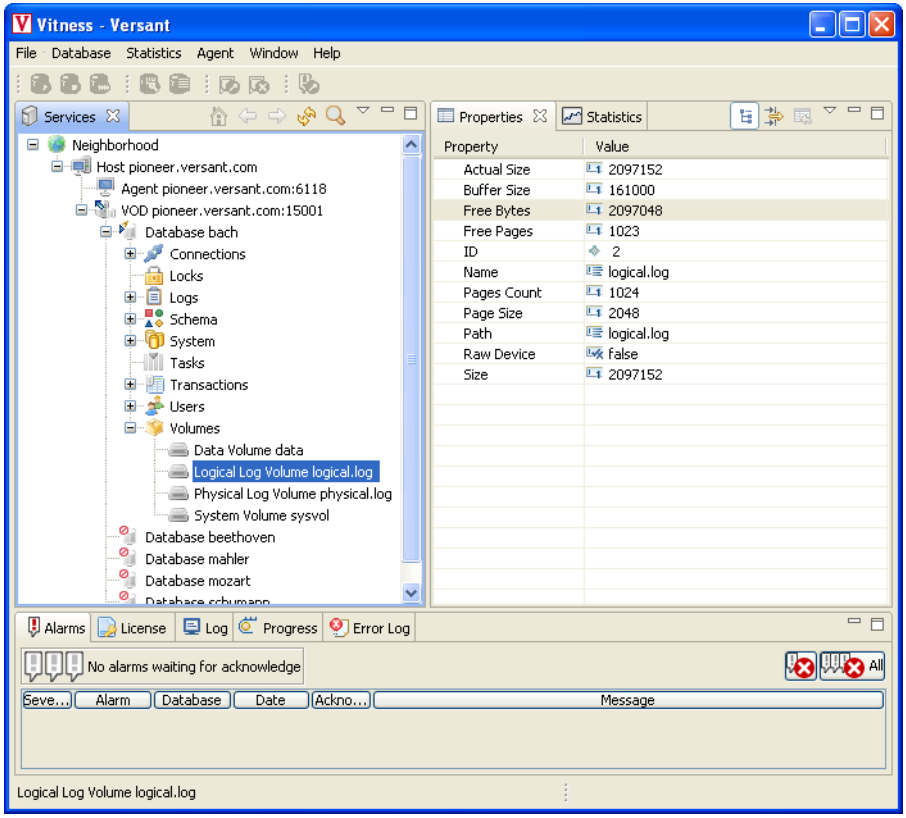


Figure 2.10. Log volumes monitoring

In Vitness a kiloByte (Kb) is 1000 bytes. This may cause file sizes, etc., reported by Vitness to be different from those reported by other tools (where a kiloByte is 1024 bytes).

2.4. Data Volumes

Each mounted data volume is monitored to determine the free space available and also other volume related information.

In the first list are items that are set when the volume is created.

- Volume ID
- Volume name
- Volume path
- Volume size (bytes)
- Extent size

The following lists the items that are dynamically monitored.

- Number of extents
- Available extents
- Available bytes

In the Vitness Database Monitor a graphical representation of the volumes shows the percent of used space in each volume of the monitored database. An alarm is sent when a volume reaches a user definable used space threshold. The total space available in the entire database is monitored as well and an alarm is sent when the total free space falls below a configurable threshold. If the SNMP protocol adaptor is enabled such events trigger specific traps. The alarm thresholds can be set through the Vitness Database Monitor or using an SNMP set request. (Alarms are described in detail in Section 2.14, “Alarms” [p. 51]. For for inforamtion about SNMP and Vitness refer to Chapter 4, *VOD Agent SNMP Support* [p. 75].)

Each data volume of the monitored database is displayed in the **Volumes Property View**.

For each volume all the space related information is sampled and displayed and the total size of the monitored database and its free space available are reported.

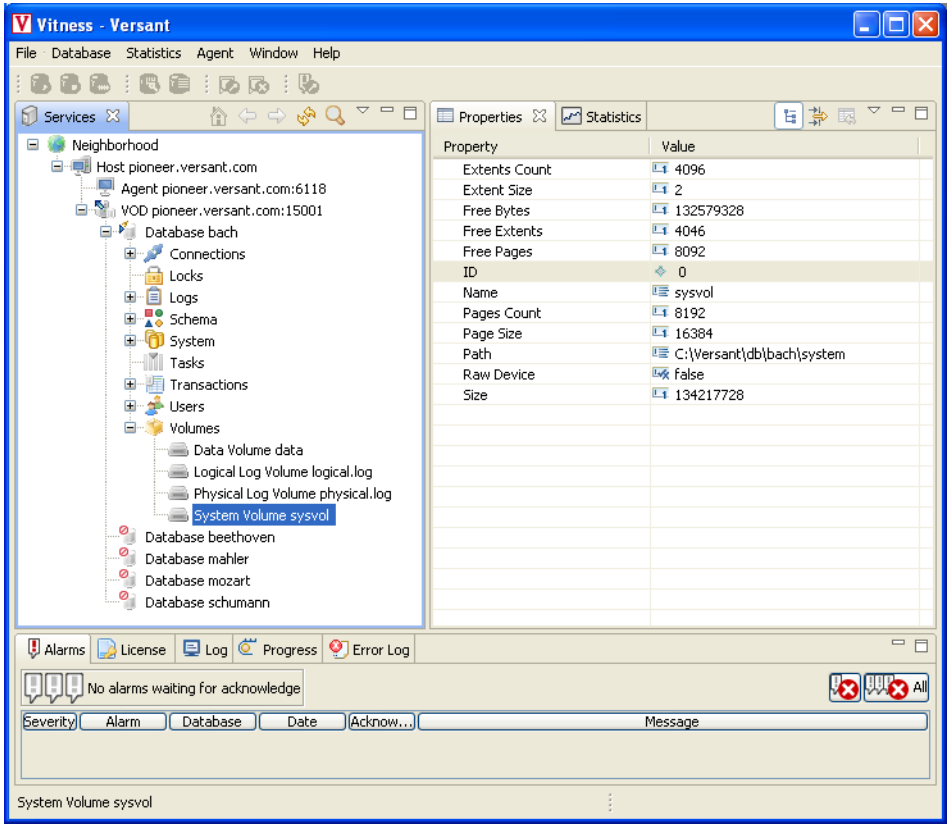


Figure 2.11. Data volume monitoring

2.5. Database LOGFILE

The monitored database LOGFILE is sampled and parsed in order to detect the occurrence of relevant events. The detected events are those generated by the monitored database itself, such as server side errors, or by actions performed with Versant Object Database utilities such as addvol, stopdb, etc.

The content of the LOGFILE of the monitored database is shown in the Database Log View. To open the Database Log View click on the **Database Log** and select the **Show** from Monitor menu or right click and select **show** from the pop-up menu. This file contains all of the messages logged by the Versant Object Database server and by the Versant Object Database utilities.

The messages logged to the LOGFILE may contain information related to errors generated by the server or to actions taken by the database administrator. These messages are analysed to detect alarm conditions that need to be reported to the Vitness Database Monitor (refer to Section 2.14, “Alarms” [p. 51]) and, if the SNMP adaptor is activated (refer to Chapter 4, *VOD Agent SNMP Support* [p. 75]), to trigger a specific SNMP trap.

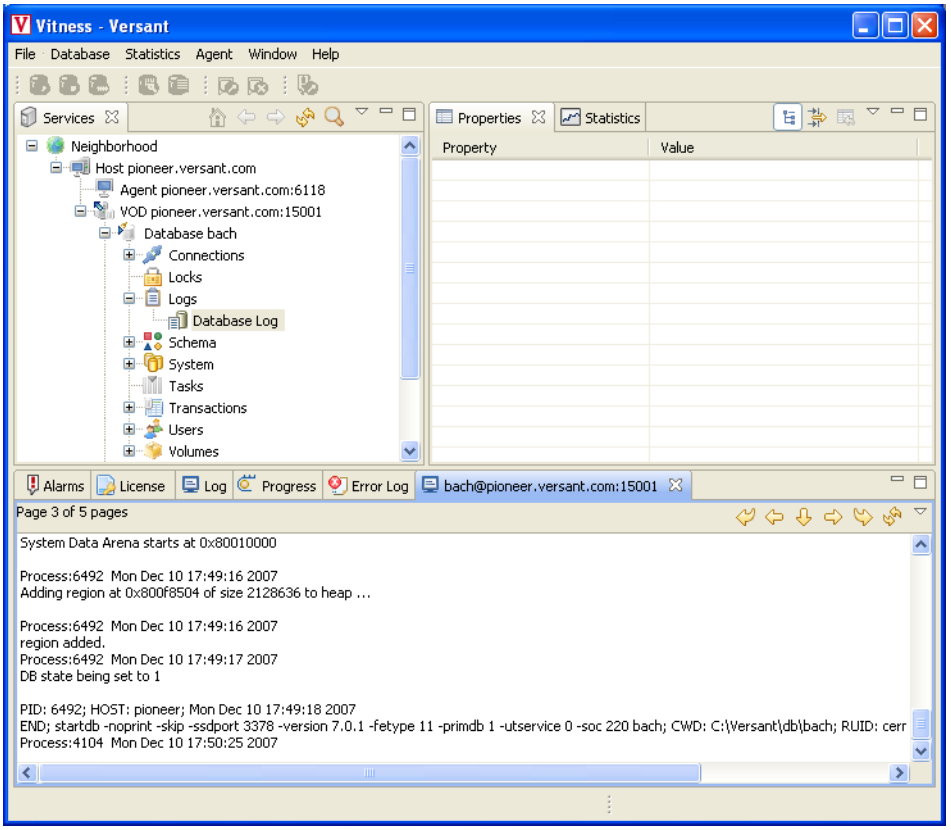


Figure 2.12. LOGFILE monitoring

2.6. Statistics

Performance data collection is one of the fundamental monitoring activities for the correct tuning of a database, and more generally of a system, in a production environment. The Versant Object Database server allows the activation of statistics that can provide valuable data related to every aspect of server behavior and performance.

Typically, a graphical representation of such data helps in quickly understanding whether the system is healthy or requires intervention. The real time display of statistics can help in troubleshooting performance bottlenecks and, by comparing data collected in different situations, allows you to pinpoint the specific area of attention. The historical data collected allows you to study the long term trend of the system and help identify potential problems in advance.

The Vitness Database Monitor allows you to collect performance data and display them in both a graphical and numerical form, save the data to a file, and analyze information related to the database transactional activity such as connections, transactions and locks.

It is important to note that the Vitness Database Monitor itself keeps an open session with the monitored database. This session is used to collect the connections, transactions, locks related information, and, when activated, to collect statistics. The name of this session is in the form `VMC_`*`dbname@host`*.

2.6.1. The Vitness Database Monitor Statistics View

To open the Statistic View click on the Database and select **start statistic** from the statistic menu or right click and select **start statistic** from the pop-up menu.

The **Statistics** view in the Vitness Database Monitor is shown below.

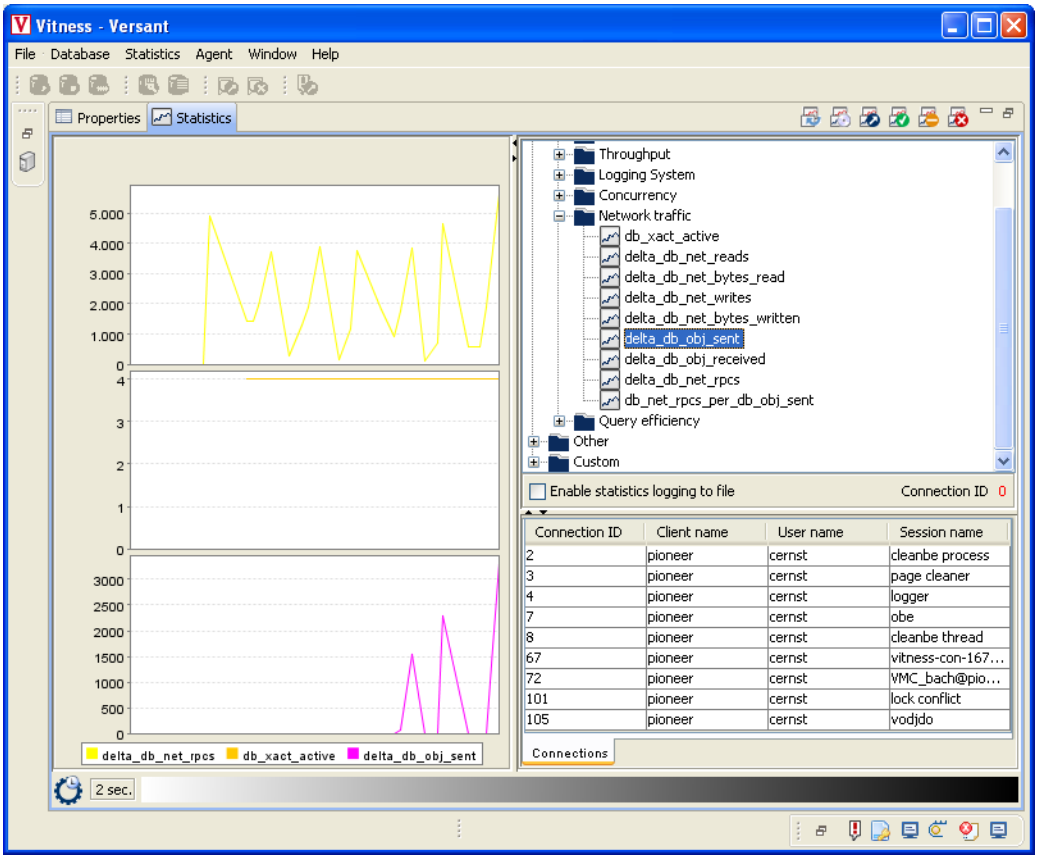


Figure 2.13. The Statistics view

The Statistics view toolbar is shown below.

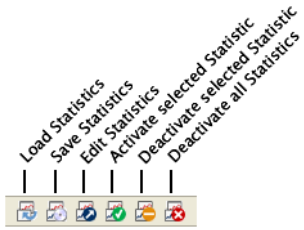


Figure 2.14. Statistics View Toolbar

The panel is divided into three main areas.

- The statistics selection view is where all if the available statistics are grouped and may be selected.
- The graph view is where data for the selected statistics are plotted.
- The data display view is further divided into three tabbed panels for the dynamic display of data related to connections, transactions, and locks. These are described in following sections.

2.6.2. Activating and Viewing Statistics

All the available statistics are grouped in branches of the statistics tree. From there they can be selected and dragged over the graph area to be displayed. More statistics can be displayed using drag and drop, in a single or in multiple graphs.

If the selected statistics is dropped over an existing graph, it will be added to it, otherwise, if dropped over an empty area, a new graph will be displayed.

Each time a new statistic is added to the graphs area, the numerical display is updated.

Each graph can be detached by right-clicking on it and selecting **Detach** from the pop up menu.

The numerical values of the collected data that are plotted in the graphs, are also displayed in the **Statistics** tabbed panel of the data display view. Such values can be saved to a file checking the **Enable statistics logging to file** checkbox just below the selection tree. The collected data will be saved as configured in the **Statistics logging** panel of the **Preferences** window.

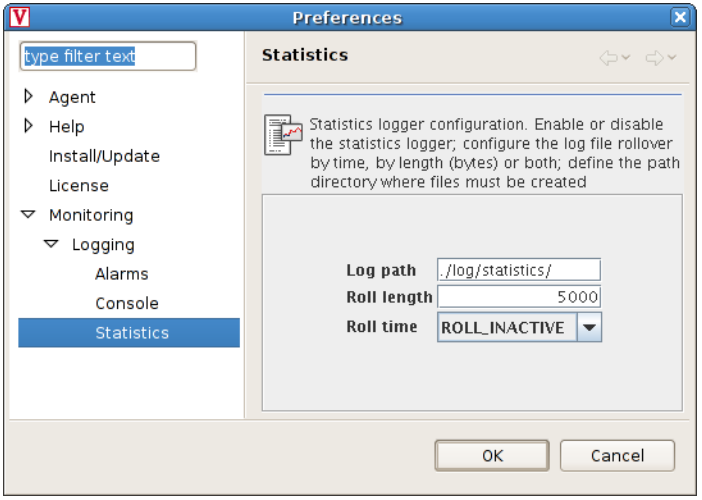


Figure 2.15. Statistics logging configuration

Refer to Section 2.1.3, “Logging Parameters” [p. 10] for the documentation of the required parameters.

2.6.3. Creating Custom Statistics

It is possible to create custom statistics from those already defined for the Versant Object Database, defining their formula in a user interface and saving them to disk.

The configuration and definition of a list of custom statistics is defined in an XML file named `Statistics.xml`. This file, in the `.vire` sub-directory of your home directory, is created the first time a custom statistic is created.

The **Statistics configuration** window can be invoked by the **Statistics/Custom statistics...** menu, or clicking on the **Edit custom statistics** button on the top toolbar.

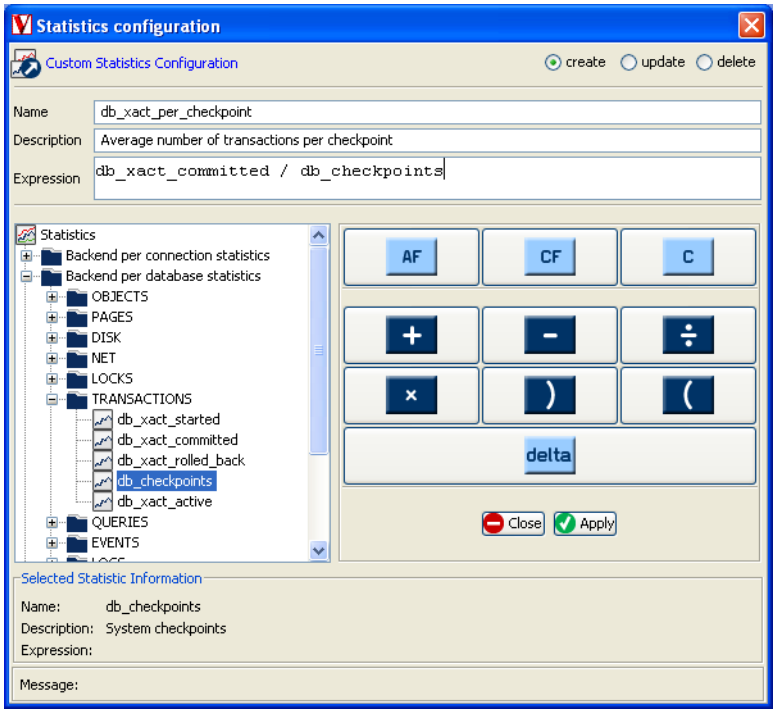


Figure 2.16. Creating custom statistics

Custom statistics can be created, modified and deleted by checking the appropriate item of the top radio button.

To create a new statistics expression, select the single components of the expression from the tree that groups all the base statistics, add each one to the expression pressing the **AF** button, and compose with the operators. When the expression is complete, add a name and a description and click on the **Apply** button: the new custom statistic will be save under the **Custom\My Statistics** group. Custom statistics are saved when you press the **Close** button. The expressions are saved in a file named `custom-statistic.xml` in the `.vire` sub-directory of your `home` directory.

2.6.4. Statistics Groups

The following sections list the statistics available to you in the Vitness Database Monitor. They are arranged in the same groups in which they are displayed. For more information about statistics in Versant Object Database, refer to the chapter *Statistics Collection* in your *Versant Database Fundamentals Manual*. The full list of statistics available in Versant Object Database is given in Appendix A, *Versant Object Database Statistics* [p. 153].

2.6.4.1. Tuning

Many of the parameters that affect database operation can be modified. But in order to know what values are best, you need to have statistical information about the database. The following statistics groups measure many of the parameters responsible for optimal server/database performance. For information about modifying these parameters refer to the *Database Profiles* chapter in your *Versant Database Administration Manual*.

Many of the statistics have a *delta* implementation. These are the statistics names beginning with `delta_`. The delta statistics display the data change in the last sample period, as opposed to the normal cumulative statistics.

2.6.4.1.1. Server cache efficiency

This set of statistics allows you to measure the effectiveness of the Versant Object Database server cache. A high cache hit ratio means that most of of the objects sent to clients were found in the server cache, that is, read from server-side memory rather than from disk. Memory reads are significantly faster so a high cache ratio is desirable. For optimal performance this ratio should be above 0.8 and preferrably as close 1.0 as possible.

- `db_xact_active`
- `delta_db_data_reads`

  formula: `% db_data_reads`
- `delta_db_data_located`

  formula: `% db_data_located`
- `db_cache_hit_ratio`

  formula: `db_data_located / (db_data_located + db_data_reads)`
- `db_inst_cache_hit_ratio`

  formula: `% db_data_located / (% db_data_located + % db_data_reads)`

2.6.4.1.2. Throughput

These statistics can be used to give you information about the commit throughput of the system. A high throughput is important for update intensive applications.

- db_xact_active
- delta_db_xact_committed

  formula: % db_xact_committed
- delta_db_xact_rolled_back

  formula: % db_xact_rolled_back
- delta_db_net_rpcs

  formula: % db_net_rpcs
- delta_db_data_located

  formula: % db_data_located
- delta_db_data_reads

  formula: % db_data_reads
- delta_db_data_writes

  formula: % db_data_writes
- delta_db_checkpoints

  formula: % db_checkpoints

2.6.4.1.3. Logging System

The statistics in this group allow you to monitor the efficiency of the logging system. Logging checkpoints occur whenever the physical or logical log file becomes full. For a well tuned logging system you want to see that there is at least thirty seconds between checkpoints and that each checkpoint is completed quickly, within a few seconds. For information on setting the log file size refer to the *Tuning Parameters* section in your *Versant Database Administration Manual*.

- delta_db_checkpoints

  formula: % db_checkpoints
- delta_db_xact_committed

  formula: % db_xact_committed

- delta_db_bf_llog_flushes

  formula: % db_bf_llog_flushes

- delta_db_bf_plog_flushes

  formula: % db_bf_plog_flushes

- delta_db_bf_llog_bytes_written

  formula: % db_bf_llog_bytes_written

- delta_db_bf_plog_bytes_written

  formula: % db_bf_plog_bytes_written

- delta_db_data_writes

  formula: % db_data_writes

2.6.4.1.4. Concurrency

These statistics allow you to monitor the locking concurrency conditions of your database. If you see large values for lock waits, lock wait time, or deadlocks this is an indication that lock conflicts are present. Such conflicts can have a dramatic impact on performance. Changes to your application code may be needed to reduce the possibilty of lock conflicts.

- db_xact_active
- db_lock_waits
- db_obe_locks_waiting
- delta_db_lock_timeouts

  formula: % db_lock_timeouts

- delta_db_lock_wait_time

  formula: % db_lock_wait_time

- delta_db_lock_deadlocks

  formula: % db_lock_deadlocks

2.6.4.1.5. Network traffic

The statistics in this group monitor network communication traffic between database clients and the server. Large numbers of remote procedure calls (RPCs) per object, for example, implies that network traffic could be greatly reduced by using *group operations* in your application. Refer to the descriptions for `greadobjs()` and `gwriteobjs()`, for example, in your Versant API documentation.

- `db_xact_active`
- `delta_db_net_reads`

  formula: `% db_net_reads`
- `delta_db_net_bytes_read`

  formula: `% db_net_bytes_read`
- `delta_db_net_writes`

  formula: `% db_net_writes`
- `delta_db_net_bytes_written`

  formula: `% db_net_bytes_written`
- `delta_db_obj_sent`

  formula: `% db_obj_sent`
- `delta_db_obj_received`

  formula: `% db_obj_received`
- `delta_db_net_rpcs`

  formula: `% db_net_rpcs`
- `db_net_rpcs_per_db_obj_sent`

  formula: `% db_net_rpcs / (% db_obj_sent )`

2.6.4.1.6. Query efficiency

The following statistics relate to query performance. A large value for `db_qry_scan_objs` (seconds spent in sequential scan query) indicates that you can significantly improve performance by adding an appropriate index (or indexes). Large values for btree- and hash-based queries can indicate that indexes may not be effective.

- `db_xact_active`
- `delta_db_qry_scan_objs`

  formula: `% db_qry_scan_objs`
- `delta_db_qry_btree_objs`

  formula: `% db_qry_btree_objs`
- `delta_db_qry_hash_objs`

  formula: `% db_qry_hash_objs`
- `delta_db_qry_btree_time`

  formula: `% db_qry_btree_time`
- `delta_db_qry_hash_time`

  formula: `% db_qry_hash_time`
- `delta_db_qry_scan_time`

  formula: `% db_qry_scan_time`

2.6.4.2. Other

2.6.4.2.1. Backend per connection statistics

2.6.4.2.1.1. OBJECTS

- `be_obj_sent`
- `be_obj_received`

2.6.4.2.1.2. PAGES

- be_data_reads
- be_data_writes
- be_data_located
- be_vm_maj_faults

2.6.4.2.1.3. NET

- be_net_rpcs
- be_net_reads
- be_net_bytes_read
- be_net_read_time
- be_net_writes
- be_net_bytes_written
- be_net_write_time

2.6.4.2.1.4. LOCKS

- be_locks_granted
- be_lock_waits
- be_obe_locks_waiting
- be_lock_timeouts
- be_lock_wait_time
- be_lock_deadlocks

2.6.4.2.1.5. TRANSACTIONS

- be_xact_started
- be_xact_committed

- be_xact_rolled_back
- be_xact_active

2.6.4.2.1.6. QUERIES

- be_qry_btree_objs
- be_qry_hash_objs
- be_qry_scan_objs
- be_qry_btree_time
- be_qry_hash_time
- be_qry_scan_time

2.6.4.2.1.7. EVENTS

- be_ev_defined
- be_ev_sys_raised
- be_ev_sys_delivered
- be_ev_user_raised
- be_ev_user_delivered

2.6.4.2.1.8. RUNNING TIME

- be_user_time
- be_system_time
- be_real_time

2.6.4.2.1.9. LATCHES

- be_latch_released

2.6.4.2.1.10. GRANTED

- be_latch_granted

- be_latch_granted_sda
- be_latch_granted_heap
- be_latch_granted_voldev
- be_latch_granted_st
- be_latch_granted_ss
- be_latch_granted_sdhs
- be_latch_granted_sdhs_bkt
- be_latch_granted_ps
- be_latch_granted_tr
- be_latch_granted_ev
- be_latch_granted_plog
- be_latch_granted_llog
- be_latch_granted_cp
- be_latch_granted_cp_wait
- be_latch_granted_sch
- be_latch_granted_sce
- be_latch_granted_phy
- be_latch_granted_bf
- be_latch_granted_bf_bkt
- be_latch_granted_bf_free
- be_latch_granted_l2file
- be_latch_granted_sd
- be_latch_granted_sc

- be_latch_granted_bf_dirty
- be_latch_granted_log_unit
- be_latch_granted_tre
- be_latch_granted_lock
- be_latch_granted_l2file_da

2.6.4.2.1.11. WAITED

- be_latch_waits
- be_latch_waits_sda
- be_latch_waits_heap
- be_latch_waits_voldev
- be_latch_waits_st
- be_latch_waits_ss
- be_latch_waits_sdhs
- be_latch_waits_sdhs_bkt
- be_latch_waits_ps
- be_latch_waits_tr
- be_latch_waits_ev
- be_latch_waits_plog
- be_latch_waits_llog
- be_latch_waits_cp
- be_latch_waits_cp_wait
- be_latch_waits_sch
- be_latch_waits_sce

- be_latch_waits_phy
- be_latch_waits_bf
- be_latch_waits_bf_bkt
- be_latch_waits_bf_free
- be_latch_waits_l2file
- be_latch_waits_sd
- be_latch_waits_sc
- be_latch_waits_bf_dirty
- be_latch_waits_log_unit
- be_latch_waits_tre
- be_latch_waits_lock
- be_latch_waits_l2file_da

2.6.4.2.1.12. WAIT TIME

- be_latch_wait_time
- be_latch_wait_time_sda
- be_latch_wait_time_heap
- be_latch_wait_time_voldev
- be_latch_wait_time_st
- be_latch_wait_time_ss
- be_latch_wait_time_sdhs
- be_latch_wait_time_sdhs_bkt
- be_latch_wait_time_ps
- be_latch_wait_time_tr

- be_latch_wait_time_ev
- be_latch_wait_time_plog
- be_latch_wait_time_llog
- be_latch_wait_time_cp
- be_latch_wait_time_cp_wait
- be_latch_wait_time_sch
- be_latch_wait_time_sce
- be_latch_wait_time_phy
- be_latch_wait_time_bf
- be_latch_wait_time_bf_bkt
- be_latch_wait_time_bf_free
- be_latch_wait_time_l2file
- be_latch_wait_time_sd
- be_latch_wait_time_sc
- be_latch_wait_time_bf_dirty
- be_latch_wait_time_log_unit
- be_latch_wait_time_tre
- be_latch_wait_time_lock
- be_latch_wait_time_l2file_da

2.6.4.2.2. Backend per database statistics

2.6.4.2.2.1. OBJECTS

- db_obj_sent
- db_obj_received

2.6.4.2.2.2. PAGES

- db_data_reads
- db_data_writes
- db_data_located

2.6.4.2.2.3. DISK

- db_disk_free
- db_disk_reserved

2.6.4.2.2.4. NET

- db_net_rpcs
- db_net_reads
- db_net_bytes_read
- db_net_read_time
- db_net_writes
- db_net_bytes_written
- db_net_write_time

2.6.4.2.2.5. LOCKS

- db_locks_granted
- db_lock_waits
- db_obe_locks_waiting
- db_lock_timeouts
- db_lock_wait_time
- db_lock_deadlocks

2.6.4.2.2.6. TRANSACTIONS

- db_xact_started
- db_xact_committed
- db_xact_rolled_back
- db_checkpoints
- db_xact_active

2.6.4.2.2.7. QUERIES

- db_qry_btree_objs
- db_qry_hash_objs
- db_qry_scan_objs
- db_qry_btree_time
- db_qry_hash_time
- db_qry_scan_time

2.6.4.2.2.8. EVENTS

- db_ev_defined
- db_ev_sys_raised
- db_ev_sys_delivered
- db_ev_user_raised
- db_ev_user_delivered

2.6.4.2.2.9. LOGS

- db_bf_llog_flushes
- db_bf_plog_flushes
- db_bf_llog_bytes_written

- db_bf_plog_bytes_written
- db_bf_llog_full
- db_bf_llog_end
- db_bf_plog_full
- db_bf_plog_end

2.6.4.2.2.10. LATCHES

- db_latch_released

2.6.4.2.2.11. GRANTED

- db_latch_granted
- db_latch_granted_sda
- db_latch_granted_heap
- db_latch_granted_voldev
- db_latch_granted_st
- db_latch_granted_ss
- db_latch_granted_sdhs
- db_latch_granted_sdhs_bkt
- db_latch_granted_ps
- db_latch_granted_tr
- db_latch_granted_ev
- db_latch_granted_plog
- db_latch_granted_llog
- db_latch_granted_cp
- db_latch_granted_cp_wait

- db_latch_granted_sch
- db_latch_granted_sce
- db_latch_granted_phy
- db_latch_granted_bf
- db_latch_granted_bf_bkt
- db_latch_granted_bf_free
- db_latch_granted_l2file
- db_latch_granted_sd
- db_latch_granted_sc
- db_latch_granted_bf_dirty
- db_latch_granted_log_unit
- db_latch_granted_tre
- db_latch_granted_lock
- db_latch_granted_l2file_da

2.6.4.2.2.12. WAITS

- db_latch_waits
- db_latch_waits_sda
- db_latch_waits_heap
- db_latch_waits_voldev
- db_latch_waits_st
- db_latch_waits_ss
- db_latch_waits_sdhs
- db_latch_waits_sdhs_bkt

- db_latch_waits_ps
- db_latch_waits_tr
- db_latch_waits_ev
- db_latch_waits_plog
- db_latch_waits_llog
- db_latch_waits_cp
- db_latch_waits_cp_wait
- db_latch_waits_sch
- db_latch_waits_sce
- db_latch_waits_phy
- db_latch_waits_bf
- db_latch_waits_bf_bkt
- db_latch_waits_bf_free
- db_latch_waits_l2file
- db_latch_waits_sd
- db_latch_waits_sc
- db_latch_waits_bf_dirty
- db_latch_waits_log_unit
- db_latch_waits_tre
- db_latch_waits_lock
- db_latch_waits_l2file_da

2.6.4.2.2.13. WAIT TIME

- db_latch_wait_time

- db_latch_wait_time_sda
- db_latch_wait_time_heap
- db_latch_wait_time_voldev
- db_latch_wait_time_st
- db_latch_wait_time_ss
- db_latch_wait_time_sdhs
- db_latch_wait_time_sdhs_bkt
- db_latch_wait_time_ps
- db_latch_wait_time_tr
- db_latch_wait_time_ev
- db_latch_wait_time_plog
- db_latch_wait_time_llog
- db_latch_wait_time_cp
- db_latch_wait_time_cp_wait
- db_latch_wait_time_sch
- db_latch_wait_time_sce
- db_latch_wait_time_phy
- db_latch_wait_time_bf
- db_latch_wait_time_bf_bkt
- db_latch_wait_time_bf_free
- db_latch_wait_time_l2file
- db_latch_wait_time_sd
- db_latch_wait_time_sc

- db_latch_wait_time_bf_dirty
- db_latch_wait_time_log_unit
- db_latch_wait_time_tre
- db_latch_wait_time_lock
- db_latch_wait_time_l2file_da

2.6.4.2.2.14. AT TABLE

- db_at_root_read
- db_at_root_located
- db_at_leaf_read
- db_at_leaf_located

2.6.4.2.3. Derived

2.6.4.2.3.1. Per connection backend statistics

- be_cache_hit_ratio

  formula: `be_data_located / (be_data_located + be_data_reads)`

- be_inst_cache_hit_ratio

  formula: `% be_data_located / (% be_data_located + % be_data_reads)`

- be_cpu_time

  formula: `be_system_time + be_user_time`

- be_run_time

  formula: `be_real_time - be_net_read_time - be_net_write_time - be_latch_wait_time`

- be_latch_holds

  formula: `be_latch_granted - be_latch_released`

2.6.4.2.3.2. Per database backend statistics

- db_cache_hit_ratio

  formula: `db_data_located / (db_data_located + db_data_reads)`

- db_inst_cache_hit_ratio

  formula: `% db_data_located / (% db_data_located + % db_data_reads)`

- db_latch_holds

  formula: `db_latch_granted - db_latch_released`

- db_at_root_cache_hit_ratio

  formula: `% db_at_root_located / ( % db_at_root_located + % db_at_root_read)`

- db_at_leaf_cache_hit_ratio

  formula: `% db_at_leaf_located / ( % db_at_leaf_located + % db_at_leaf_read)`

- db_at_cache_hit_ratio

  formula: `( % db_at_root_located + % db_at_leaf_located ) / ( % db_at_root_located + % db_at_root_read + % db_at_leaf_located + % db_at_leaf_read)`

2.7. Connections

The **Connections** in the Service Browser Tree and the Connection Property View reports information on all active sessions for the monitored database. The connection ID, the client host name, the user name and the session name are displayed for the selected connection.

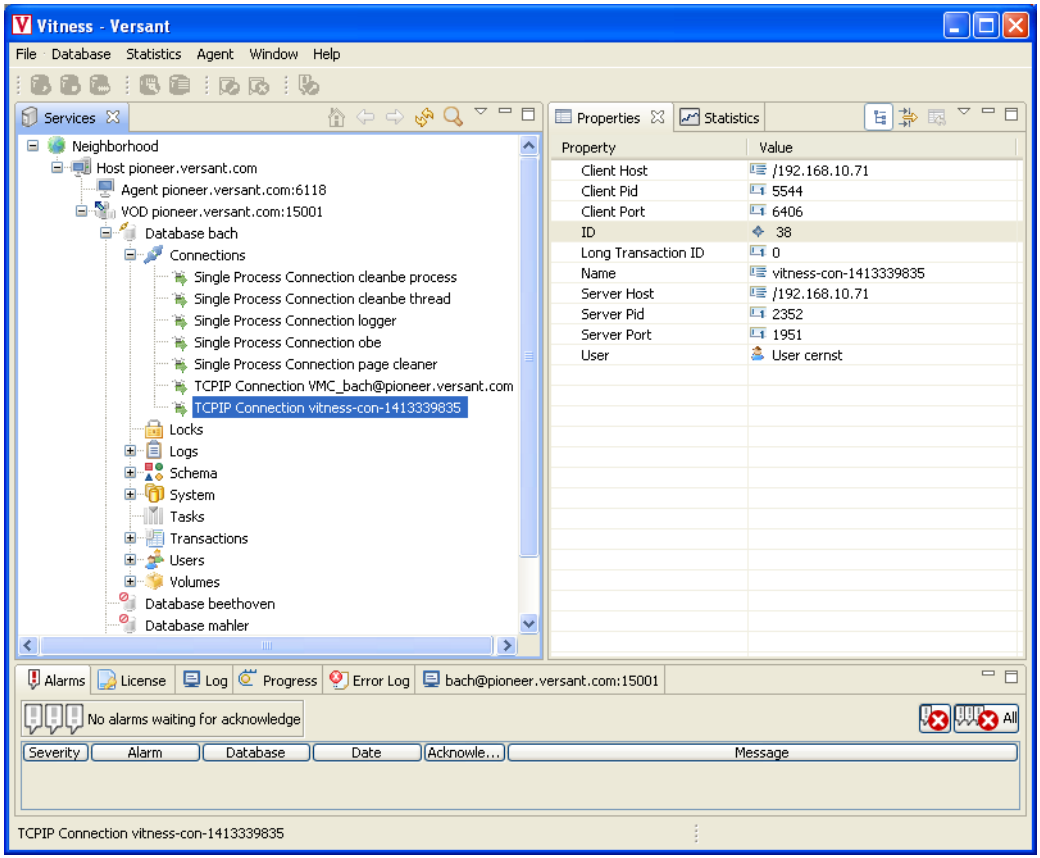


Figure 2.17. Active connections display view

2.8. Transactions

The transactions in the Service Browser View and Transaction Property View reports information on all active transactions. The transaction ID, the connection ID, the transaction name, the associated flags and the total number of locks the selected transaction is holding are reported.

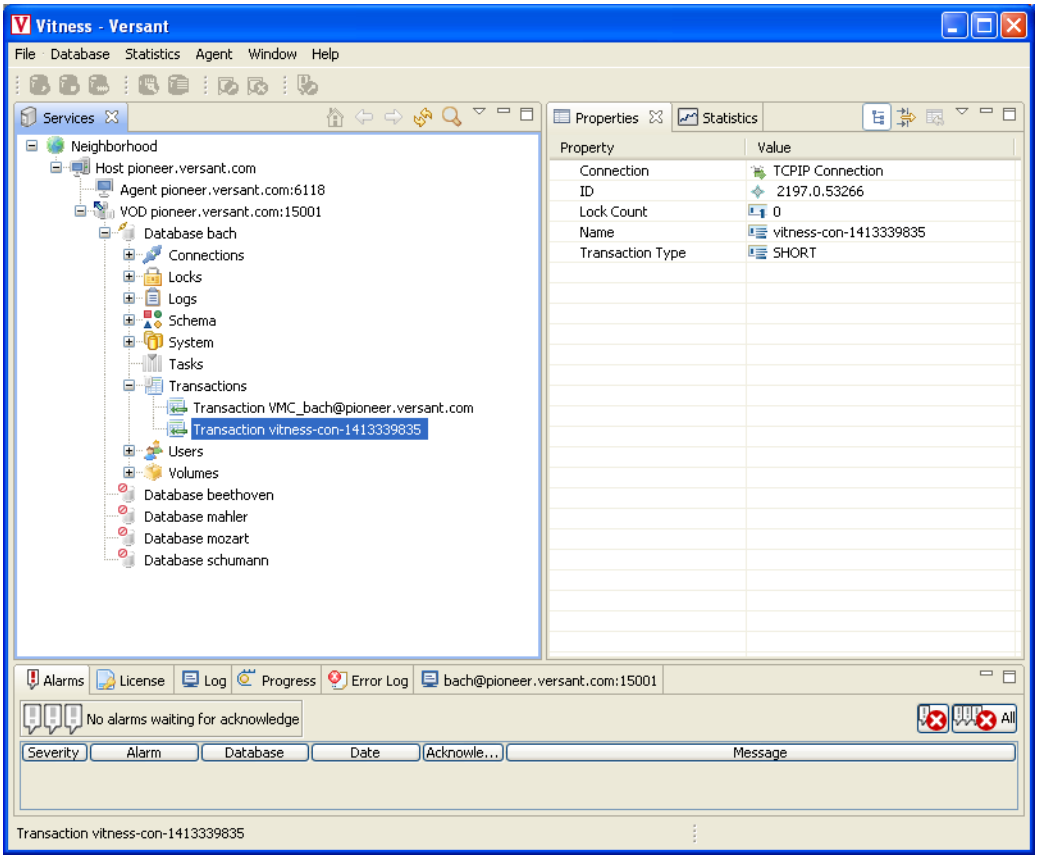


Figure 2.18. Active transactions display view

2.9. Locks

The locks in the Service Browser View and Lock Property View shows information on the locks maintained on the monitored database by each active transaction. For each lock, the lock mode, the LOID (logical object identifier) of the locked object, the class of the locked objects, the lock flags and the transaction ID are reported.

It is possible to display selected types of locks among those maintained on the monitored database by checking the filters in the top area of the locks display panel.

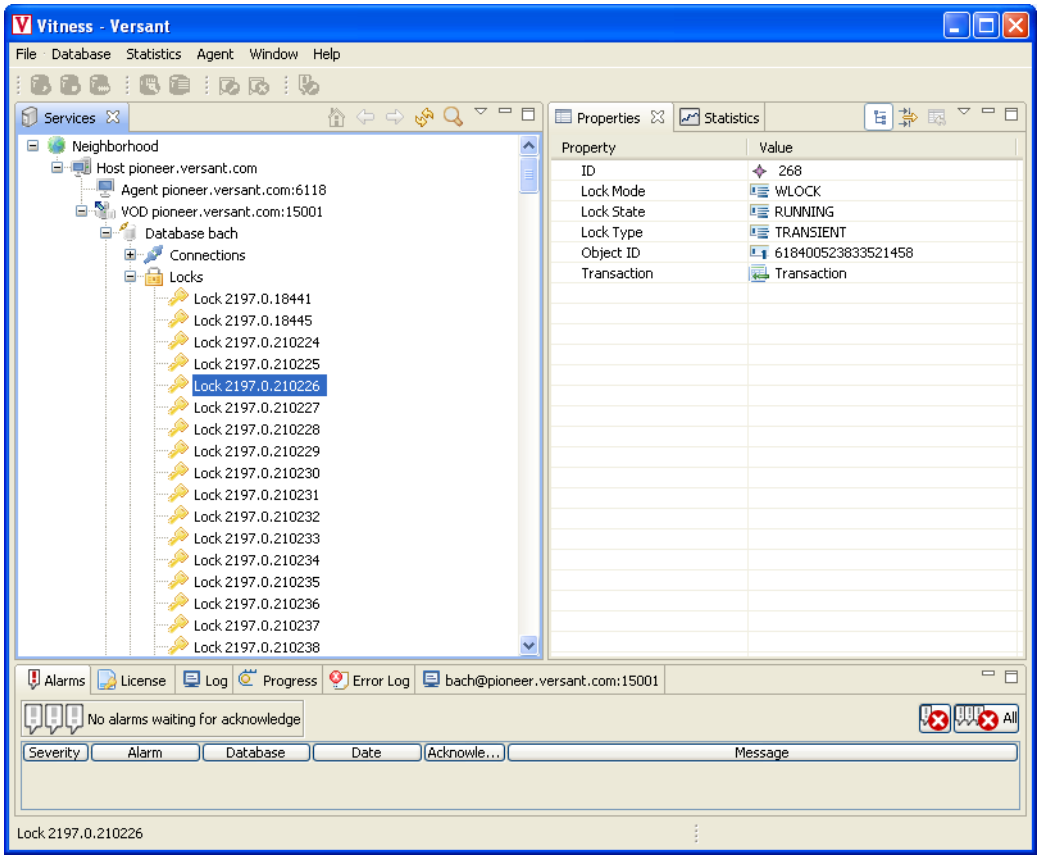


Figure 2.19. Locks display view

2.10. Users

The users in the Service Browser View and User Property View reports information on all active users. For each user, the creation time, name, user privilege and user type are reported.

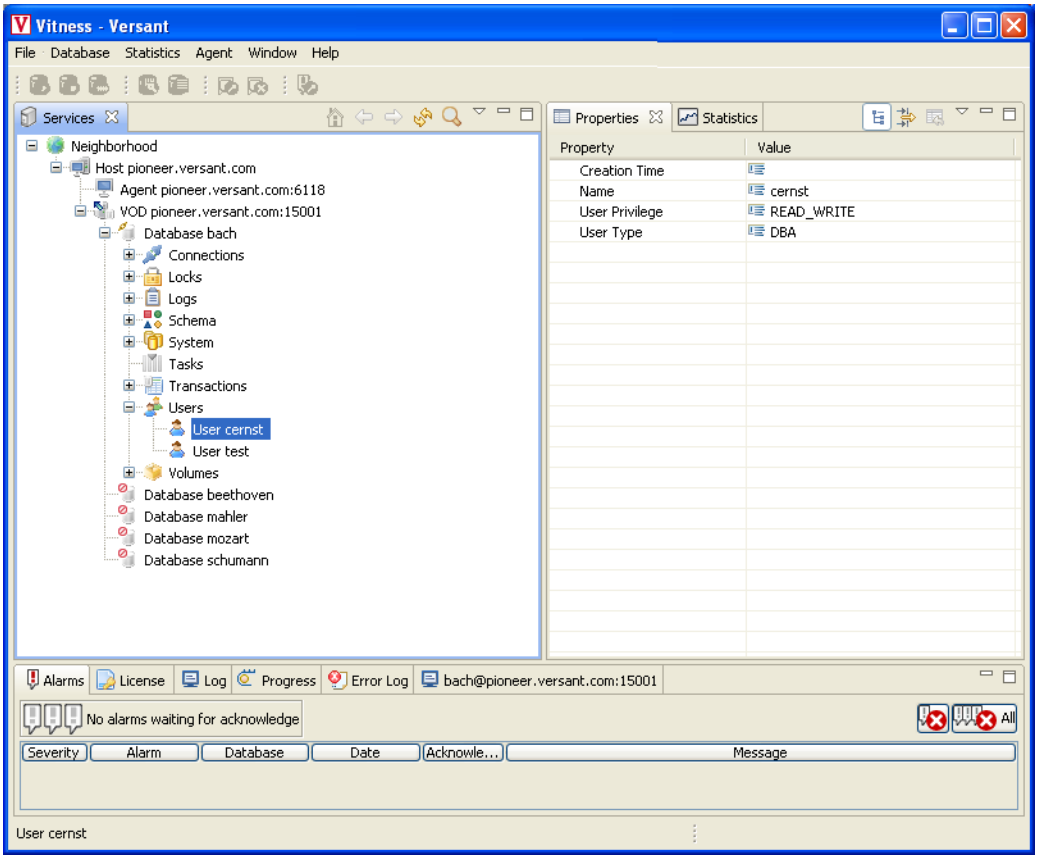


Figure 2.20. Users display view

2.11. Tasks

The tasks in the Service Browser View and Task Property View shows information on all of the tasks. The current count, end time, error code, ID, name, parameters, parent task, start time, sub tasks, task level, task state, task type, total count and user of the selected task are reported.

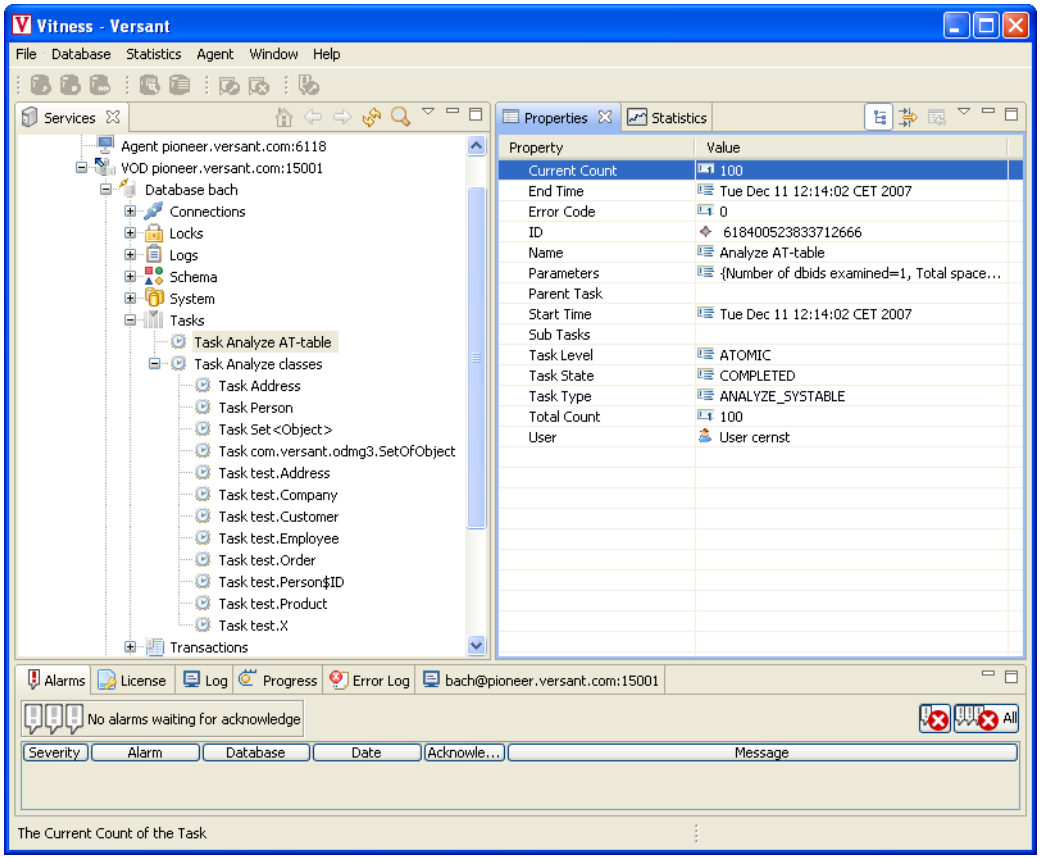


Figure 2.21. Tasks display view

2.12. Schema

The schema in the Service Browser View and Schema Property View provides information on all of the classes and attributes present in the schema. The name, sub classes and super classes are reported for the selected class.

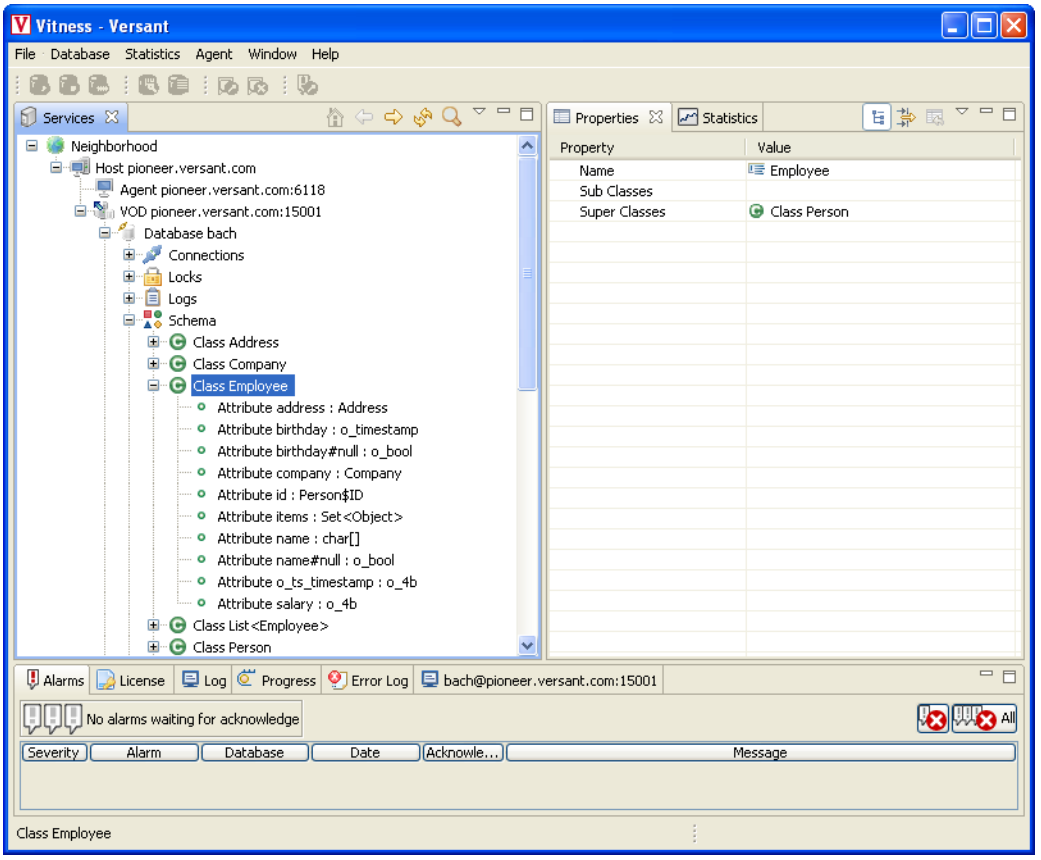


Figure 2.22. Schema display view

2.13. System

The system in the Service Browser View and System Property View provides information on all database system segements including the AT Table. It displays classes, free bytes, free pages, instance count, name, percentage empty, used bytes and used pages for the selected database system segment. For a description of the properties displayed for the AT (system) table refer to Section 6.3, “System Table De-fragmentation” [p. 136].

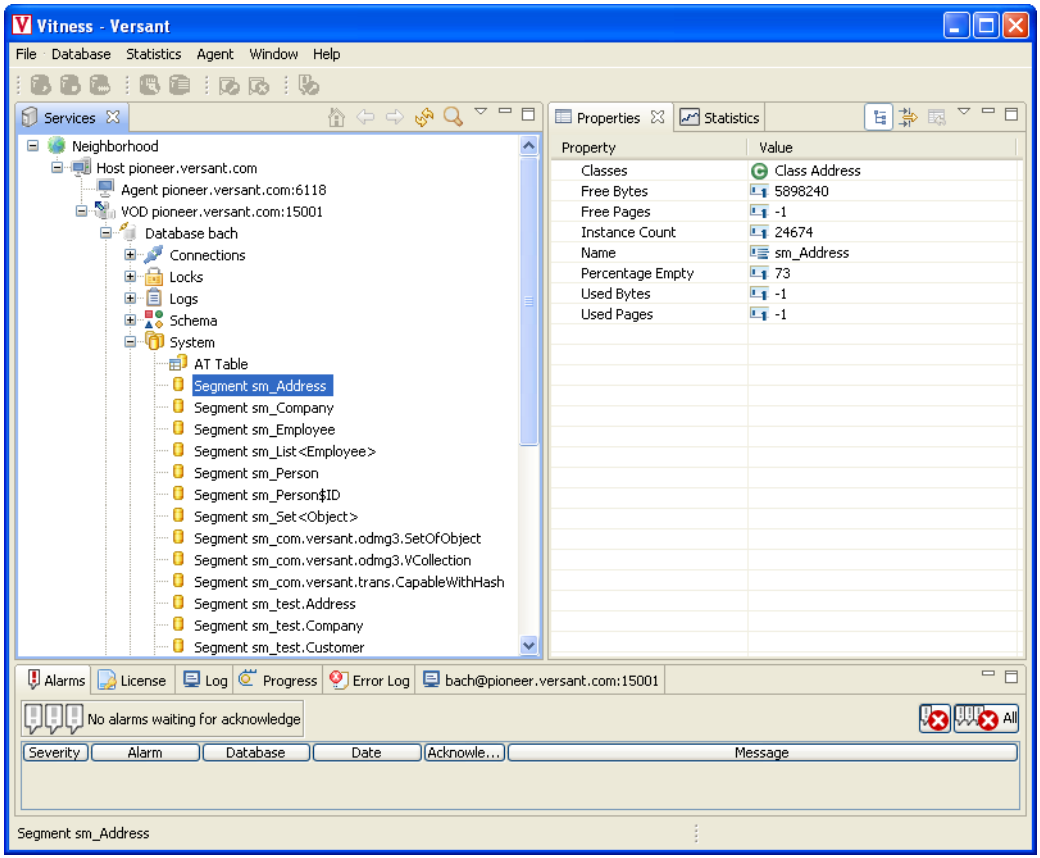


Figure 2.23. System display view

2.14. Alarms

Any relevant event generated by a monitored database is reported to the Vitness Database Monitor and displayed in the Alarms View. If the SNMP adaptor is activated (refer to Chapter 4, *VOD Agent SNMP Support* [p. 75]), to trigger a specific SNMP trap.

There are two categories of alarm notifications in the Vitness Database Monitor, filter-based alarms and threshold-based alarms. Alarm notification names have a prefix to indicate to which catagory they belong. Filter-based alarms have the prefix `FLT_` and threshold-based alarms have the prefix `THR_`.

Filter-based alarms are generated by those events that can be detected by filtering and analysing the messages in the LOGFILE of the monitored database, logged by the Versant Object Database server and by the Versant Object Database utilities. Such messages can be related to server errors or administrative actions. The following table lists the filter-based alarms.

Table 2.1. Filter-based alarms

Alarm name	Alarm type
FLT_VirtualSystem	Virtual System Errors, 0001-0899
FLT_SystemLevel	System Level Errors, 0900-0999
FLT_VersantServer	Versant Object Database Server Errors, 1000-2999
FLT_DbStateChange	The database state has been changed
FLT_VolumeAdded	A new volume has been added to the database
FLT_DbStarted	The database has been started
FLT_DbStopped	The database has been stopped

For any details on Versant Object Database error codes and messages, please refer to the Versant Object Database documentation.

Threshold-based alarms indicate such conditions as database full and log file expanded. The following table lists the threshold-based alarms.

Table 2.2. Threshold-based alarms

Alarm name	Alarm type
THR_DatabaseFull	The database has reached the threshold set for the *db full* condition
THR_LogicalLogExpanded	The Logical Log volume has been expanded
THR_PhysicalLogExpanded	The Physical Log volume has been expanded
THR_VolumeFull	The specified data volume has reached the threshold set for the *volume full* condition

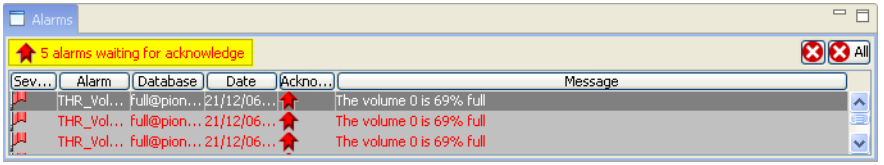


Figure 2.24. The alarm notification panel

For each alarm, on a single line of the panel, the following information is displayed.

- The alarm severity
- The alarm name
- The source database
- The alarm timestamp
- The acknowledge status
- The alarm message

Some alarms need to be acknowledged. Due to their importance, it is considered necessary that the administrator of the database is aware that such events have occurred. This is shown in the Alarms view with a flashing message and with a red arrow in the acknowledge status column of the alarm.

Double-clicking on one of the alarms brings up a detail window from which it is possible to acknowledge the alarm if necessary.

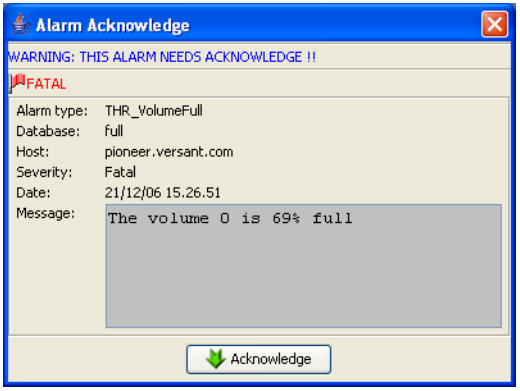


Figure 2.25. Alarm detail window

It is possible to configure the Vitness Database Monitor behavior with respect to alarms from the **Alarms configuration** window. To configure alarms, select **Configure Alarms** from the main **Agent** menu, or press the  button in the toolbar.

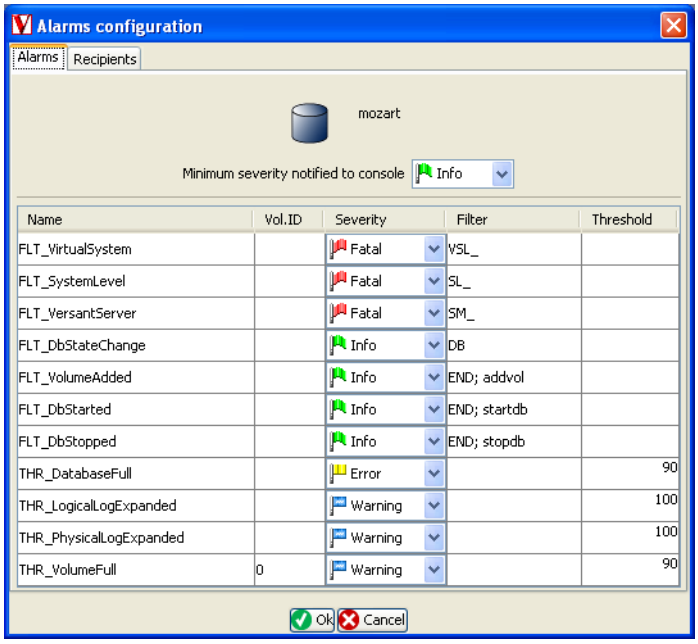


Figure 2.26. Alarms configuration window

The configuration is performed independently for each monitored database. For each type of alarm, the severity level and the filter or threshold condition can be set. Such settings are stored permanently in the `agent.xml` configuration file of the VOD Agent that is monitoring that database.

The alarms can be logged to a file. Alarm logging can be enabled and configured in the **Logging Alarms** view in the **Preferences** window. Refer to Section 3.1.6, “Logging Parameters” [p. 69] for a description of the required parameters.

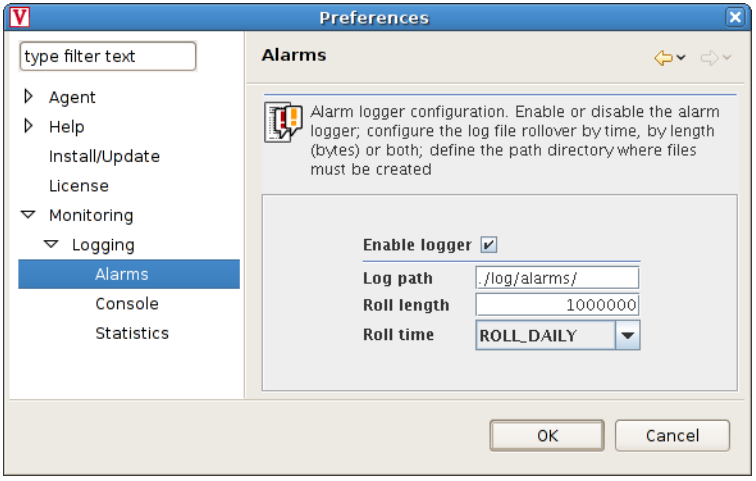


Figure 2.27. Alarm logging configuration

2.14.1. Alarm notification by e-mail

Alarms can be notified by e-mail to a list of interested recipients. The configuration parameters for this feature are described in Section 3.1.5, “Alarm Notification Parameters” [p. 66].

The **Alarms configuration** window has a tabbed panel to configure the severity level of the alarms to be notified by e-mail and the list of recipients. Choose **Configure Alarms** from the main **Agent** menu, or press the button in the toolbar. Select the **Recipients** tab.

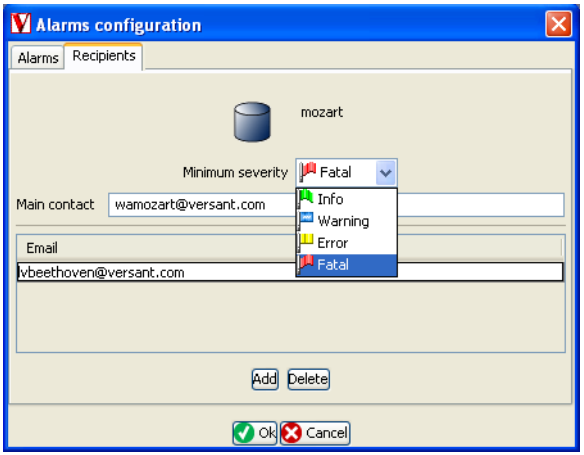


Figure 2.28. Alarm notification by e-mail

Chapter 3. The VOD Agent

The VOD Agent collects data from the databases and makes them available to the Vitness Database Monitor or, through a protocol adaptor, to a SNMP management station as a Management Information Base (MIB). (MIB is explained in Section 4.5, “The Versant Management Information Base (MIB)” [p. 101].)

Several aspects of the database structure and behavior are monitored, and relevant events are reported as alarms to the Vitness Database Monitor and, if the SNMP protocol adaptor is enabled, a specific trap is triggered and sent to management stations.

3.1. VOD Agent Configuration

The configuration of the VOD Agent is defined in an XML file with the default name `agent.xml` located in the `bin` directory of your VOD Agent installation. The elements specified in the configuration file are related to communication parameters, logging activity, standalone permanent monitoring of databases, alarm notification, security policy, and SNMP protocol adaptor activation.

The parameters for standalone permanent monitoring of databases are mainly related to SNMP monitoring and are described in Section 4.2, “Configuring Vitness as an SNMP Agent” [p. 78].

The parameters related to alarm notification are described in Section 3.1.5, “Alarm Notification Parameters” [p. 66].

The security policy parameters are described in Section 3.2, “VOD Agent and Java Security” [p. 72].

The parameters related to the SNMP protocol adaptor are described in Chapter 4, *VOD Agent SNMP Support* [p. 75].

3.1.1. Little-endian and Big-endian Platforms

Normally the VOD Agent automatically detects whether the platform it is running on is little- or big-endian. It is possible, however, that a particular platform is not properly recognized with respect to byte ordering with the consequence that the VOD Agent cannot properly read the monitored database file structure.

In such a case, it is possible to specify which byte ordering the current platform uses by setting the optional `bigEndian` parameter of the `<monitor>` element in the `agent.xml` file to the appropriate `true` or `false` value.

```
<monitor bigEndian="false">
```

3.1.2. Raw Devices

If the databases to be monitored have volumes on raw devices that cannot be directly accessed as normal block devices the VOD Agent needs to know how to replace the character device with the proper corresponding block device. The optional `<monitor>` element in the `agent.xml` configuration file allows you to define such a correspondence.

The `raw-replacement` parameter of the `<raw-devices >` element is used to specify which block device the VOD Agent must use instead of the related character device. The character device is defined by the `raw` parameter. For example, for Solaris SCSI disks, the following lines specify the character device path and the related block device.

```
<monitor>
  <raw-devices raw="/dev/rdsk" raw-replacement="/dev/dsk"/>
  . . .
</monitor>
```

In this case, each database volume whose path begins with `/dev/rdsk` will be accessed using the related block device with the same name whose path begins with `/dev/dsk`.

Please refer to your platform-specific documentation for more information on disk device configuration.

3.1.3. Internationalization and File Encoding

The VOD Agent can be affected by local file encoding such that it might not be unable to properly read the database volumes on OS platforms localized for non-Latin languages such as Japanese, Chinese, Korean, etc. In such cases it is necessary to inform the JVM about the correct file encoding to use.

The file encoding is defined by a system property that can be set on the command line that starts the VOD Agent in the startup script generated during the installation of the product (Section 1.2, “Installation and Start-up” [p. 3]).

```
-Dfile.encoding=encoding
```

Possible values for the `file.encoding` property value *`encoding`* are given in the following table.

Table 3.1. Platform specific file encoding

Default Encoding	OS Platform
CP1252	Windows
ISO8859_1	Solaris

For other platforms, please refer to your platform specific documentation. More information on Java internationalization support can be found at http://java.sun.com/javase/technologies/core/basic/intl/ .

3.1.4. VOD Agent Communication Parameters

The VOD Agent uses RMI to communicate with the Vitness Database Monitor and SLP (Service Location Protocol) to discover the VOD Agent and Versant Object Database. Before communication can be established between the two components the Vitness Database Monitor must be able to discover where a VOD Agent is.

3.1.4.1. Service Location Protocol—SLP

The *Service Location Protocol* (SLP) is a network protocol that provides automatic detection of devices and the services offered by the devices. SLP is designed to scale from small to enterprise level networks. The protocol is formally defined in the Standard Tracks document RFC 2608.

A common implementation of the Service Location Protocol is *OpenSLP* . This is used on many UNIX systems.

Vitness uses *jSLP* , an opensource SLP implementation in pure Java. This is enabled in the VOD Agent by default.

The jSLP Configuration File

Certain jSLP parameters for the VOD Agent are configured in the file `jslp.properties` in the `bin` directory where you installed the VOD Agent. In nearly all cases, the Vitness Database Monitor will be able to locate the VOD Agent instances and Versant Object Database servers in the network. In certain circumstances, you may find it necessary to modify the entries in this file.

Following is an example `jslp.properties` file. Of particular interest are the entries for `net.slp.interfaces` and `net.slp.port`. These are described, briefly, following the example file.

Example 3.1. Example `jslp.properties` file

```
##
## Comma separated list of IP addresses of the interfaces on which jSLP
## should listen for SLP messages. Currently, jSLP only listens to the
first
## address. You might have to set this property on multi-homed machines
and on
## linux if your JVM thinks your machine's IP is 127.0.0.1
##
#net.slp.interfaces=
##
## number that denotes the (non-standard) port where jSLP is going to
## operate on. Note that this prevents interoperability with other SLP
## entities running on the standard port, if multicast convergence is
used.
##
##net.slp.port=427
##
## predefined scopes for the SA
##
#net.slp.useScopes
##
## predefined DA addresses
## default: none
##
#net.slp.DAAddresses
##
## perform no active or passive DA discovery. Only valid if
#net.slp.DAAddresses are defined.
## default:       false
##
#net.slp.noDADiscovery
##
## wait time for initial DA discovery etc.
## default: 1000
##
#net.slp.waitTime
##
## trace traffic to DA
## default: false
```

```
##
#net.slp.traceDATraffic
##
## trace messages
## default: false
##
#net.slp.traceMsg
##
## trace dropped messages
## default: false
##
#net.slp.traceDrop
##
## trace registrations / deregistrations
## default: false
##
#net.slp.traceReg
##
## TTL for multicast messages. Note: decreasing this value will lead to
## localized query results and peers at different locations in the network
## might get different results
## default: false
##
#net.slp.multicastTTL
##
## total timeout for multicast convergence in mSec.
## default: 15000
##
#net.slp.multicastMaximumWait
##
## timeouts for the rounds during multicast convergence. Note that the
## number of timeouts affects the maximum total number of rounds for
multicast
## convergence.
## default: 500,750,1000,1500,2000,3000
##
#net.slp.multicastTimeouts
##
## Number of mSecs until jSLP stops waiting for a reply to a UDP request
## message and timeframe for retransmissions of failed UDP messages.
## default: 5000
##
```

```
#net.slp.datagramMaximumWait
##
## timeouts for the retransmissions of failed UDP messages. Note that the
## number of timeouts does NOT affect the maximum number of
retransmissions.
## This number is limited by net.slp.datagramMaximumWait.
## default: 3000,3000,3000,3000,3000
##
#net.slp.datagramTimeouts
##
## maximum size of a UDP datagram in Bytes
## default: 1400
##
#net.slp.MTU
##
## enable security
## default: false
##
#net.slp.securityEnabled
##
## a comma separated list of SPIs to use if security is enabled.
## default: none
##
#net.slp.spi
##
## the location of the private key in DER format for SPI SPI
## default: none
##
#net.slp.privateKey.SPI
##
## the location of the public key in DER format for SPI SPI
## default: none
##
#net.slp.publicKey.SPI
```

`net.slp.interfaces`. The value for the `net.slp.interfaces` entry is a list of IP addresses of the interfaces where jSLP should listen for SLP messages.

`net.slp.port`. The value for the `net.slp.port` entry is the port jSLP will use to listen. This is needed only if you are not using the standard jSLP port number 427.

More Information

You can find much more information about Service Location Protocol (SLP), OpenSLP, and jSLP in the Internet. A place you can start is the Wikipedia entry for Service Location Protocol. The OpenSLP and jSLP projects are both hosted at Sourceforge.net®. The Web links are OpenSLP and jSLP.

3.1.4.2. RMI Communication

The VOD Agent can be configured by setting the appropriate communication parameters in the `agent.xml` file.

```
<rmi rmiDiscoveryMode="MULTICAST ">
<rmi-registry external="false" port="6118"/>
      <rmi-discovery>
       <multicast ip-address="230.0.0.1" port="4446"/>
       <unicast port="5000" portRange="10"/>
      </rmi-discovery>
</rmi>
```

Figure 3.1. VOD Agent communication parameters

The `<rmi>` element has one parameter, `rmiDiscoveryMode` , used to specify which discovery methods the VOD Agent should use. The possible values are given in the following list.

- `MULTICAST`
- `REGISTRY`
- `MULTICAST_AND_REGISTRY`
- `NONE`

The default value is `MULTICAST`.

The `<rmi-registry>` element defines the parameter required for the RMIRegistry based on discovery method.

Table 3.2. VOD Agent `<rmi-registry>` parameters

Parameter	Value	Default
`external`	`true` or `false` to enable or disable the use of an external RMIRegistry process	`false`
`port`	The port used for RMI communication	`6118`

If the `external` parameter is set to `false` the VOD Agent will automatically start the RMIRegistry at start up. This requires that no other instances of the RMIRegistry are already running on the server using the same port, otherwise the VOD Agent logs an error message and exits.

If the `external` parameter is set to `true` the VOD Agent will try and use an external, already running RMIRegistry process. In this case the registry must have been started in an environment where the `CLASSPATH` environment variable includes the Vitness jar file.

The `<rmi-discovery>` element defines the parameters required for the multicast discovery method.

Table 3.3. VOD Agent `<multicast>` and `< unicast>` parameters

Element	Parameter	Value	Default
`<multicast>`	`ip-address`	Multicast group ID. A class “D” IP address in the range 224.0.0.0 - 239.255.255.255. (The IP address 224.0.0.0 and the IP addresses in the range 239.0.0.0 to 239.255.255.255 are reserved for site-local “ administratively scoped” applications, and should not be used.)	`230.0.0.1`
`<multicast>`	`port`	The number of the UDP port used for multicast	`4446`
`<unicast>`	`port`	The number of the first UDP port used for unicast	`5000`
`<unicast>`	`portRange`	The range to scan for an available port starting from the first one	`10`

Any changes in the port and address parameters must be reflected in the same parameters of the Vitness Database Monitor configuration file (refer to Section 2.1.1, “VOD Agent and Versant Object Database Discovery” [p. 8]).

If the VOD Agent is used exclusively as an SNMP agent, and never accessed by a Vitness Database Monitor, both of the discovery methods can be disabled by setting the `rmiDiscoveryMode` parameter to `NONE`.

For further information on the multicast communication pattern used in Java, please refer to the JDK documentation.

3.1.5. Alarm Notification Parameters

The VOD Agent forwards alarms to the Vitness Database Monitor which displays them in the **Alarm** pane (refer to Section 2.14, “Alarms” [p. 51]). The VOD Agent can be configured to also send alarm notifications by e-mail to a list of interested recipients.

Notification by e-mail is optional and care must be used when configuring such a feature. It is recommended that a high enough severity level is set in order to avoid a message flood generated by irrelevant alarms that do not necessarily indicate a serious alarm condition.

The alarm notification related parameters are specified in the `< agent-alarm>` element of the `agent.xml` file.

Example 3.2. Alarm notification configuration

```
<agent-alarm minAcknowLevel="3"
    smtp-host="mail.wind.it.net"
    smtp-user="vitness"
    smtp-password="topsecret"
    smtp-sender="vitness@database-host.com">
  <alarm-db dbname="demo" minSeverityLevel="0">
  . . .
    <recipient-container minSeverityLevel="3">
      <to-recipient email="your_address1@host.com"/>
      <recipient email="your_address2@host.com"/>
    </recipient-container>
  </alarm-db>
</agent-alarm>
```

Only a subset of these parameters are intended to be modified directly by the user. All the others should only be modified using the Vitness Database Monitor.

The parameters and their values are described in the following table.

Table 3.4. Alarm notification parameters

Parameter	Value	Default
`minAcknowLevel`	The minimum severity level that requires alarm acknowledge in the Vitness Database Monitor.	`2` (Error)
`smtp-host`	The SMTP mail server host name	
`smtp-user`	The SMTP user for the mail server	
`smtp-password`	The SMTP user password for the mail server	
`smtp-sender`	The SMTP sender e-mail address	
`minSeverityLevel`	The minimum severity level of notified alarms for this database	`0` (Info)

For each database it is possible to specify the minimum severity level that is notified via e-mail to the list of recipients defined in the `< recipient-container>` element:

Table 3.5. Alarm e-mail notification parameters

Parameter	Value	Default
`minSeverityLevel`	The minimum severity level of alarms notified by e-mail for this database	`2` (Error)
`to-recipient email`	The e-mail address of the main recipient of the alarm notification, used for the TO: address	
`recipient email`	The e-mail address of the secondary recipients of the alarm notification, used for the CC: address	

The severity levels are defined as follows.

Table 3.6. Alarm Notification Levels

Numeric	Level
`0`	Information
`1`	Warning
`2`	Error
`3`	Fatal

To activate e-mail notification it is necessary that the SMTP host and the main recipient are both defined. If any of the two parameters are not present or not defined, e-mail notification is disabled.

Appropriate security permission for accessing the SMTP server must be set in the `java.policy` file (refer to Section 3.2, “VOD Agent and Java Security” [p. 72]).

If the SNMP adaptor is enabled (refer to Section 4.2, “Configuring Vitness as an SNMP Agent” [p. 78]), the main recipient e-mail address and the minimum severity level can be set and queried through the `vsntDbContact` managed object. The syntax of the database contact is as follows.

```
[mail-address][N]
```

The value *N* is the minimum severity level. When the database contact is modified with an **SNMP-set** command, the VOD Agent checks the validity of the string. If the supplied string is not valid the change is refused without notice. It is recommended that after attempting to set its value an **SNMP-set** command is issued to verify the change has been accepted.

3.1.6. Logging Parameters

The VOD Agent logs its activity to a file. The related configuration parameters are specified in the `<sys-logger>` element of the `agent.xml` file.

Example 3.3. VOD Agent logging configuration

```
<sys-logger
date-format="EEE MMM dd HH:mm:ss z yyyy"
      log-filename="./log/agent"
log-level="DEBUG"
      log-name="VMCLogger"
 log-path="./log/agent/"
log-rolllimit="1000000"
log-rolltime="ROLL_DAILY"
/>
```

The parameters and their values are described in the following table.

Table 3.7. VOD Agent logging parameters

Parameter	Value	Default
`date-format`	The format of the timestamp of a log entry (for a description of the time format, refer to Section 3.1.6.1, "Time Pattern" [p. 70], below)	`EEE MMM dd HH:mm:ss z yyyy`
`log-filename`	The base name of the log file	`agent`
`log-level`	The log mask (`DEBUG`, `INFO`, `WARNING`, `ERROR`, `FATAL`) set to accept messages at or above the specified level	`INFO`
`log-name`	The name of the Logger class (internal use only)	`VMCLogger`
`log-path`	The directory for the log files	`./log/agent`
`log-rolllimit`	Maximum size in bytes for the log file before it is rolled over. It cannot be less than 1000 (1K). The -1 value disables the size based log rotation (refer to Section 3.1.6.2, "Log File Rollover" [p. 71], below)	`1000000`
`log-rolltime`	The log roll over frequency (`ROLL_MINUTELY`, `ROLL_HOURLY`, `ROLL_DAILY`, `ROLL_MONTHLY` , `ROLL_INACTIVE`). Determines how frequently the log file will be rotated (refer to Section 3.1.6.2, "Log File Rollover" [p. 71], below)	`ROLL_DAILY`

3.1.6.1. Time Pattern

To specify the format of the timestamp a *time pattern* string is used. This string must conform to the date formatting specifications used by the `java.text.SimpleDateFormat` Java class (refer to the JDK documentation for details). In this pattern, all ASCII letters are reserved as pattern letters which are defined in the following table. The count of pattern letters determines the format.

Table 3.8. Log entry timestamp format specification

Symbol	Meaning	Presentation	Example
G	era designator	Text	AD
y	year	Number	1996
M	month in year	Text & Number	July & 07
d	day in month	Number	10
h	hour in am/pm (1~12)	Number	12
H	hour in day (0~23)	Number	0
m	minute in hour	Number	30
s	second in minute	Number	55
S	millisecond	Number	978
E	day in week	Text	Tuesday
D	day in year	Number	189
F	day of week in month	Number	2 (2nd Wed in July)
w	week in year	Number	27
W	week in month	Number	2
a	am/pm marker	Text	PM
k	hour in day (1~24)	Number	24
K	hour in am/pm (0~11)	Number	0
z	time zone	Text	Pacific Standard Time
'	escape for text	Delimiter	
"	single quote	Literal	'

3.1.6.2. Log File Rollover

When the `log-rolllimit` parameter is set, the log file will be rotated when the specified size is reached. When the file rotates, the current file is closed a new file is opened. The name of the file written to will actually be `agent.`*`n`* where *`n`* is an integer. (The file base name, `agent` here, is the default. The base name can be changed with the `log-filename` parameter. Refer to Section 3.1.6, “Logging Parameters” [p. 69], above.) The first logfile is `agent.1`, the second `agent.2`, etc. When starting, if the log files `agent.1` through `agent.`*`n`* exist, the VOD Agent will begin writing to `agent.`*`n+1`*.

When the `log-rolltime` parameter is set, the log file will be rotated with the specified frequency. The name of the file written to will actually be `agent`*`date`* where *`date`* is the date formatted as `yyyy.MM.dd-HH.mm.ss` .

If both rotation criteria are set the log file will be rotated on the basis of both conditions, whichever occurs first.

3.2. VOD Agent and Java Security

The VOD Agent is a Java application executed by a Java Virtual Machine (JVM), and it is subject to the security features of the Java Runtime Environment (JRE), with respect to accessing system resources.

Security policies for accessing system resources are specified in a `java.policy` file.

The location of the `java.policy` file is specified in the `agent.xml` file by the parameter in the `<java-security>` element.

```
<java-security policy-file="./java.policy"/>
```

The following example `java.policy` file allows the JVM in which the VOD Agent is running to access all the necessary resources.

Example 3.4. VOD Agent Java security policy file

```
grant {
permission java.net.SocketPermission "230.0.0.1:*", "connect,accept";
permission java.net.SocketPermission "*:1024-65535", "connect,accept";
permission java.net.SocketPermission "*:80", "connect";
permission java.net.SocketPermission "*:161-162", "connect,accept, listen";
permission java.lang.RuntimePermission "*";
permission java.io.FilePermission "<<ALL FILES>>","read,write";
permission java.util.PropertyPermission "*","read,write";
};
```

The policy file should be edited appropriately if changes are made in the configuration of the VOD Agent with respect to the port used for communication purposes such as agents discovery and SNMP messaging.

If alarm notification by e-mail is configured a line that defines the security permission for the SMTP mail server must be added to the `java.policy` file, as follows.

```
permission java.net.SocketPermission "smtp-host:25",
"connect,accept";
```

Here, `smtp-host` is the SMTP mail server name or IP address and `25` is the standard UDP port for the SMTP protocol.

For further details on Java security, please refer to the related JDK documentation.

3.3. Shutting Down the VOD Agent

You can shutdown a running VOD Agent manually by executing the **agent.sh** command file (**agent.bat** for Windows) with the **-stop** option (i.e., **agent -stop**).

Chapter 4. VOD Agent SNMP Support

The VOD Agent offers an SNMP interface that can be enabled to access the monitored elements using the standard SNMP protocol. For an introduction to SNMP refer to Section 4.1, “Introduction to SNMP” [p. 76], below.

The Vitness SNMP Agent is designed in order to serve the following purposes.

- Equip the relevant resources of a Versant Object Database system to be able to translate information about the resources into a Versant MIB (refer to Section 4.5, “The Versant Management Information Base (MIB)” [p. 101])
- Store and return attribute values after get requests
- Respond to set requests
- Send traps to management applications
- Cooperate with other agents

In general, the following areas are of common concern to Versant Object Database systems and application managers.

Fault management

The surveillance, detection, and processing of conditions that indicate significant changes in resource status, serious enough to require corrective action. The triggering of automated and human responses that correct detected problems, test the reliability of fixes, test for impact on the enterprise as a whole.

Configuration management

The processing of changing information about the number, specific characteristics, and connectivity of the resources in the system

Performance management

The regular collection and correlation of

- Resource workloads
- Capacity utilization
- Bottlenecks indicators

This information is used to

- Monitor trends
- Improve performance and availability
- Anticipate the need for new resources or expanded capacity

Accounting

The tracking of resource usage by individual applications, groups of applications, individual users, and groups of users

Security

The monitoring and control of access to resources and data

The current release of Vitness provides features that can support fault management and performance management with resources monitoring. The Vitness framework also provides a number of tools for administration, and more facilities will be added in future releases. However, Vitness SNMP support is intended more for monitoring Versant Object Database products than for managing them. Vitness SNMP support is most valuable for tracking the status of an entire system of Versant Object Database based applications, to verify normal operations, and to spot and react to potential problems as soon as they are detected.

For purposes of investigating and solving specific issues other tools such as the Vitness Database Monitor can be used to gather statistics and the administration facilities of Vitness and the Versant Object Database utilities for configuring and managing the database servers may be more appropriate. The Vitness SNMP support is designed to query status but not to change system parameters.

4.1. Introduction to SNMP

The Simple Network Management Protocol (SNMP) was originally developed for use as part of the TCP/IP suite as a means for network monitoring. The genesis of SNMP based management is the Internet Engineering Task Force (IETF), an open organization to create standards for the Internet. The initial targets for this effort were TCP/IP routers, TCP/IP gateways, and hosts (i.e., computer systems). However, the SNMP based management approach is inherently generic so that it can be used to manage many types of systems.

The model of systems and network management that is used for SNMP includes the following key elements.

- Management station
- Management agent
- Management information base

- Network management protocol

SNMP management is based on the agent/manager model described in the network management standards maintained by IETF. In this model, a network/systems manager exchanges monitoring and control information about system and network resources with distributed software processes called *agents* through the SNMP protocol.

Any system or network resource that can be monitored or that is manageable through the exchange of information is a *managed resource*. This could be a software resource such as a Versant Object Database database or a hardware resource such as a router.

The management station refers to a node from which managed elements are monitored. Typically, it is a stand alone workstation that is on the same network as the managed elements. other terms used for it include management console, management system, or managing node.

Agents function as “collection devices” that typically gather and send data about the managed resource in response to a request from a manager. In addition, agents often have the ability to issue unsolicited reports to managers when they detect certain predefined thresholds or conditions on a managed resource. In SNMP terminology, these unsolicited event reports are called *trap notifications*.

The means by which resources in a network or in a system may be monitored and managed is to represent such resources as objects. Each object is, essentially, a data variable that represents one aspect of the managed resource. The collection of objects is referred to as a *management information base* or *MIB*. For the management station the objects of the MIB function as a collection of access points to the SNMP agent . These objects are standardized across systems of a particular class. A management station performs the monitoring function by retrieving the value of MIB objects.

The network management protocol used to link the management station and agents is the SNMP. This protocol includes the following key capabilities.

Get
: enables the management station to retrieve the value of objects at the agent

Set
: enables the management station to set the value of objects at the agent

Trap
: enables an agent to notify the management station of significant events

SNMP was designed to be an application level protocol that is part of the TCP/IP protocol suite. It is intended to operate over the user datagram protocol (UDP).

The SNMP reference architecture is shown in Figure 4.1, “SNMP protocol architecture” [p. 78], below.

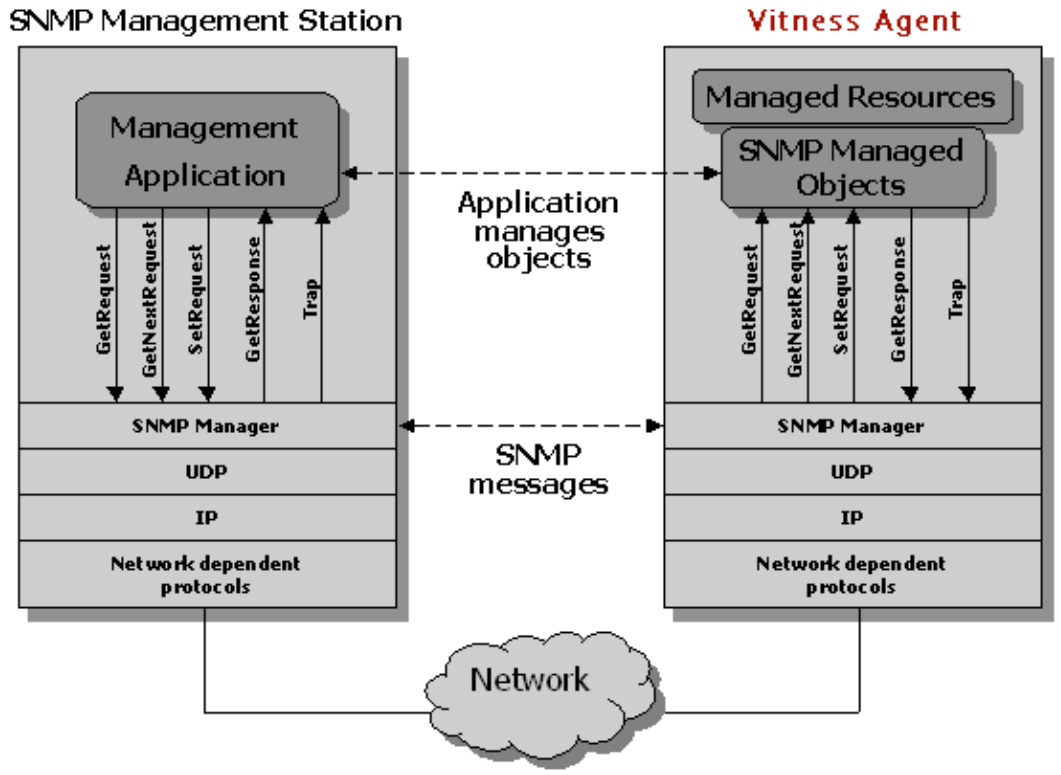


Figure 4.1. SNMP protocol architecture

4.2. Configuring Vitness as an SNMP Agent

The SNMP interface of the VOD Agent is enabled by activating a SNMP protocol adaptor at the start of the VOD Agent. When the SNMP enabled VOD Agent starts up, a cold start (0) generic trap is sent to all the configured destinations (refer to Section 4.5.2, “Traps” [p. 106]).

To activate the SNMP protocol adaptor, the `enabled` parameter of the `<snmp-adaptor>` element in the `agent.xml` configuration file must be set to `true`.

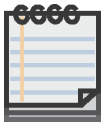
The format of the `<snmp-adaptor>` element in the `agent.xml` configuration file is shown below.

Example 4.1. SNMP adaptor configuration parameters

```
<snmp-adaptor
  enabled="true"
  config-file="./agent-snmp.xml"
  trap-resend-interval="0"
/>
```

The `<snmp-adaptor>` element parameters define the configuration of the SNMP protocol adaptor. The configuration is read by the VOD Agent when it is started. The parameters are summarized in the following table.

Parameter	Value	Default
`enabled`	`true` or `false` to activate or deactivate the SNMP protocol adaptor.	`false`
`config-file`	The SNMP agent configuration file. This is supplied as the `bin/agent-snmp.xml` file.	
`trap-resend-interval`	Number of seconds the agent waits before resending a trap. If the value is `0`, the trap is sent only once. This value only affects repeatable traps (refer to Section 4.5.2, “Versant Traps” [p. 108]).	`5`



By default Vitness SNMP Agent is disabled when Vitness is installed. This means that it must be explicitly enabled after the installation of Vitness.

TheVitness SNMP Agent can be run stand-alone, without the need of a connected Vitness Database Monitor, by configuring which databases are permanently monitored in the `agent.xml` file. This can be done either with the Vitness Database Monitor, as explained in Section 2.2.1, “Standalone Database Monitoring” [p. 15], or editing the configuration file directly. The Vitness SNMP Agent populates the MIB with the managed objects of all the databases being monitoring.

The `<monitor>` element in the `agent.xml` file specifies with `<db>` child elements which databases are to be monitored by the stand-alone agent. The `<db>` element entries have the following format.

Example 4.2. Stand-alone VOD Agent configuration parameters

```
<db
  db-createtime="timestamp"
  db-id="dbid"
  db-name="dbname"
  db-owner="dbowner"
  db-path="dbpath"
  db-type="dbtype"
  db-version="dbversion"
/>
```

The <db> element parameters provide information about the databases to be monitored. The parameters are listed below.

Parameter	Value	Type
db-createtime	The date and time of the database creation	optional
db-id	The database ID	required
db-name	The name of the database	optional
db-owner	The owner (DBA) of the database	optional
db-path	The path of the database	required
db-type	The type of database (group or personal)	optional
db-version	The version (Versant Object Database release) of the database	optional

All attributes of the <db> element must be provided but may have no value (i.e., ""). To activate the monitoring of a database the only attributes that must have a value are the db-id and the db-path. These are the required values in the table above. Additional parameters are optional and may be specified as desired.

```
<monitor>
  <db
    db-createtime=""
    db-id="1"
    db-name=""
    db-owner=""
    db-path="/usr1/local/versant/db/database1"
    db-type=""
    db-version=""
  />
  <db
```

```
    db-createtime=""
    db-id="2"
    db-name=""
    db-owner=""
    db-path="/usr1/local/versant/db/database2"
    db-type=""
    db-version=""
  />
</monitor>
```

If the configuration is made from the Vitness Database Monitor, all the parameters are automatically set (refer to Section 2.2, “Database Monitoring” [p. 11]). If the configuration is made by editing the `agent.xml` file, providing the optional information allows the VOD Agent to properly set the value of the related managed objects in the MIB. Such information can be retrieved using the dblist utility provided with the Versant Object Database. If this information is not provided the related objects in the MIB will have a default value. Any other managed object will be properly set and updated by the Vitness SNMP Agent

As soon as the VOD Agent is started, monitoring of the configured databases is started. If the SNMP protocol adaptor is enabled, the MIB is populated and made accessible to management stations.

4.3. SNMP Agent Configuration

The configuration of the SNMP agent is specified in an XML file named `agent-snmp.xml`. This default location for this file the `bin` directory of your VOD Agent installation. The following describes the entries for this file.

4.3.1. The <properties> Section

The properties node defines some agent properties and initial values for MIB variables.

version

The version number of this agent: 1 for SNMPv1, 2 for SNMPv2c, or 3 for SNMPv3.

encryptPasswordAndCommunity

If set to `yes`, all the unencrypted communities and passwords are changed to an encrypted format and saved to a file when the agent starts. Changes can be made by replacing entries in the file with a new, unencrypted value which will automatically be encrypted again when the agent restarts. If set to `no`, values will not be changed even if they're in encrypted format (but they will still be decrypted by the agent). The length of communities and passwords must be less than 32 characters in order to be successfully encrypted.

readCommunity

A comma-separated list of community names for the SNMP GET/ GETNEXT/ GETBULK operations. If this value is empty, no READ actions are allowed. All community names are case insensitive.

writeCommunity

A comma-separated list of community names for the SNMP SET operation. If this value is empty, no SET action is allowed. All community names are case insensitive.

maxPacketSize

The maximum number of bytes in a packet. Note that the maximum ethernet packet size is 1500 bytes.

useThreadPool

If set to `yes`, a thread pool is used to improve performance. This also means that more system resources will be needed by the agent.

engineID

The SNMPv3 engineID of this agent. If empty, the first IP address found for the hostname of the machine will be used.

engineBoots

The number of times the agent will reboot. Its value is updated whenever the agent restarts under SNMPv3.

informTimeout

The timeout value for the agent to send an INFORM request, in milliseconds.

informRetries

The number of INFORM requests that will be sent after timeout if no response is received.

reloadConfigOID

The OID for reloading this file, which may be any integer. If the SNMP manager issues a SET request using only this OID, it forces the agent to reload the config file.

port

The SNMP agent UDP port number. Its default value is 161.

ipAddress

The IP address of this agent, which is used as one of the fields in SNMPv1 trap. The local IP address will be used if this field is empty.

masterAgentPort

This is the master agent TCP port number.

remoteMasterAgentPort

For subagents, this property specifies the remote master agent's listening port.

remoteMasterAgentIpAddress

For subagents, this property specifies the remote master agent's IP address or host name.

subagentIpAddresses

A comma-separated list of authorized subagent IP addresses. An empty value here means that all IP addresses are accepted.

managerIpAddresses

A comma-separated list of authorized SNMP manager IP addresses. An empty value here means that all IP addresses are accepted.

allowNonV3ReadRequestsForV3Agent

If set to `yes`, an SNMPv3 agent will process SNMPv1/v2c non-set requests (GET/GET_NEXT, GET_BULK). The default value is `no`.

allowNonV3SetRequestsForNonV3Agent

If set to `yes`, an SNMPv3 agent will process SNMPv1/v2c ForV3Agent SET requests. The default value is `no`.

noCommunityNameCheck

If set to `yes`, the agent will ignore community names in ck checking permissions. The default value is `no`.

communityViewEnabled

If set to `yes`, the community name is tied to views. The default value is `no`. This option only takes effect if the agent is from a version prior to SNMPv3.

system.sysObjectID

This value is used as the Enterprise OID in an SNMPv1 trap.

system.sysLocation
system.sysContact
system.sysObjectID

These are static values for MIB objects. Use the format `GroupName.ScalarObject`. Their values will be updated and saved if changed by SNMP SET. For example, if the manager issues a SNMP SET request and changes sysContact successfully, then the sysContact value in the config file will also be updated and saved.

If the version number is 3, then the agent is an SNMPv3 agent. It will then reject all SNMPv1/v2c requests for security concerns. If the version number is 2, however, the agent will still handle SNMPv1 requests.

The agent's config file can be reloaded at run time. The SNMP manager sends a SET request whose OID is the value of `reloadConfigOID`, and varbind's value can be any integer. Upon receiving this request, the agent will reload its config file and re-initialize its internal states. For example, if the value of `reloadConfigOID` is `.1.3.6.1.2.1.5000.1.0`, the SNMP manager can force the agent to reload config with the following request.

```
java snmpset localhost .1.3.6.1.2.1.5000.1.0 i 1
```

To disable this feature, just leave `reloadConfigOID` blank in the config file.

The attributes `managerIpAddresses` and `subagentIpAddresses` are comma-separated lists of hosts, each of which can be expressed in any of the following formats:

- A numeric IP address, such as 192.168.2.20,
- A host name, such as server.somewhere.com,
- or CIDR format subnet notation, such as A.B.C.D/16. For example,
  - 192.168.1.0/24 includes all addresses between 192.168.1.0 and 192.168.1.255,
  - 192.168.1.0/25 includes all addresses between 192.168.1.0 and 192.168.1.127,
  - and 192.168.1.128/25 includes all addresses between 192.168.1.128 and 192.168.1.255.

4.3.2. The `<trapSink>` Section

This section defines the properties of trap receivers. An agent will send traps to all defined trap receivers.

SNMPv1 and SNMPv2 TrapSink

`hostname`

The host name or IP address of the trap receiver

`port`

The port number of the trap receiver

`community`

The trap receiver's community name

version
The trap receiver's SNMP version number

isInform
Set this parameter to `yes` to send SNMP INFORM request instead of traps. INFORM requests are more reliable than traps.

snmpV3TrapSink

hostname
The host name or IP address of the trap receiver

port
The port number of the trap receiver

isInform
Set this parameter to `yes` to send SNMP INFORM requests instead of traps

userName
Set this parameter to one of the user names in the trap receiver's user list

auth
The authentication algorithm used, either `MD5` or `SHA`

authPassword
An authentication password

priv
The privacy algorithm used, either `DES` or `AES`. The default is `DES`.

privpassword
A privacy password

In the snmpV3TrapSink section, the security level is determined by authPassword and privPassword . If both are empty strings, the security level is set to noAuthNoPriv. If only privPassword is empty, the security level is set to authNoPriv. If neither of them is an empty string, the security level is set to authPriv.

For example, assume that three trapsinks are defined: one SNMPv1, one SNMPv2, and one SNMPv3. If the agent sends an SNMPv2 trap, it will be sent to all three trapsinks. It will be converted to an SNMPv1 trap before it is sent to the SNMPv1 trapsink. If the agent sends an SNMPv1 trap, it will be converted to an SNMPv2 trap before being sent to the SNMPv2 and SNMPv3 trapsinks.

4.3.3. The <user> Section

This section defines the properties of authorized users.

name
User name. Two users cannot have same name.

auth
The authentication algorithm used, either MD5 or SHA.

authPassword
Authentication password.

priv
The privacy algorithm used, either DES or AES. The default is DES.

privPassword
Privacy password.

group
The group that this user is associated with. A user can only be associated with one group.

Unlike the snmpV3TrapSink node, the user's security level is determined by the security level of its group rather than the values of authPassword and privPassword. If the security level is authNoPriv, then the privPassword field will be ignored.

4.3.4. The <group> Section

This section defines SNMPv3 group properties.

name
Group name. Two groups cannot have same name.

securityLevel
The group's security level, which must be one of {noAuthNoPriv, authNoPriv, authPriv}

match
Context matching method, which must be set to either prefix or exact. (See below)

`contextPrefix`

If match is set to `prefix`, then context matching only checks whether the context starts with the string contextPrefix. If match is set to `exact`, then the context must be exactly matched.

`readView`

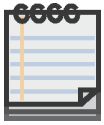
The view associated with this group for “READ” operations such as GET, GETNEXT, and GETBULK.

`writeView`

The view associated with this group for “WRITE” operations such as SET.

`notifyView`

The view associated with this group for notification operations.



SecurityLevel must be one of the set {noAuthNoPriv, authNoPriv, authPriv}. “noAuthNoPriv” means that neither authentication nor encryption is applied to packets. “authNoPriv” means that only authentication is applied to packets. “authPriv” means that both authentication and encryption are applied to packets. All users in a group have the same security level.

4.3.5. The `<view>` Section

This section defines the SNMPv3 VACM view properties.

`name`

View name. Multiple views can have the same name, in this case, the user tied to this view name is associated with all of them.

`type`

View type, which is set to either “included” or “excluded”. A view subtree can be defined as either including or excluding all the object instances that it contains.

`subTree`

Subtree OID. A subtree is a node in the MIB’s naming hierarchy and all of its subordinate elements.

`mask`

A list of ones and zeroes separated by '.' or ':'. A view mask can be defined to reduce the amount of configuration information required for fine-grained access control. Each element in this bit mask specifies whether or not the corresponding sub-identifiers must match when determining whether an OBJECT IDENTIFIER is in this family of view subtrees. Thus a '1' indicates that an exact match must occur in that element, and a '0' indicates that any sub-identifier value matches (a 'wild card').

For example,

```
<view name = "view1"
        type = "included"
        subTree = ".1.3"
        mask = ".1.1"
    />
```

defines a view named view1, which includes all tree nodes whose OIDs start with “.1.3”. To take another example,

```
<view name = "view1"
            type = "included"
            subTree = ".1.3.6.1.2.1"
            mask = ".1.1.1.1.1.0"
            />
```

defines a view which includes all tree nodes whose OIDs start with “.1.3.6.1.2”. The last digit of the mask is 0, which means it does not care about the subtree’s OID at that index. Here’s another example:

```
<view name = "view1"
            type = "included"
            subTree = ".1.3.6.1.2.1.7"
            mask = ".1.1.1.1.1.0.1"
            />
```

In this view, all OIDs matching .1.3.6.1.2.[1, 2, ...].7.* are included. The character ‘*’ represents any series of valid integers separated by periods, and the character ‘?’ represents any single valid OID subelement. Finally, in

```
<view  name = "view1"
      type         = "included"
      subTree   = ".1.3.6.1.2.1"
      mask       = ".1.1.1.0.1.0"
            />
```

all OIDs matching .1.3.6.?.2.* are included, such as “.1.3.6.1.2.1.*”, “.1.3.6.2.2.1.*”, and “.1.3.6.3.2.2.*”.

4.3.6. The <communityView> Section

This section defines the mapping between community names and view names. It takes effect only if `communityViewEnabled` (in the properties section of the config file) is set to `yes`, and it only applies to SNMPv1/v2c agents.

If a community name is tied to a view name, only the subtree(s) defined by the specified view name will be visible to requests made under the community name. Multiple views can have the same name, in this case, the community name tied to this view name is associated with all of them.

`community`
The community name, which must be unique.

`readView`
View name for the read community.

`writeView`
View name for the write community.

4.3.7. The <proxy> Section

When this agent serves as a proxy for other agents, this section defines how SNMP requests are delegated.

`ipAddress`
The IP address or host name of the agent being proxied.

`port`
The listening port number of the agent being proxied.

`included`
If set to `yes`, the agent being proxied handles the requests falling under subTree. Otherwise, the agent being proxied handles all requests except for those falling under subTree. Only one proxied agent of the latter type is allowed in the config file.

`subTree`
Subtree OID(s). You can specify multiple subtrees, separated by commas.

`timeOut`
The timeout value before querying the agent being proxied, in milliseconds.

version
The SNMP version number of the agent. The possible values are {1, 2, 3}. The default value is 1.

readCommunity
The community name for SNMP GET / GET_NEXT/ GET_BULK operations of the agent being proxied. This value does not matter if version number is 3.

writeCommunity
The community name for SNMP SET operations of the agent being proxied. This value does not matter if version number is 3.

userName
(SNMPv3 property) SNMPv3 user name.

auth
(SNMPv3 property) The authentication algorithm used, either `MD5` or `SHA`.

authPassword
(SNMPv3 property) Authentication password.

priv
(SNMPv3 property) The privacy algorithm used, either `DES` or `AES`. The default is `DES`.

privPassword
(SNMPv3 property) Privacy password.

The following example defines an agent being proxied. Its IP address is 127.0.0.1, and it listens on port 200. All the requests falling under subtree 1.3.6.1.2.1 will be handled by this SNMP agent instead of the master agent.

```
<proxy
                ipAddress="127.0.0.1"
                port="200"
                included="yes"
                subTree=".1.3.6.1.2.1"
                timeout="5000"
                version=”1”
                readCommunity="yourReadCommunity"
                writeCommunity="yourWriteCommunity"
                userName=" "
                auth=""
```

```
                authPassword=""
                priv=""
                privPassword=""
                />
```

The following proxy section defines a different type of proxied agent. Because included is set to no this agent handles all SNMP requests except for those with OIDs falling under the subtrees .1.3.1.4.15145.10 and .1.3.1.4.15145.20, which are handled by the master agent itself.

```
<proxy
                    ipAddress="127.0.0.1"
                    port="200"
                    included="no"
                    subTree=".1.3.1.4.15145.10,
                    .1.3.1.4.15145.20"
                    timeout="5000"
                    readCommunity="yourReadCommunity"
                    writeCommunity="yourWriteCommunity"
                    />
```

4.3.8. The <trapProxy> Section

If the master agent also needs to forward traps sent from proxied agents, you can add a trapProxy XML node to the config file. Only one trap proxy can be defined in the config file. The following example defines a trapProxy node that will start a trap receiver listening on port 192. Traps received will be forwarded to all trap sinks defined in the config file.

```
<trapProxy
                trapForwarderPort="192"
                />
```

4.4. The VOD Agent as SNMP Subagent

The Simple Network Management Protocol (SNMP) has been widely used in enterprise networks to effectively manage systems, network devices, and networks. The widespread use of SNMP has raised many issues relating to managing systems and networks. One of the benefits of SNMP is how quickly solutions may be created to support the increasing numbers of networking components and applications.

Within SNMP networks, the number of entities (systems, components, and applications) that need to be managed is growing rapidly. There is a need to respond to the industry's demand for more flexible and dynamic management of multiple devices.

The initial network management solution based on SNMP, allowed developers to create one monolithic agent per system/device listening on a single port (port 161). It was soon discovered that this SNMP solution had many constraints and was not flexible enough to effectively manage all the devices necessary.

New technology was needed to produce multiple agents by different people, that could manage different components and applications separately within a device. This resulted in the new extensible agent technology or master/subagent technology.

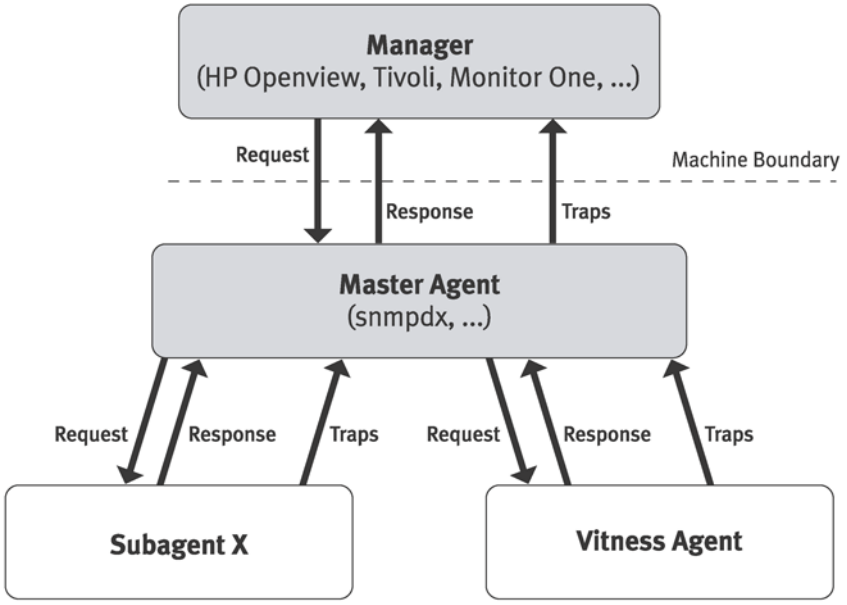


Figure 4.2. Master Agent/Subagent architecture

The master agent receives SNMP requests from the system managers and sends responses to these requests after determining appropriate values from the subagents. The subagents provide management of different components based on the MIB specifically designed for such components. Each subagent registers with the master agent. During registration, it informs the master agent of the MIB subtree it manages.

SNMP agents that have configurable port options, as does the VOD Agent, can be registered as subagents to take advantage of the extensible master agent architecture. That means that the Vitness SNMP Agent can

transparently be integrated in environments where existing master agents are already deployed by configuring it to be run as a subagent.

Configuration of the Vitness Subagent

Though configuration of SNMP subagents will vary on different platforms, the basic principle to use a specific port to allow SNMP requests to be routed by the master agent to the subagent is valid as long as the master agent implements a master/subagent architecture that allows the separate execution of multiple subagents on a system.

In all such cases the VOD Agent can be configured to run as an SNMP subagent by just specifying the number of the port that the master agent uses to communicate with it. This is done in the `snmp-port` attribute of the `<snmp-adaptor>` element in the `agent.xml` configuration file.

The following sections describe the configuration of the VOD Agent as a SNMP subagent for these platforms.

- Windows
- Redhat Enterprise Linux
- Solaris 9
- Solaris 10
- Platforms using net-snmp

For information on how to configure an SNMP subagent on other platforms, not described here, please refer to the documentation provided with the master agent for the platform.

4.4.1. Subagent Configuration on Windows

First ensure that you have the Windows SNMP service installed. To install SNMP on Windows use the Windows Components Wizard (**Add/Remove Windows Components** in the **Add or Remove Programs** dialog) to add the **Managing and Monitoring Tools**.

For Windows, Vitness SNMP subagent support is provided by a special DLL file, `subagent.dll`. You will find this file in the `bin` directory where you installed the VOD Agent (normally `C:\VitnessAgent`). Also provided is a Windows registry file, `subagent.reg`, in the `snmp\integration\win32` directory.

Example 4.3. Vitness Subagent Windows Registry File `subagent.reg`

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\iReasoning]

[HKEY_LOCAL_MACHINE\SOFTWARE\iReasoning\Agent]

[HKEY_LOCAL_MACHINE\SOFTWARE\iReasoning\Agent\CurrentVersion]
"JavaAgentIpAddress"="localhost"
"Enabled"="Yes"
"JavaAgentPort"=dword:00000489
"JavaAgentReadCommunity"="public"
"JavaAgentWriteCommunity"="public"
"Pathname"="C:\\Versant\\VitnessAgent\\bin\\subagent.dll"

[HKEY_LOCAL_MACHINE\SOFTWARE\iReasoning\Agent\CurrentVersion\Subtrees]
"1"=".1.3.6.1.4.1.8884"

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SNMP\Parameters\ExtensionAgents]
"0"="SOFTWARE\\iReasoning\\Agent\\CurrentVersion"
```

If necessary, adjust the path to the `subagent.dll` file in the registry file. Register the settings by executing (double-clicking) the `subagent.reg` file. Restart the Windows SNMP Service from **Windows Services**.

The path to the `subagent.dll` in the registry file may not contain white-space characters.

Subagent Versant post is defined as a DWORD HEX value.

4.4.2. Subagent Configuration on Redhat Enterprise Linux 4 with SNMP Daemon

The Redhat SNMP Daemon (snmpd) that is shipped with Redhat Enterprise Linux 4 supports several masteragent/subagent maechanisms for the addition of a subagent (Proxy, AgentX, etc.). It is based on the net-snmp open-source SNMP agent (information at http://net-snmp.sourceforge.net/). For Vitness, adding a SNMP subagent to the snmpd masteragent is accomplished by adding a `proxy` entry to the snmpd configuration file, `/etc/snmp/snmp.conf`. For the syntax for the `proxy` entry, refer to the man page for **snmpd.conf**.

Here is an example of the `proxy` configuration file entry specifying a Vitness Subagent running on port 1161.

Example 4.4. SNMP Subagent Configuration—Redhat Linux

```
 . . .
# proxy Versant SNMP Agent via its Enterprise OID .1.3.6.1.4.1.8884
proxy -v 2c -c public localhost:1161 .1.3.6.1.4.1.8884
 . . .
```

You must be sure that the appropriate entries in the Vitness `agent-snmp.xml` configuration file match those in the Linux `snmpd.conf` file. Following is a fragment of the `agent-snmp.xml` configuration file showing the correct settings in the `<properties>` section to match the `snmpd.conf` from the example above.

```
 . . .
<properties
version="2"
port="1161"
encryptPasswordAndCommunity="no"
readCommunity="public"
writeCommunity=""
maxPacketSize="4000"
useThreadPool="yes"
engineID=""
 . . .
```

4.4.3. Subagent Configuration on Sun Solaris 9 with Solstice Enterprise Agent

The Sun Solstice Enterprise Agent (`snmpdx`), shipped with the Solaris Operating Environment, implements a master/subagent architecture that allows the separate execution of multiple subagents on a system, with the master agent acting as the subagent scheduler and main interface to the SNMP management application. The Solstice Enterprise Agent, currently supports SNMPv1 and is shipped with a basic MIB-II subagent.

The `snmpdx` master agent runs as a daemon process listening to UDP port 161 for SNMP requests. The Master Agent also opens another port, by default UDP port 162, to receive SNMP trap notifications from various subagents. These traps are forwarded to various managers as determined by the configuration files.

Upon invocation, `snmpdx` reads its various configuration files and takes appropriate actions by activating subagents, determining the subtree OID for various subagents, populating its own MIBs, and so forth. The master agent invokes and registers subagents, sends requests to and receives responses and traps notifications from subagents.

Any communication from subagents to the master agent is done through UDP port 161. The subagents send the traps to the master agent through UDP port 162 and the master agent then decides which managers will receive the trap.

For further details on the Sun Solstice Enterprise Agent, please refer the Sun documentation. A reference is given in Appendix C, *References* [p. 193].

The VOD Agent can be configured to be a subagent of the `snmpdx` master agent so that each request received by the master agent that is relative to the Versant MIB subtree is routed to the VOD Agent.

Sample configuration files are provided in the `/snmp/integration/Solaris9` subdirectory of the Vitness installation. Such files should be copied into the default Solaris `snmpdx` configuration directory, `/etc/snmp/conf`, and edited as necessary.

The files provided by Versant are as follows.

`versant-agent.rsrc`
: The Vitness subagent resource file

`versant-agent.reg`
: The Vitness subagent registration file

`versant-agent-run.sh`
: The Vitness subagent start-up command script

The resource file for the VOD Agent is read by the master agent as soon as this becomes active. This file contains information about the registration file associated with the subagent in addition to the other information related to invoking the subagent.

Example 4.5. Vitness subagent resource file

```
# Versant Resource Configuration file
#
# File Name: versant-agent.rsrc
# Fri Mar 28 18:18:14 GMT 2003

resource =
{
  {
    registration_file = "/etc/snmp/conf/versant-agent.reg"
    policy = "spawn"
    type = "legacy"
    command = "/etc/snmp/conf/versant-agent-run.sh"
  }
}
```

The variables in the resource group are related to the Vitness subagent as described below.

`registration_file`
: This field specifies the registration file for the Vitness subagent. The master agent reads the various entries in this file and creates appropriate entries in its MIB table.

`policy`
: The value`spawn` specifies that the master agent invoke the Vitness subagent as stated in the `command` field of the resource entry.

`type`
: The value `legacy` specifies that the subagent will be statically configured at start-up.

`command`
: This is the name of the subagent executable or start-up script. A default `versant-agent-run.sh` script is provided with Vitness. A command may use the `$PORT` macro to provide the port number from which the subagent receives SNMP requests. That is, it is assigned a value by the master agent in the registration file of each subagent. For the Vitness subagent, the port used to receive SNMP requests is specified in the `snmp-port` attribute of the `<snmp-adaptor>` element in the `agent.xml` configuration file.

The registration file contains information pertinent to the VOD Agent such as the name of the agent, the subtree OIDs from the Versant MIB managed by the subagent, request time out, and preferred port number.

Example 4.6. Vitness subagent registration file

```
# Versant Agent Registration File
#
# File Name: versant-agent.reg
# Fri Mar 28 18:18:14 GMT 2003

macros = {
  regOID = enterprise.8884
}

agents =
{
  {
    name = "versant-agent"
    subtrees = {  regOID }
    timeout = 200000
    watch-dog-time = 60
    port = 1161
  }
}
```

The following describes the variables related to the Vitness subagent used in the `agents` group in the example above.

`name`

Specifies the name of the subagent. The master agent uses the agent name as a key in the agent table MIB.

`subtrees`

Specifies the subtree OIDs that are managed by the Vitness subagent. The Versant Enterprise OID is specified in the `macros` group as `enterprises.8884`.

`timeout`

The timeout variable is registered with the master agent. The master agent waits for the specified number of microseconds to receive a response to its SNMP requests. Each agent specifies its own timeout, though this timeout may not be greater than the `max_agent_time_out` defined in the master agent resource configuration file.

watch_dog_time

The master agent uses this timeout to determine if the subagent is active. The master agent polls the subagent only if there has been no activity between the master agent itself and the subagent for the watch_dog_time interval.

port

Specifies the number of the port that the subagent uses to to receive SNMP requests from the master agent. This port number must be specified in the snmp-port attribute of the <snmp-adaptor > element in the agent.xml configuration file. The port used by VOD Agent to send traps to the master agent is UDP port 162. This is the standard port that the master agent uses to receive traps and is specified by the trap-port attribute of the <snmp-adaptor> element in the agent.xml configuration file.

The Vitness subagent start-up command is a simple script that invokes the VOD Agent start-up script.

```
#!/bin/sh
# This scripts starts the Versant Sub Agent
AGENTPORT=$1
TARGET_DIR=/usr/local/versant/VitnessAgent
cd $TARGET_DIR
./versant-agent-run.sh
```

The recommended access permission of the Vitness subagent configuration files are as follows.

Example 4.7. Vitness subagent configuration files permissions

```
-rwx------   1 root     sys        versant-agent-run.sh
-rw-------   1 root     sys        versant-agent.reg
-rw-------   1 root     sys        versant-agent.rsrc
```

Starting the Vitness SNMP Subagent on Solaris 9

Once the Vitness subagent configuration has been defined in the /etc/snmp/conf directory, go to the /etc/init.d directory and execute the following commands:

# ./init.snmpdx stop	Stop the already running snmpdx master agent
# ./init.snmpdx start	Restart the snmpdx master agent which starts the Vitness subagent

4.4.4. Subagent Configuration on Sun Solaris 10 with SMA SNMP Daemon

The Solaris SMA SNMP Daemon (snmpd) that is shipped with Solaris 10 is based on the net-snmp open-source SNMP agent (information at http://net-snmp.sourceforge.net/). To add the Vitness SNMP Agent subagent to the snmpd master agent a `proxy` entry is required in the `/etc/sma/snmp/snmpd.conf` configuration file. For the syntax for the `proxy` entry, refer to the man page for `snmpd.conf`.

Here is an example of the `proxy` configuration file entry specifying a Vitness Subagent running on port 1161.

Example 4.8. SNMP Subagent Configuration—Solaris 10

```
 . . .
# proxy Versant SNMP Agent via its Enterprise OID .1.3.6.1.4.1.8884
proxy -v 2c -c public localhost:1161 .1.3.6.1.4.1.8884
 . . .
```

You must be sure that the appropriate entries in the Vitness `agent-snmp.xml` configuration file match those in the Linux `snmpd.conf` file. Following is a fragment of the `agent-snmp.xml` configuration file showing the correct settings in the `<properties>` section to match the `snmpd.conf` from the example above.

```
 . . .
<properties
version="2"
port="1161"
encryptPasswordAndCommunity="no"
readCommunity="public"
writeCommunity=""
maxPacketSize="4000"
useThreadPool="yes"
engineID=""
 . . .
```

4.4.5. Subagent Configuration for net-snmp Agents

The net-snmp SNMP agent is an open-source project (information at http://net-snmp.sourceforge.net/) and is the basis for SNMP support on a number of platforms (refer, e.g., to Section 4.4.2, “Subagent Configuration

on Redhat Enterprise Linux 4 with SNMP Daemon" [p. 94] and Section 4.4.4, "Subagent Configuration on Sun Solaris 10 with SMA SNMP Daemon" [p. 100], above).

If you are using net-snmp as the SNMP master agent (for example, under AIX), the configuration for Vitness SNMP Subagent will be nearly identical to the configurations described for Redhat Linux or Solaris 10, above. Refer to either of those sections for information.

4.5. The Versant Management Information Base (MIB)

A MIB is based on a hierarchical relationship between managed objects. This hierarchical relationship is a tree structure, called the MIB tree or Object Identifier tree. Each managed object in the MIB is assigned a unique number called an *object identifier* or *OID* . An OID consists of a left-to-right sequence of integers. This sequence defines the location of the object in the MIB tree. By specifying a unique path through the tree to the object, the OID allows the object to be identified uniquely. Each node in the path defined in an OID has both a number and a name associated with it. The path `.1.3.6.1.4.1`, e.g., defines the `private.enterprises` OID and each number beneath that node on the tree represents the branches in the tree reserved for a particular vendor.

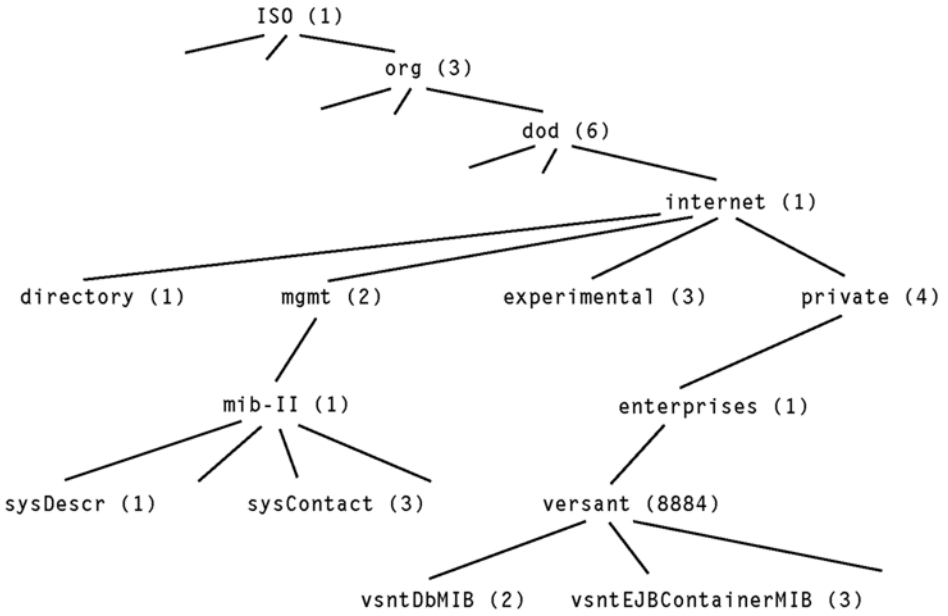


Figure 4.3. Object Identifier tree

The Versant MIB is registered at the location `.1.3.6.1.4.1.8884` in the tree. And the Versant Object Database Server MIB consists of all OIDs below `.1.3.6.1.4.1.8884.2`.

All objects contained under the `mgmt` branch (i.e., all objects with OIDs beginning with `.1.3.6.1.2`) are considered standard and are tightly regulated by the IETF (Internet Engineering Task Force, http://www.ietf.org/). For example, the standard MIB-II MIB lives under the `mgmt` branch and is supported by all SNMP compliant agents.

Extending the management reach of an SNMP manager to encompass a Versant domain requires a MIB that defines the manageable Versant Object Database managed objects for the SNMP management system. The Versant MIB is defined in an SNMP-compliant file written in Abstract Syntax Notation One (ASN.1). This file is `versant-mib.txt` and is packaged with Vitness under the `snmp/mib` directory.

The Vitness SNMP Agent implements the Versant MIB as well as a subset of the MIB-II, as defined in the RFC-1213, limited to the system and `snmp` groups. In particular the `snmp` group objects can give an idea of how much the SNMP agent is busy.

Please refer to the RFC-1213 document for the MIB-II (refer to Appendix C, *References* [p. 193]).

The Versant MIB and the MIB-II subset files are packaged with Vitness in the *Vitness*/snmp/mib subdirectory of the Vitness installation.

4.5.1. Managed Elements

The Versant MIB is designed following the Versant Object Database server concept, where on a single host multiple, independent databases can be deployed. Following this principle, the MIB maintains a group of environment information common to all the databases deployed on the host (`versant.vsntEnvInfo`), and a group of database specific objects represented in tables of monitored databases whose managed objects are made accessible via the SNMP protocol (`versant.vsntDbMIB.vsntDbObjects`).

The managed objects in the `vsntDbObjects` group are grouped into the three main tables.

`vsntDbTable`
Holds objects related to the databases.

`vsntLogVolTable`
Holds the log volumes.

`vsntDataVolTable`
Holds the data volumes of each database.

4.5.1.1. Managed Elements in the `vsntEnvInfo` group

The objects in this group are common to each database deployed on the host and represent the Versant Object Database environment.

Table 4.1. Managed Objects in the vsntEnvInfo group

Object Name	Description
`vsntPrivateMibOID`	The authoritative identification for the private MIB for this database, based on the vendor, e.g., {enterprises `8884` *`optional subidentifiers`*} for Versant Object Database databases. If no OBJECT IDENTIFIER exists for the private MIB, attempts to access this object will return `noSuchName` (SNMPv1) or `noSuchInstance` (SNMPv2).
`vsntVendorName`	The name of the vendor whose ODBMS manages this database, i.e., Versant Corporation, for informational purposes.
`vsntEnvDbIDNode`	The host name of the machine containing the Versant Object Database `osc-dbid` database system file.
`vsntEnvDbIDPath`	The local directory containing the Versant Object Database `osc-dbid` database system file.
`vsntEnvDbPath`	The database root directory.
`vsntEnvRootPath`	The Versant Object Database software root directory.
`vsntEnvRuntimePath`	The Versant Object Database software runtime path.

4.5.1.2. Managed Elements in the `vsntDbObjects` group

The objects in this group are related to each database deployed on the host, and are grouped into three tables.

Table 4.2. Managed Objects in the vsntDbObjects group

Object Name	Description
`vsntDbTable`	The table of databases in a system (host or server).
`vsntLogVolTable`	The list of log volumes of a monitored database. The table will have only two rows for each database, one for the logical log volume and one for the physical log volume.
`vsntDataVolTable`	The list of all the mounted volumes of each monitored database. Each row in the table contains, for each monitored database, volume related information and a triggering condition for a specific trap that is sent if the space available in a volume falls below the configured value.

The `vsntDbTable` table contains information on monitored databases, and is indexed on the `vsntDbID` object.

Table 4.3. Managed Objects in the `vsntDbTable` table

Object Name	Description
`vsntDbID`	The ID of the monitored database.
`vsntDbName`	The name of this managed database.
`vsntDbRelease`	The release of this managed database.
`vsntDbContact`	The textual identification of the contact person for this managed database, together with information on how to contact this person. Note that the agent may need to keep this in other persistent storage, e.g., a configuration file. Note that a compliant agent does not need to allow write access to this object. (Refer to Section 3.1.5, “Alarm Notification Parameters” [p. 66].)
`vsntDbType`	The type (personal, group) of the database.
`vsntDbOwner`	The owner (dba) of the database.
`vsntDbCreateTime`	The date and time of creation of the database.
`vsntDbPath`	The path of the database.
`vsntDbSize`	The total size of the monitored database in bytes: Mb, Gb, etc.
`vsntDbFreeSpace`	The total amount of free space in the monitored database in bytes: Mb, Gb, etc.
`vsntDbFullThreshold`	The threshold beyond which a *db full* trap is to be triggered for the monitored database.
`vsntDbMode`	The state of this database access mode. The meaning of the values are: `dba-single` (1), the database is in DBA-only single-connection mode; `dba-multi` (2), the database is in DBA-only multi-connection mode; `multiuser` (3), the database is in multi-user mode; `unstartable` (4), the database is in unstartable mode

The `vsntLogVolTable` table contains information on the managed databases log volumes, and is indexed on the `vsntDbID` (from the `vsntDbTable` table) and `vsntLogID` objects:

Table 4.4. Managed Objects in the `vsntLogVolTable` table

Object Name	Description
`vsntLogID`	The conventional ID of the monitored log volume: `llog` (0), the logical log volume; `plog` (1), the physical log volume
`vsntLogName`	The name of the monitored log volume
`vsntLogPath`	The complete path of the monitored log volume
`vsntLogInitialSize`	The initial size in kilobytes (1024) of the monitored log volume
`vsntLogActualSize`	The actual size in kilobytes (1024) of the monitored log volume
`vsntLogFreeKBytes`	The number of available kilobytes (1024) in the monitored log volume
`vsntLogIsOnRawDev`	It is `true` if the monitored log volume is on raw device, `false` otherwise

The `vsntDataVolTable` table contains information on the managed databases data volumes, and is indexed on the `vsntDbID` (from Table 4.3, "Managed Objects in the `vsntDbTable` table" [p. 105]) and `vsntVolID` objects.

Table 4.5. Managed Objects in the `vsntDataVolTable` table

Object Name	Description
`vsntVolID`	The ID of the monitored data volume
`vsntVolName`	The name of the monitored data volume
`vsntVolPath`	The complete path of the monitored data volume
`vsntVolSize`	The size in kilobytes of the monitored data volume
`vsntVolExtSize`	The number of pages in an extent of the monitored data volume
`vsntVolNumExt`	The total number of extents in the monitored data volume
`vsntVolFreeExt`	The number of available extents in the monitored data volume
`vsntVolFreeKBytes`	The number of available kilobytes in the monitored data volume
`vsntVolFullThreshold`	The threshold beyond which a Volume Full trap is to be triggered for the monitored data volume

4.5.2. Traps

The VOD Agent uses traps to notify management applications of important resource and agent events and conditions. Version 1 of the SNMP standard includes six generic traps and one enterprise-specific trap, as defined by the trap type field of the trap PDU.

Generic Trap Type	Meaning	Event
Cold Start (0)	The agent has been started.	Startup sequence is complete and the agent is ready to process incoming requests.
Warm Start (1)	The agent has been reconfigured.	Anytime resources or agent configuration changes.
Link Down (2)	The agent has stopped.	Whenever the agent, or its resource, becomes unstable or unavailable.
Link Up (3)	A network interface link has come up.	N/A
Authentication Failure (4)	The agent received a request with an unknown community string.	Generated on behalf of the agent by the SNMP Administrator.
EGP Neighbour Loss (5)	An external gateway protocol (EGP) peer neighbour is down. This trap is relevant only to instrumenting a device on an EGP network.	N/A
Enterprise Specific (6)	Context dependent. A specific trap subtype further clarifies the meaning.	Whenever managers need to be notified of a resource-specific event.

Several types of enterprise specific traps are defined to represent resource events of interest to management application. Such specific trap types include the following.

- State change traps to report on changes in the states of Versant Object Database resource's managed objects
- Threshold traps to alert the manager when resource performance crosses a specified threshold
- Alarm traps to report exceptional conditions or events

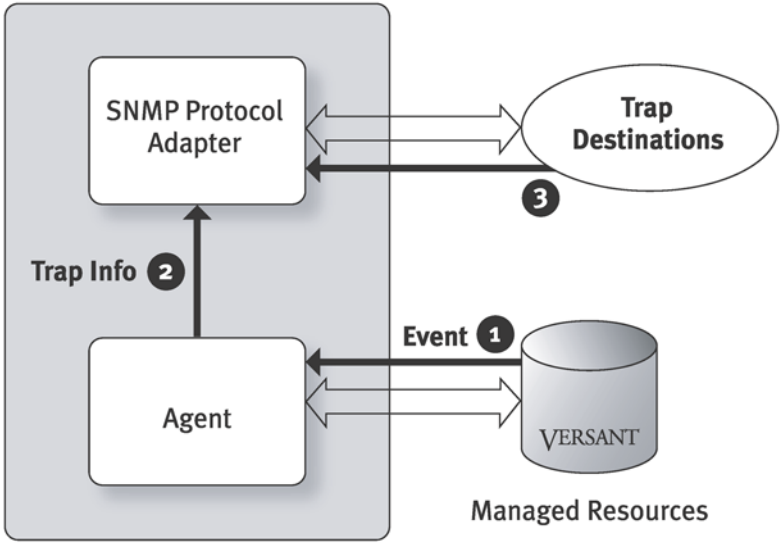


Figure 4.4. Trap data flow

The Startup data flow includes the generation of a cold start trap.

1. A trap-worthy event occurs.
2. The agent prepares a trap message containing the following information.
   a. The generic trap type (6 for enterprise specific)
   b. The specific trap type
   c. A variable bindings list that includes any attribute/value pairs that provide details about the managed object that is the subject of the trap message
3. The SNMP Protocol Adaptor encodes the trap PDU and sends one to each of the configured trap destinations.

Versant Traps

The Versant MIB defines a number of enterprise specific traps. Such traps carry information related to the event that triggered them using the following trap variables.

Table 4.6. Versant trap variables

Variable name	Description
`trapTime`	The time the event that triggered the trap occurred
`trapLogSeverity`	The severity of the trap's message
`trapDesc`	The description of the event
`trapMonitorType`	A string that should be either `"CounterMonitor"`, `"GaugeMonitor"`, or `"StringMonitor"`
`trapMonitorThreshold`	An ASCII string representation of the value which triggered this monitor
`trapMonitorValue`	An ASCII string representation, which is the derived value

The traps currently defined and implemented by the Vitness SNMP Agent are described in the following table.

Table 4.7. Versant enterprise specific traps

Trap name	Variables	Description
`vsntLogExpanded`	`trapTime`, `vsntDbName`, `trapDesc`	The specified Logical Log Volume has been expanded.
`vsntVolFull`	`trapTime`, `vsntDbName`, `vsntVolID`, `trapDesc`	The specified Data Volume has reached the *volume full* threshold.
`vsntVolAdded`	`trapTime`, `vsntDbName`, `vsntVolID`, `vsntVolName`, `trapDesc`	The specified Data Volume has been added to the database.
`vsntDbFull`	`trapTime`, `vsntDbName`, `trapDesc`	The *db full* threshold has been reached.
`vsntDbStartUp`	`trapTime`, `trapDbName`, `trapDesc`	The database has been started.
`vsntDbShutDown`	`trapTime`, `trapDbName`, `trapDesc`	The database has been shut down.
`vsntModeChange`	`trapTime`, `trapDbName`, `vsntDbMode`, `trapDesc`	The database mode has changed. The mode sent with the trap is the new state.
`vsntLoggedEvent`	`trapTime`, `trapDbName`, `trapLogSeverity`, `trapDesc`	The trap is generated after an event is logged in the database LOGFILE.

The `vsntLogExpanded` and `vsntDbFull` traps are sent repeatedly after a configurable interval of time until the triggering condition is reset, i.e., the log volume is compacted or more space is made available in the database, respectively).

Chapter 5. Vitness Database Administration

The database display in the Service Browser View provides a visual indication of the database status and mode. And you can change the status and mode from within Vitness.

5.1. Database Status and Modes

The database status is either active (started) or not active (stopped). The database mode in Versant Object Database determines the type of connection and access level allowed. Vitness displays the database status and mode for the databases in the Service Browser View and Statusline.

Following is a quick overview of the database modes.

Database Modes

DBA-only single-connection

The database will be set to a DBA-only single-connection mode where only the DBA can establish a connection. There may only be a single connection to the database at a time.

Multi-user

The database will be set to the Multi-user mode. This is the default database mode. It allows multiple users.

DBA-only multi-connection

The database will be set to the DBA-only mutiple connections mode. This allows only the DBA to establish a connection to the database but the DBA may establish multiple concurrent connections.

Unstartable

The Database will be set to the unstartable mode. This mode prevents the Database from being started.

The database can be set to the unstartable mode only if it is not active (i.e., not started or stopped).

Read-only

The database will be set to the read-only mode. In this mode the database can only be accessed with read-only transactions. Use this mode, for example, when the database will be used on a read-only medium.

The database can be set to the read-only mode only if it is not active (i.e., not started or stopped).

Recover

Recover mode is typically shown for an FTS database. It indicates one of the states of resynchronization i.e., this database is being recovered from a failure and is being resynchronized with its replica database pair.

Restore

Restore mode indicates that the database is currently being restored from a backup.

Restore-Suspended

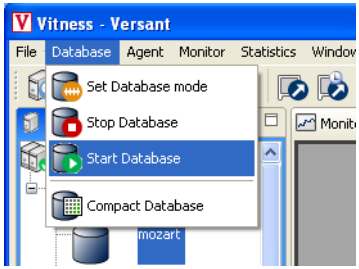
Restore-Suspended mode indicates that the database is a warm standby database and that it is in restore suspended mode i.e., it is in a partially restored mode.

Figure 5.1. Database Status and Mode Display

5.1.1. Setting Database Status—Starting and Stopping a Database

Start Database

To start a database, select it from the Service Browser View and click  in the toolbar or select **Start Database** from the **Database** main menu.



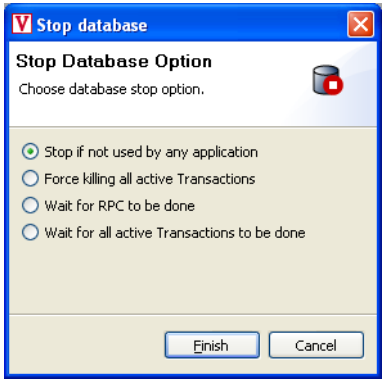
You can also right-click the database and select **Start Database** from the context menu.

If the database has administrator (DBA) authorization enabled, you will be prompted for the administrator password.

Stop Database

To stop a database, select the database and click the  button in the tool bar or select **Stop Database** in the **Database** main menu or context menu for the database. As for Start Database you will be prompted for the administrator password if required.

Next, you will be prompted for the database stop option. The following dialog is displayed.



Select the option you wish from the list and click **Finish** to begin the shutdown. Depending on the option you choose, the shutdown may take some time or may not be possible. The stop database options are described in the following list.

Stop Database Options

Stop if not used by another application

The database is stopped only if there are no other applications with an open session to the database.

Force killing all active Transactions

The database is stopped and any running transactions are aborted.

Wait for RPC to be done

The stop database operation waits until all currently running remote procedure calls (RPCs) are completed.

Wait for all active Transactions

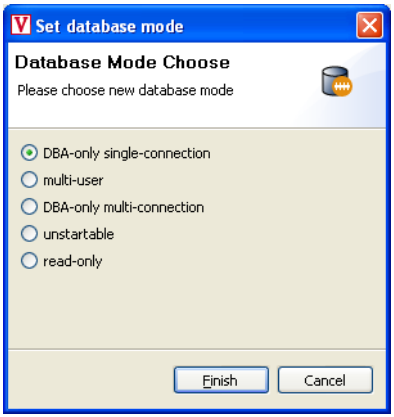
The stop database operation waits until all active transactions are completed.

5.1.2. Setting Database Modes

You can set the mode of a database from Vitness. To set the mode, select the database in the Service Browser View. Click the  button in the toolbar or select **Set Database mode** from the **Database** menu.

You can also right-click on the database and select **Set Database mode** from the context menu.

The following dialog is displayed.



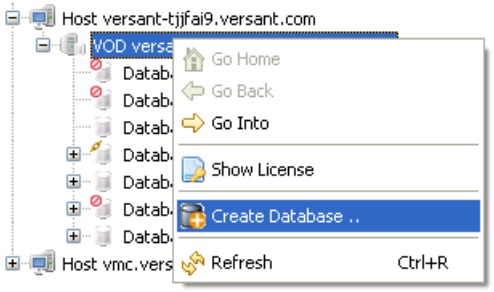
The database modes are summarized in Database Modes [p. 111], above. Select the desired mode and click **Finish**. If the database has DBA autorization set, you will be prompted for the administrator password.

5.2. Database Creation and Management

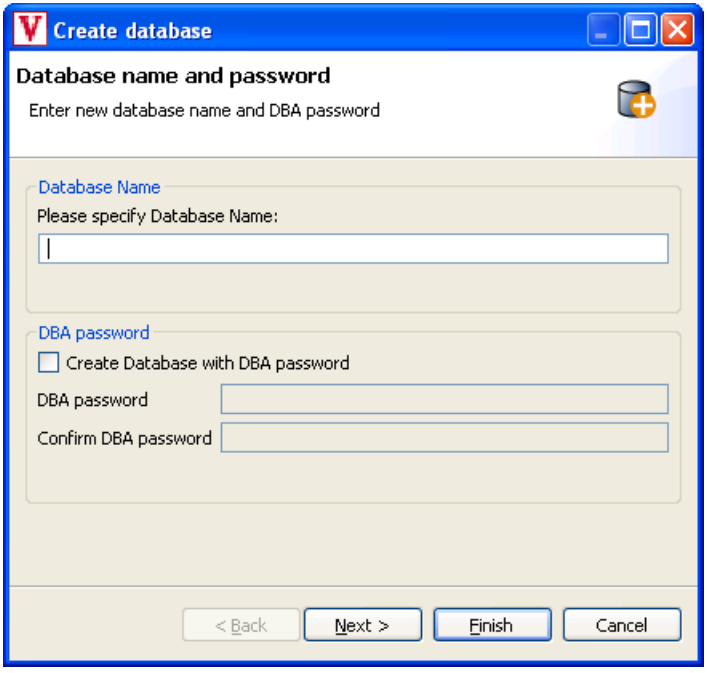
The procedures for creating, removing and copying a database are described in the following sections. The additional options of adding volume and authorizing the database are also introduced.

5.2.1. Create Database

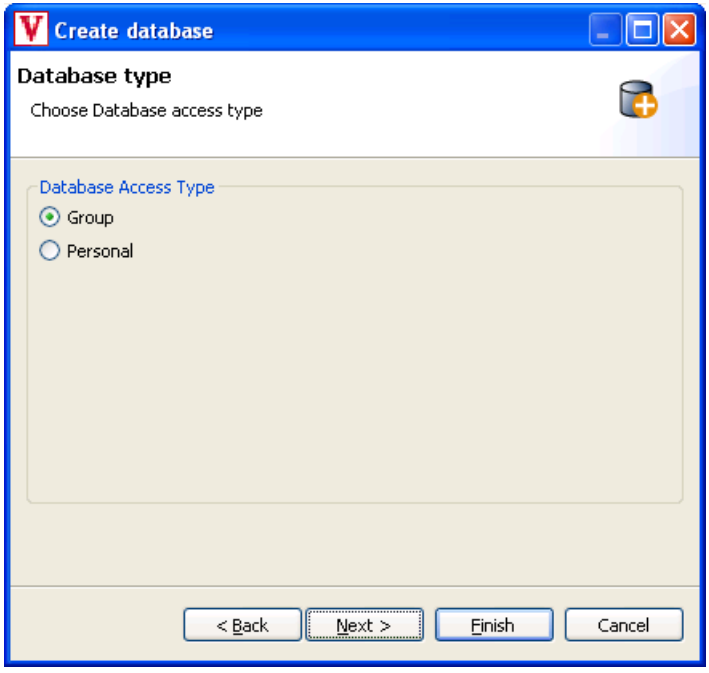
This command is available in the context menu, and the Database menu, when a server node is selected in the tree.



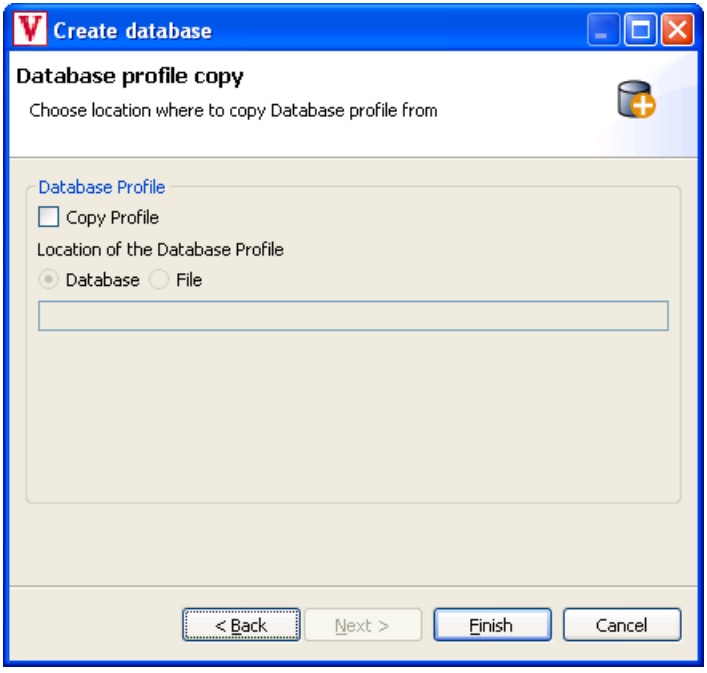
The **Create Database** command opens a wizard dialog that provides several options.



In the first dialog the Database name is to be specified and you can optionally also choose to protect the database with a DBA password.



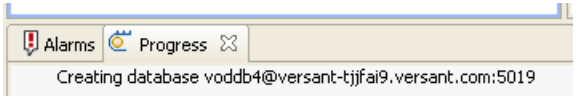
The second dialog asks if the new database should be a group database - the default or, a personal database.



The third dialog allows you to specify the location for the database profile file(`profile.be`) that is used in creating the new database. The location can be another database on another server, or it can also be a file path to a location accessible from within the server environment.

The path syntax is server dependent but, a syntax such as `/versant/dbtemplate/profile.be` is always valid. Make sure the `profile.be` is appropriate for the new database. For example, the volume path names in that `profile.be` file are relative.

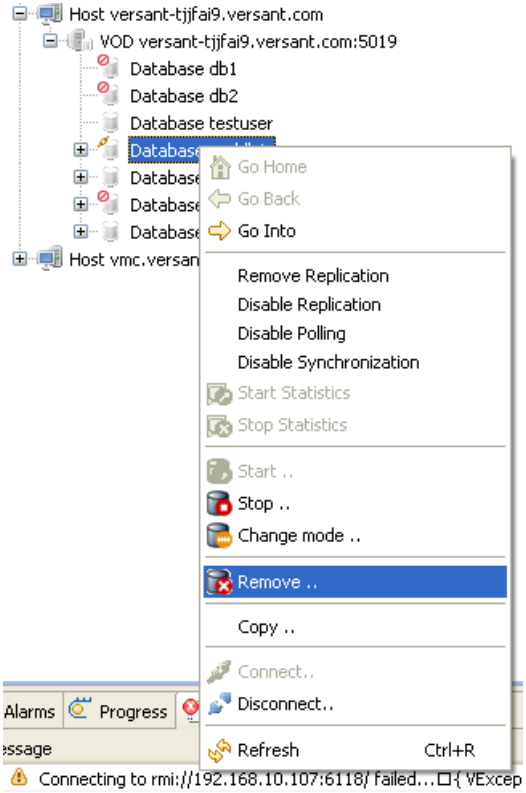
Depending on your system specifications, the database creation process can take a couple of minutes. If the database creation is successfully completed, then the database tree for that VOD server node gets updated. In the mean time the creation process status is displayed in the Progress pane.



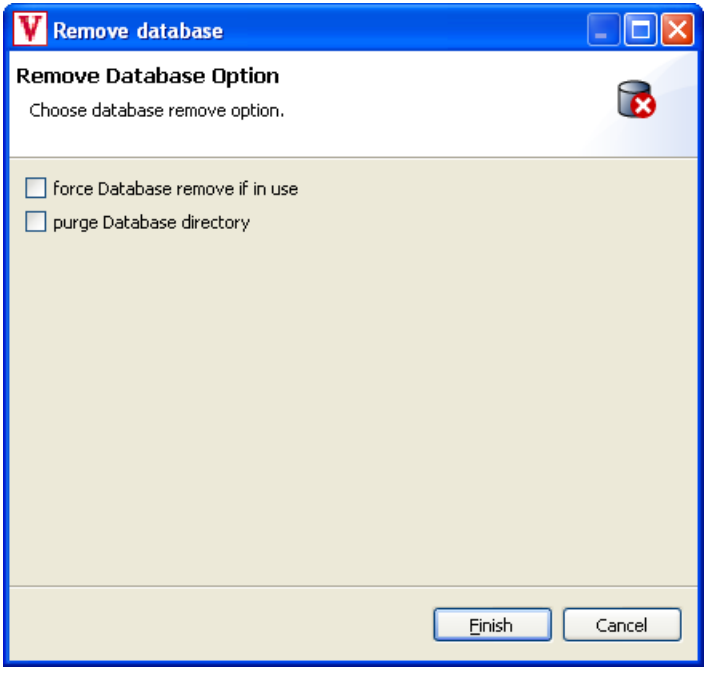
For more information, please refer to the description of the command line tools makedb, createdb in the *Versant Database Administration Manual*.

5.2.2. Remove Database

The **Remove** Database command is available in the menu or the context menu, when a database is selected.



The following dialog is displayed. There are 2 options available.



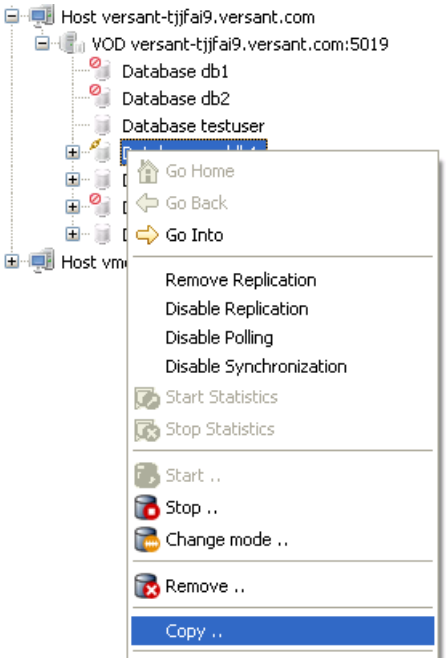
The database is only removed if it is no longer being used. Note that this includes Vitness itself. If Vitness is connected to the database the removal will fail. However, if the option **force Database remove if in use** is checked the database will be removed even if it is in use. This corresponds to the **-f** option of the command line tool removedb.

If the option **purge database directory** is checked, the database directory is removed. This corresponds to the **-rmdir** option of the command line tool removedb.

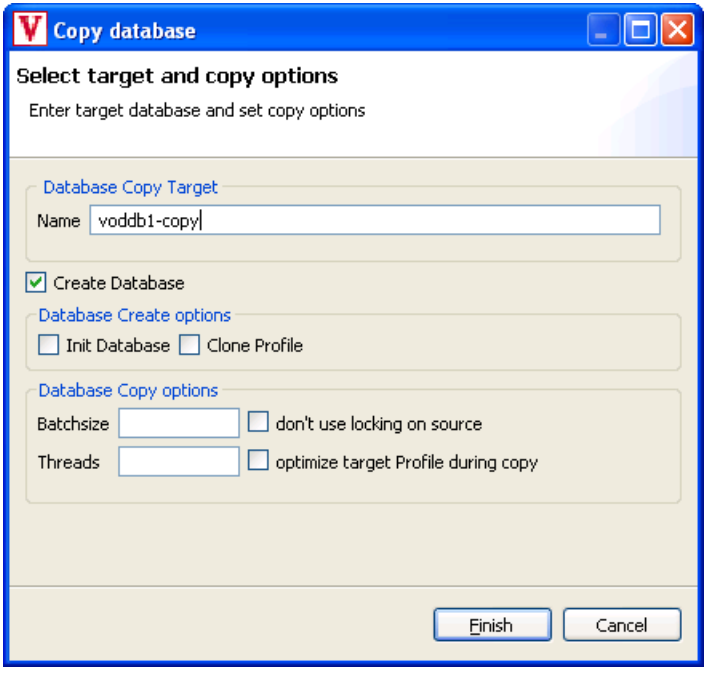
For more information, please refer to the description of the command line tool removedb in the *Versant Database Administration Manual*

5.2.3. Copy Database

The **Copy** Database command is available in the menu or the context menu, when a database is selected.



The dialog for this command provides the following options:



Create Database As default, the target database is created. You can uncheck this option inorder to use an existing database. For example, if you want to use a `profile.be` with the target that is different from the source, and different from the default `profile.be`.

There are two Database Create options, **Init Database** Initialize and grow the database volumes to their final size. **Clone Profile** Clone the `profile.be` from the source database, i.e., copy it from the source database. Make sure, that the volume paths are relative!

Batchsize Define the number of objects copied in a single transaction. The default is 1000 objects per transaction.

Threads Use the given number of threads for the copy operation. The default is 1 thread.

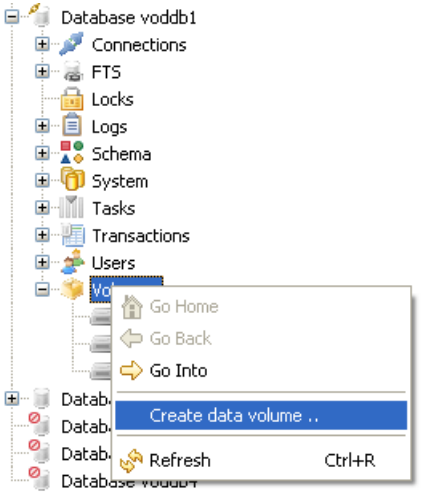
Don't use locking on source Objects are not read-locked.

Optimize target profile during copy Locking and logging are switched off in `profile.be` during the copy operation. This makes the copy operation much faster.

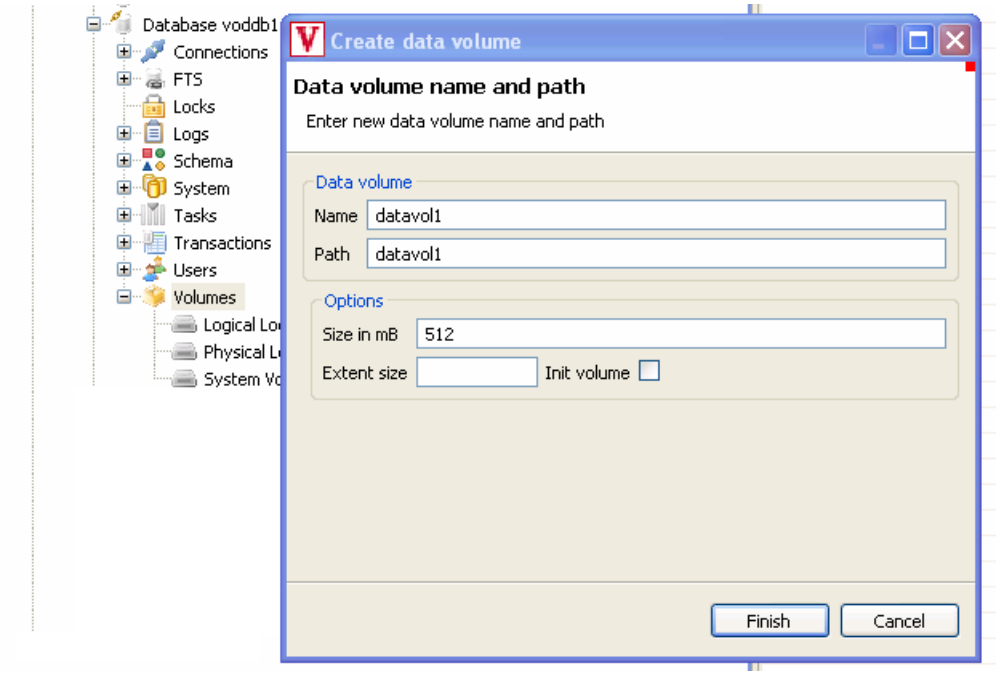
For a more detailed description of these parameters, please refer to the description of the command line tool vcopydb in the *Versant Database Administration Manual*.

5.2.4. Add Volume

To Add Volume to a database, you can select the **Create data volume** command which is available in the Volume tree node of a database.



The following dialog is displayed.



The dialog requests a name and a path for the volume. The size can be specified in Mega Byte, with the default being128 Mega Byte. The extent size can also be specified, while the default value is 2.

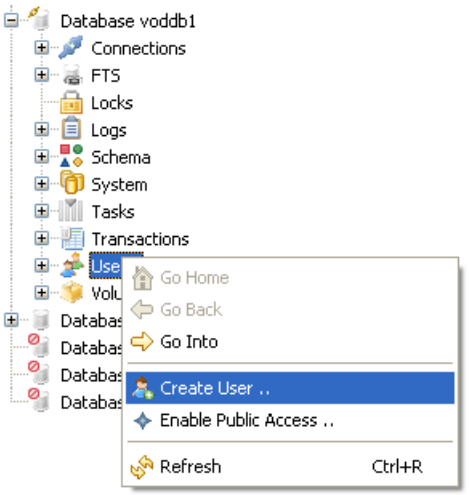
Volume once added, can not be removed from the database.

For a more detailed description of these parameters, please refer to the description of the command line tool addvol in the *Versant Database Administration Manual*.

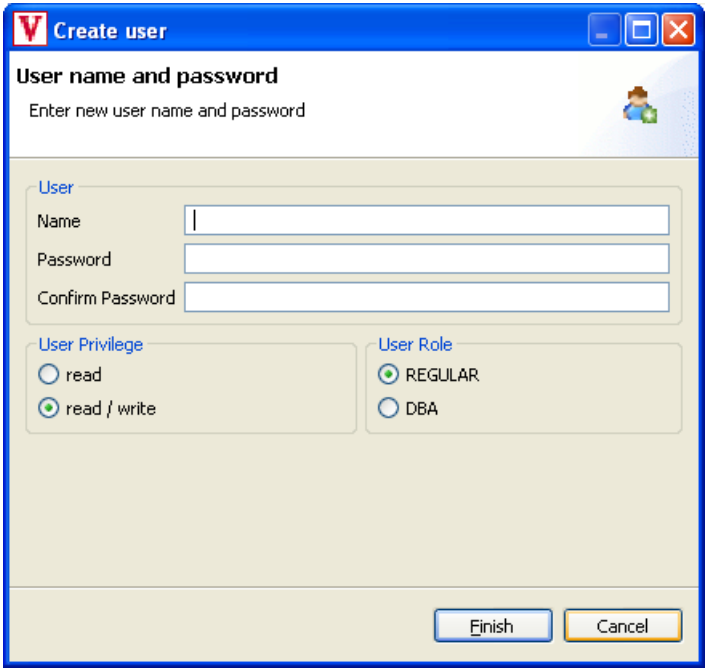
5.2.5. User Authorization

Add User

To add a user, you can select the **Create User** command from the Users tree node of a database.



The following dialog is displayed.



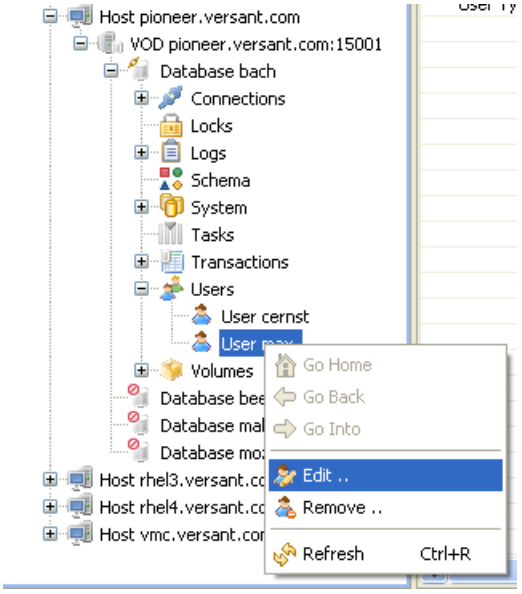
The dialog requests a name and a password for the user. Access can be restricted to read-only access or, the user can take over the role as an DBA.

With Vitness 1.4.0 an additional DBA has restricted rights compared to the owner DBA of a database. But this restriction will go away with one of the next releases. Currently the additional DBA has to have the same name as the client account from which the administration is done. This restriction will go away as well.

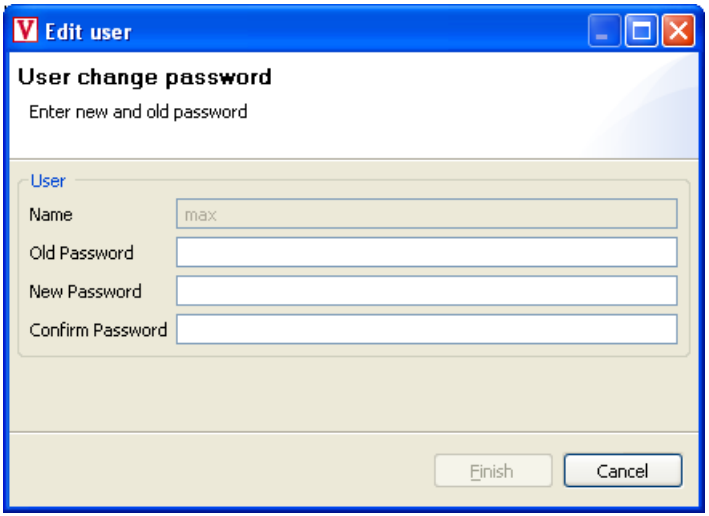
For more information, please refer to the description of the command line tool dbuser in the *Versant Database Administration Manual*

Edit User

To edit a user, you can select **Edit** command from the Users tree node of a database.



The following dialog is displayed.

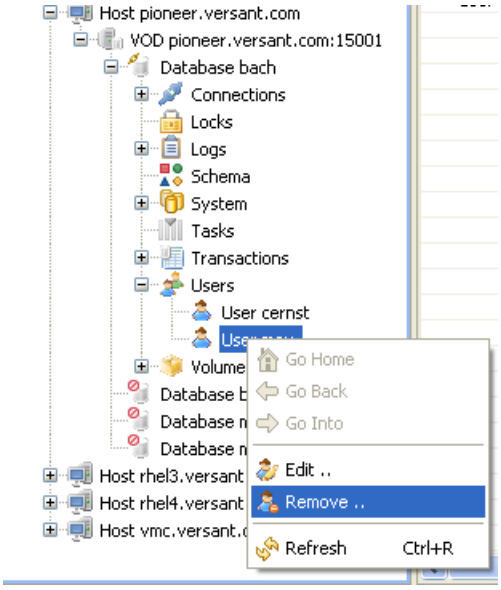


The dialog requests you to enter your old as well as new password. After confirming the new password and clicking on **Finish**, your changes will be finalized.

Currently, the only option for editing the user is to change the password.

Remove User

To remove a user, you can select **Remove** command from the Users tree node of a database.



Before the user is removed, you will be prompted for confirmation.

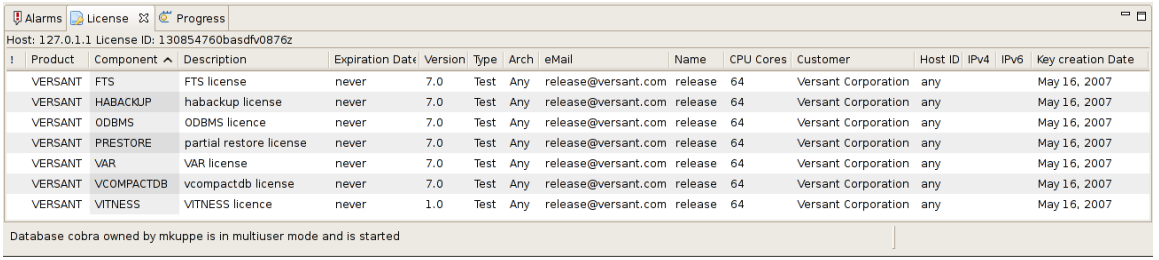
Enable/Disable Public Access

This is another menu item available in the Users context menu. Grant database access to everyone.

For more information, please refer to the description of the command line tool dbuser in the *Versant Database Administration Manual*.

5.3. Vitness License Viewer

You can view the licenses for a Versant Object Database installation from Vitness. To view the licenses, select the server installation you want to view from the Service Browser View. Now, select the **show license** from the pop-up menu. The license list is displayed as shown in the following screen image.



The information displayed is summarized in the following list.

Host

The server host machine.

License ID

The license ID is the Versant identity for your license. Pressing ctrl-**C** will copy the license ID to the clipboard. This is helpful if you are requested by support to provide the ID.

! (expiration warning)

An icon ( ) is displayed in this column if the license is about expire. You can choose whether warnings are displayed and the warning time period from the Vitness Preferences (described in Section 5.3, “License Viewer Preferences” [p. 130], below).

Product

This is the name of the vendor for the licensed product. In most cases this will be Versant.

Component

This is name of the product or product component.

Description

The product or product component in more detail.

Expiration Date

When the product or product component license expires.

Version

The version level of the product or product component.

Type

The license type, e.g., Production, Evaluation, Test, etc.

Arch(itecture)
This shows whether the license is for 32-bit, 64-bit, or any machine architecture.

eMail
The e-mail address of the licensee or contact person.

Name
The name of the licensee or contact person.

CPU Cores
The number of CPU cores allowed by the license for the server machine or “any” if there is no restriction.

Customer
The customer name. This is normally the name of a company.

HostID
The host name for a license that is restricted to a host machine. (Refer to note, below.)

IPv4
The IPv4 address for a license that is restricted to a specific address. (Refer to note, below.)

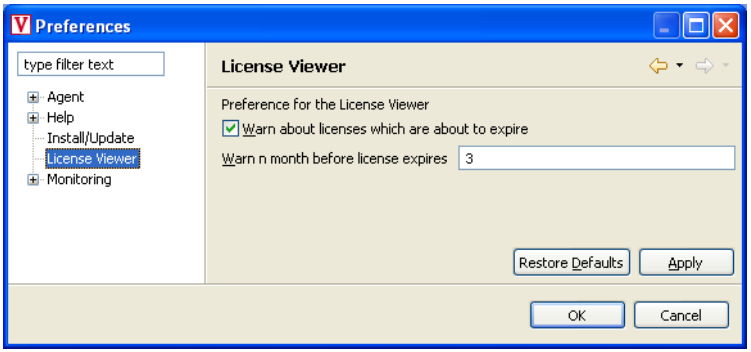
IPv6
The IPv6 address for a license that is restricted to a specific address. (Refer to note, below.)

Key creation Date
This is the date that the license file was created. This is the issue date of the license.

If a license is restricted to a specific host, only one of the HostID, IPv4, or IPv6 fields is specified depending on whether host is identified by ID (name) or IP address.

License Viewer Preferences

To set the License Viewer preferences select **Preferences...** from the **Windows** main menu and then choose **License** from the preferences tree. The following dialog is displayed.



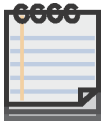
Here you can select whether expiration warnings are displayed and the period of time (in months) before the expiration date when a warning ( ) is displayed.

Chapter 6. The Vorkout Compact Database Tool

In a freshly populated database the objects are efficiently packed on disk, in storage units called pages. A compact database enjoys performance advantages caused by better allocation of data on disk. A compact database also utilizes its backend cache to the fullest extent because each page in cache contains the maximum possible number of objects.

Over time as objects grow or are deleted, empty holes are created in the tightly packed data-base resulting in fragmentation of data segments. As a result, performance starts to suffer and disk usage is also increased. The Vorkout tool is an online database tool that provides you the ability to compact or reorganize the data to reduce fragmentation and restore performance.

In addition, fragmetation of the database *system table* can lead to diminished performance. The system table is used to lookup the physical object locations in the database and can also become fragmented. Vorkout performs de-fragmentation of the system table to optimize your Versant database performance.



Vorkout is an add-on component for the Vitness framework. As a separate component it requires that the license file supplied when you purchased Vorkout be correctly installed with your Versant Object Database installation.

Vorkout uses the same software engine as the command line tool vcompactdb and is also licensed under that name.

6.1. Analyze Database Using Vorkout

To help you to determine if a database needs a reorganization, Vorkout can perform a detailed analysis of the database and report the freespace and degree of fragmentation. The analysis component of Vorkout can determine the degree of fragmentation of data segments by inspecting every page in the database. To start the analysis select the database in the Service Browser View and click the  button in the toolbar or select **Analyze Database** from the **Database** menu. Since the operation can take some time for large databases you will be asked to confirm the Analyze Database operation. The progress is displayed in the standard progress bar. You can also cancel the analysis operation from the progress display.

The data gathered by the analysis operation is used by the compact component of Vorkout as shown in the next section, Section 6.2, “Compact Database Using Vorkout” [p. 134].

6.2. Compact Database Using Vorkout

The Vorkout compact database tool is started from the Vitness console. Select the database from the Network Tree Pane and click the  button in the toolbar or select **Compact Database** from the **Database** menu.

The following dialog is displayed.

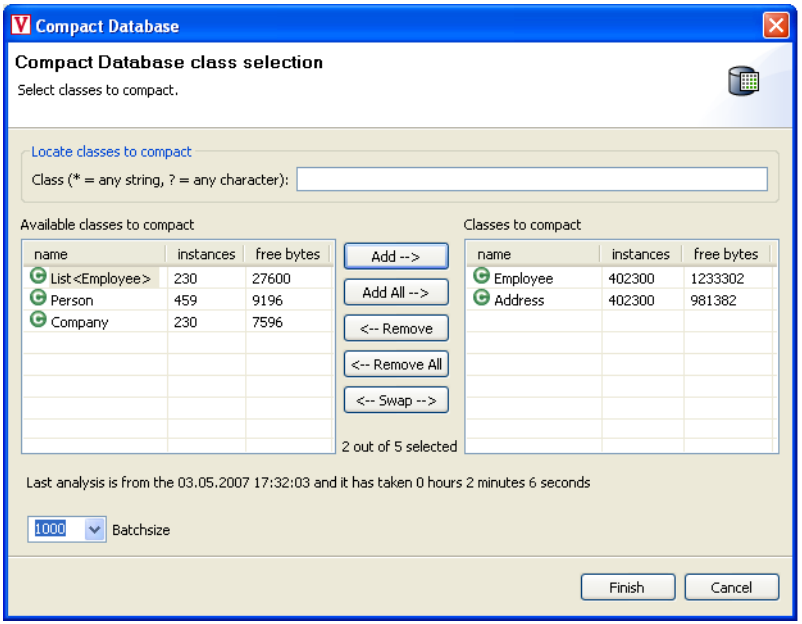


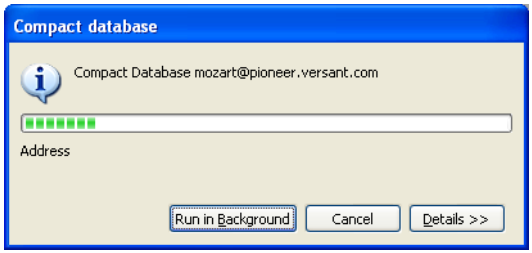
Figure 6.1. Compact Database Class Selection Dialog

This dialog allows you to select the class(es) you want to compact in the database. Only the objects of the selected class(es) will be compacted. If you have run an analysis of the database (refer to Section 6.1, “Analyze Database Using Vorkout” [p. 133]) then the number of *free bytes* (i.e., the bytes lost to fragmentation) is displayed in the **free bytes** column. (The date and time of the last analysis is displayed near the bottom of the dialog. If no analysis data is available the values in the **free bytes** column are replaced by “n/a” for not available.) If there are many instances of a class and the free bytes value is very large the class is a good candidate for compaction.

Select one or more classes from the left side (Available classes to compact) and press the **Add-->** button to move them to the right side (Classes to compact). To move all classes, click the **Add All-->** button. Alternatively,

you can select the classes you want based on their names. Type the appropriate name search string in the **Locate classes to compact** search field. This is convenient if you have many classes in your database.

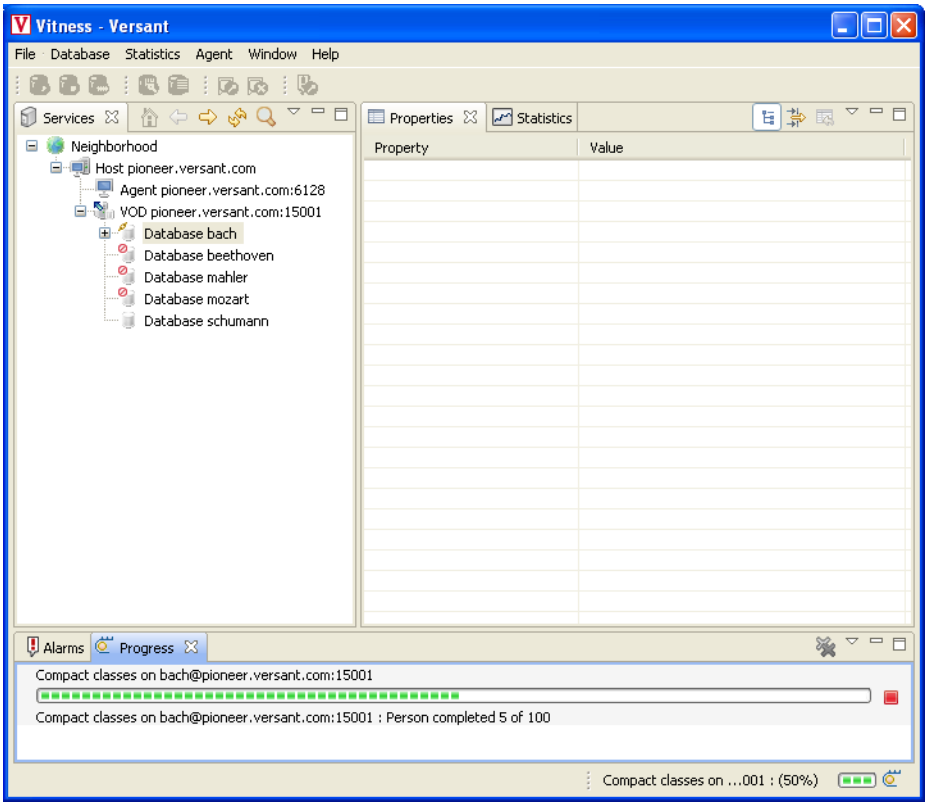
Press the **Finish** button to begin the compact database operation. The following dialog is displayed.



The database and class currently being compacted is displayed as well as a progress bar showing the progress for the entire operation.

Pressing **Cancel** will stop the compact operation following the completion of the current batch.

You can close the dialog and run the operation in the background by clicking the **Run in Background** button. The progress will then be displayed in the Vitness **Progress** pane.



Shutting down Vitness

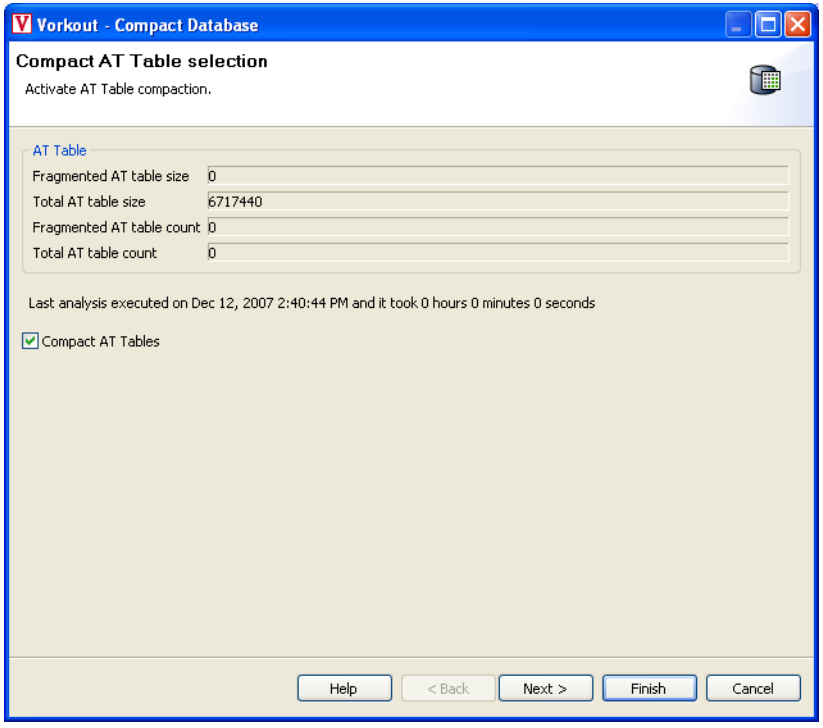
If you quit Vitness while a compact operation is running, the operation will continue in the database server process. If the compact operation is still in progress when you restart Vitness the progress will, after a short delay, again be displayed in the **Progress** pane.

6.3. System Table De-fragmentation

The system table (AT or Association Table) is a hash table that is used by the database server to lookup physical object locations within the database volumes given an objects LOID (logical object identifier) as the key. It is implemented using extensible hashing which is a form of dynamic hashing scheme. The LOID values are the keys for the hash table. Due to the random nature of insertions and deletions into this hash table, the table gets fragmented over time and this leads to a large hash bucket directory as well as a large number of

leaf pages containing very few or no key/data values. This fragmented table can lead to increased memory consumption, slower lookups, and unnecessary space usage in the database.

You select whether to perform the system table de-fragmentation by selecting the **Compact System Tables** option in the appropriate Vorkout dialog.



The system table is composed of a number of sub-tables. The **Total AT table count** refers to the number of sub-tables. The **Fragmented AT table count** refers to the number of sub-tables that are currently fragmented. The **Total AT table size** is the total size of the sub-tables in bytes and **Fragmented AT table size** gives the number of bytes of the fragmented sub-tables. If the count of fragmented sub-tables is significantly large in relation to the total or the number of fragmented bytes is over 50% of the total then de-fragmentation may improve the system performance. (These properties of the AT Table can also be viewed in the System property view. Refer to Section 2.13, “System” [p. 50].)

The tool may be run on an online database and there is minimum impact on clients. Running the tool creates a logfile in the database directory called `defrag.log` and writes important system messages there. This logfile is overwritten each time the tool is run. These messages are mostly progress messages but can be error messages if the tool encounters an error.

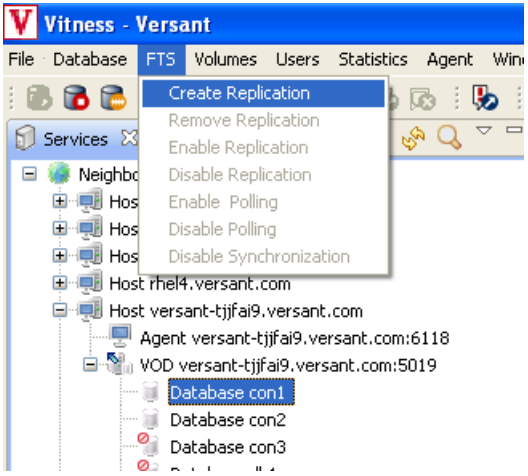
For large databases the defragmentation process might take some time but it will periodically write the progress to the logfile, `defrag.log`, as mentioned. If the de-fragmentation tool detects an error during operation, it will write an error message to the logfile and exit. Since running the de-fragmentation tool again will overwrite the previous logfile, please make a copy of that logfile and send it to Versant for analysis. Along with the de-fragmentation logfile, you should also send the database trace files and the database LOGFILE, as well.

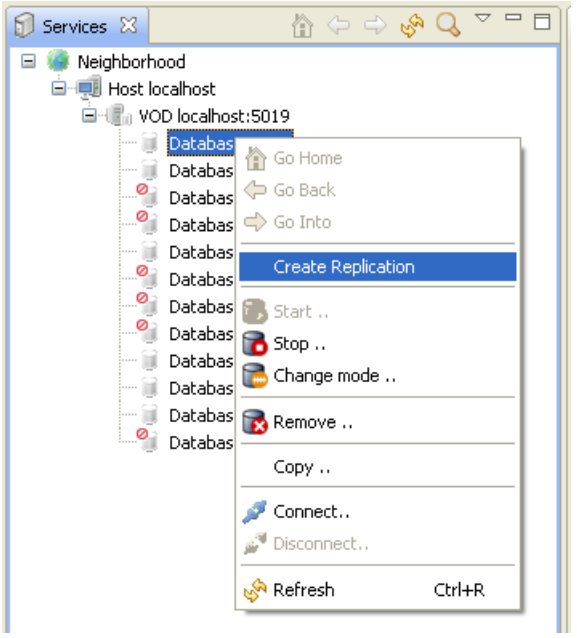
Chapter 7. Vedding

Vedding is currently also known as Fault Tolerant Server - FTS.

The FTS component performs synchronous database replication. Synchronous database replication mirrors the contents of one database in another in a predictable and orderly manner.

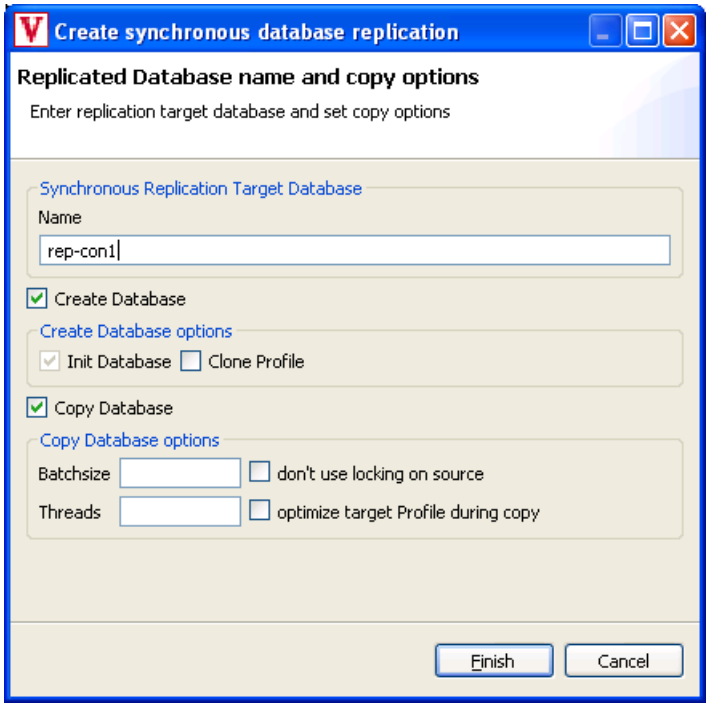
The commands to setup or administer an FTS pair of databases is enabled if a database is selected.





7.1. Create Replication

Setup a replication pair.

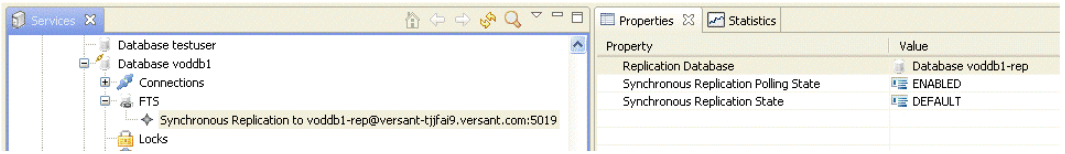


As a default, the target replica database with the given name is created. There are two Database creation options. **Init Database**, with this option the database can be initialized i.e., it grows to its final size. Initializing the database will take a while. The process status can be monitored from the Progress pane. Using the second option **Clone profile**, the `profile.be` can be copied from the source database to the target database.

For more information, please refer to the description of the command line tools createrep, and vcopydb, in the *Versant Database Administration Manual*.

7.2. Manage a replication pair

The status of a replica pair is displayed in a sub-node *FTS* in the database tree.



7.2.1. FTS State Description

An FTS database can be in any one of the following FTS states:

Default/UP

The database is running normally and in sync with its replica pair.

Unstartable/DOWN

The database is in the recovery mode and is waiting for the polling process to complete synchronization before allowing any connections.

Polling

The polling process has started from the database and is trying to connect to its DOWN replica database. Until it is not able to successfully connect, the state of the UP database will be POLLING and all the re-synchronization records will be saved.

Syncing

The polling process has succeeded in connecting to the DOWN database and is performing re-synchronization.

Syncing Done

Polling has completed synchronization process.

Suspend

This implies that the `ftstool –stopsync` has been applied on the database and as a result of which no re-synchronization records are being saved.

Awaiting Restore

Database backup with `–startsync` option has been completed and re-sync records are getting accumulated.

7.2.1.1. Polling State Description

The polling states are:

Enabled

Polling process is activated and will auto start if its replica database goes DOWN.

Disabled

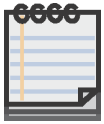
Polling process is deactivated and will not start if its replica database goes DOWN.

`Requested`

Polling process has been requested.

`Forked`

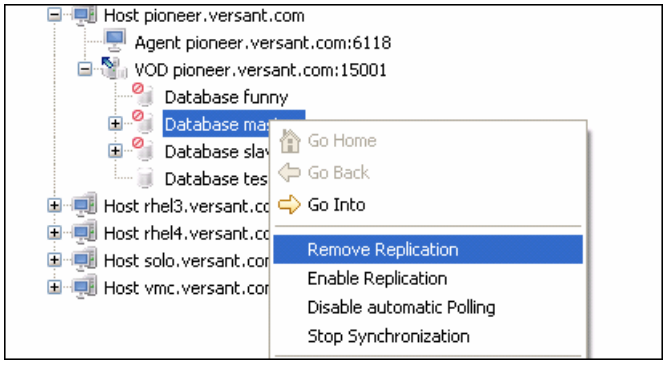
Polling process has started from the UP database.



If the database is authenticated using the password mechanism, then the DBA must ensure that the same password is specified for the replica pair. Otherwise the error SM_E_INVALID_PASSWORD may be thrown.

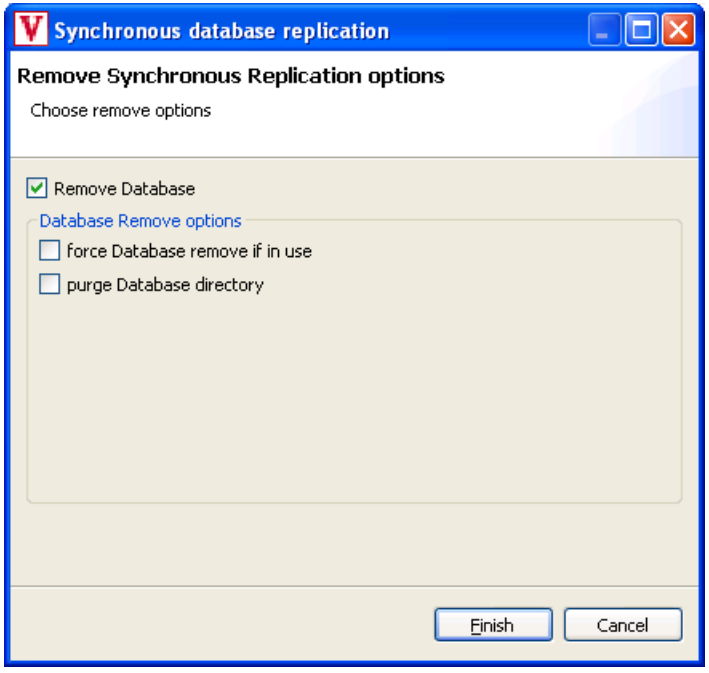
7.3. Replication Administration Commands

Once a replicated pair of databases is set up, the following administration commands are available.



Remove Replication

Removes the entries in both replica files and optionally the partner database as well.



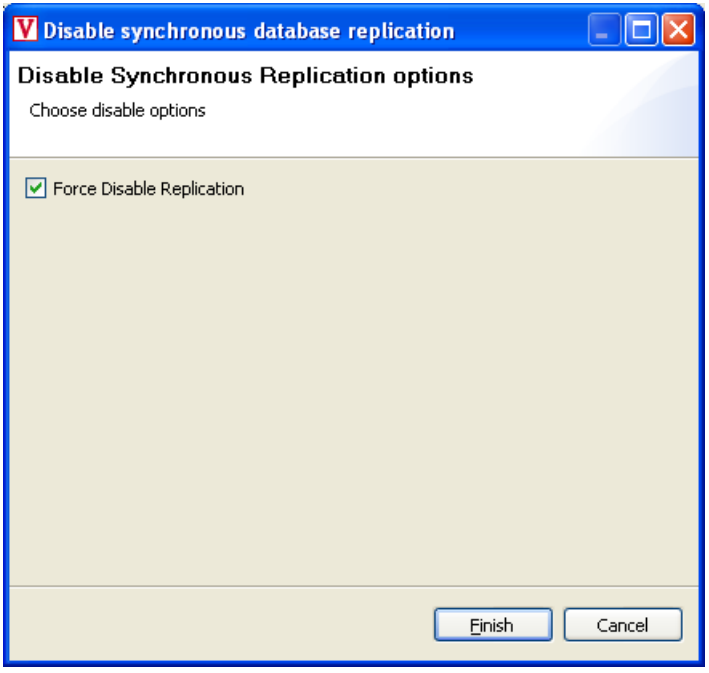
As a default the partner database of the database on which this command is called is removed as well. There are two options available for database removal. It can be removed even if it is still in use i.e., there exists at least one active session.

Clients working on one database will implicitly replicate to both. This means that as long as there is a connected client, it is always connected to both databases. The database directory can be removed as well, or can be kept with the `profile.be` file in it i.e., in a state equal to that after a makedb call.

For more information, please refer to the description of the command line tool removerep in the *Versant Database Administration Manual*.

Enable / Disable replication

This command can be used to manually force a fail-over. It disables polling process on its replica database, waits for the active transactions to complete and then stops the database. In case of long running transactions this option may not be advisable as it waits for the active transactions to complete. In such a case the `-forcedisable` option can be used as shown below.



For more information, please refer to the description of the command line tool ftstool -enable/-disable/-forcedisable, in the *Versant Database Administration Manual*.

Enable / Disable Polling

This is a persistent state of both databases in a replica pair. As a default polling is enabled after setting up a replica pair. When one of the databases fails, the running database will poll for the partner to come online again. Once this happens synchronization will start.

Disabling polling does not influence the ongoing replication. However, implicit polling / synchronization does not occur. Once polling is enabled again the database starts to poll or synchronize as necessary.

For more information, please refer to the description of the command line tool ftstool -p -1 and ftstool -p 0, in the *Versant Database Administration Manual*.

Chapter 8. Vitness Update

Vitness has a built-in mechanism for keeping itself up-to-date by checking for and downloading updates over the Internet.

8.1. Configuring Vitness Update

The update mechanism must have access to your Internet connection. If you are using a proxy, you must configure Vitness Update to use it. Do this by opening the **Preferences** window (select **Preferences** from the **Window** menu) and selecting **Network**. The following dialog is displayed.

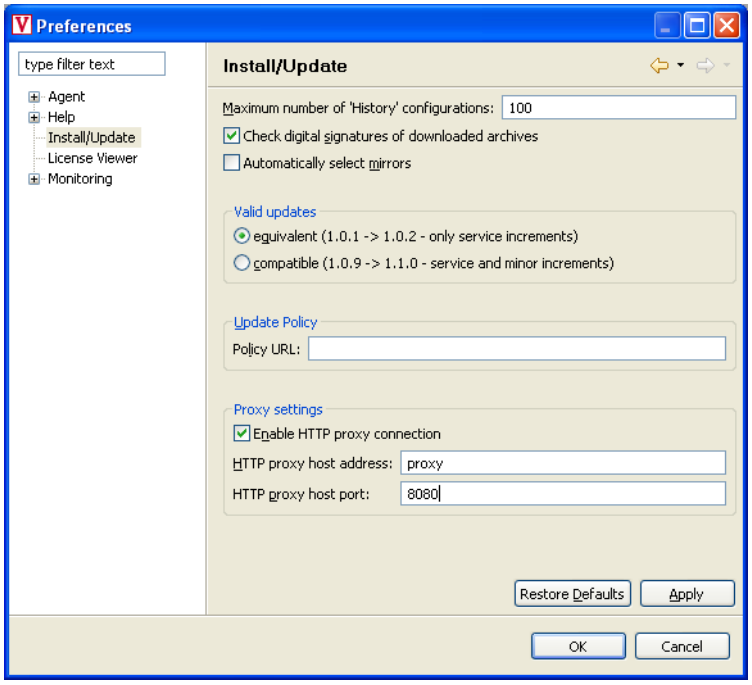


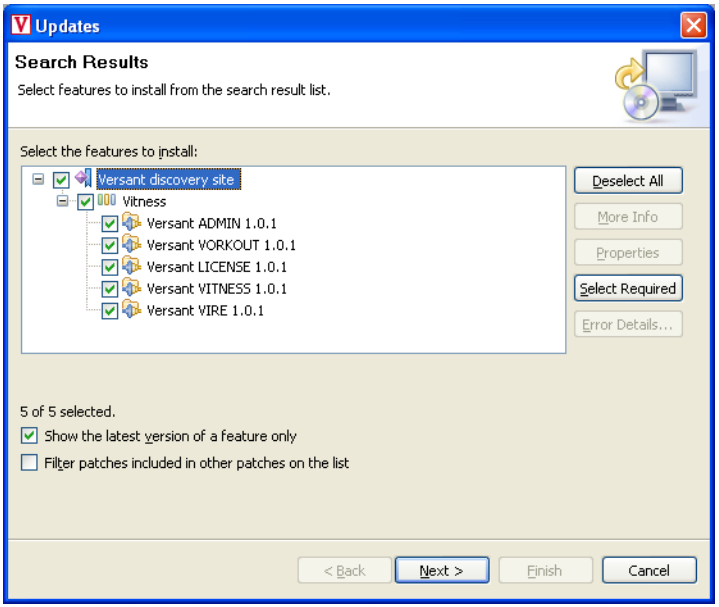
Figure 8.1. Install/Update Preferences

Provide the proxy information as required for your Internet connection and click the **Apply** or **OK** button.

8.2. Performing an Update

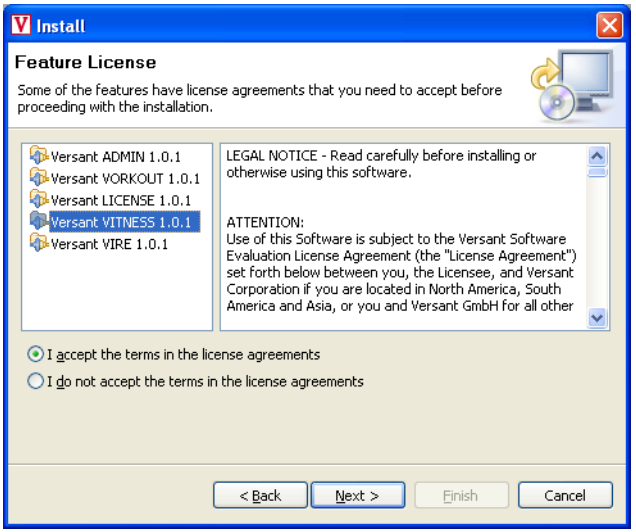
To start an update, select **Software Updates...** from the Vitness **Help** menu. This will start a search for new updates and may take some time. The search progress is shown in the common progress bar in the lower right of the main Vitness window.

Following the search a dialog is displayed as shown in the following figure. If updates are available they will be listed there.



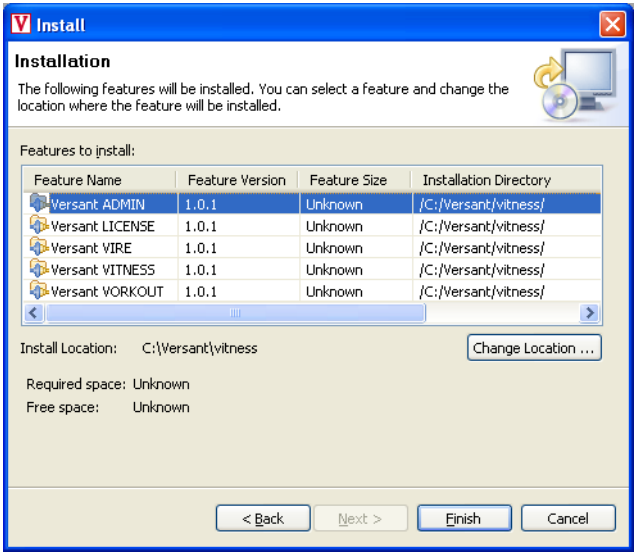
Select all new updates listed under the Versant category and click the **Next>**.

If you are updating any features that have a license agreement you will be asked to accept the agreement before proceeding to the installation.



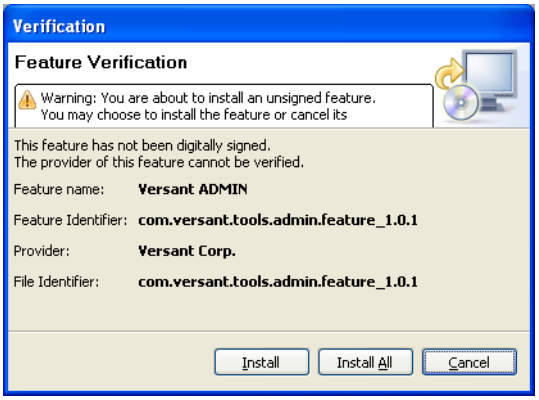
Accept the agreement or not as you desire by clicking the appropriate radio button and then proceed by again clicking **Next>**.

The selected update items are displayed.



Click **Finish** to begin the download.

You may be required to verify the installation of one or more of the selected updates.

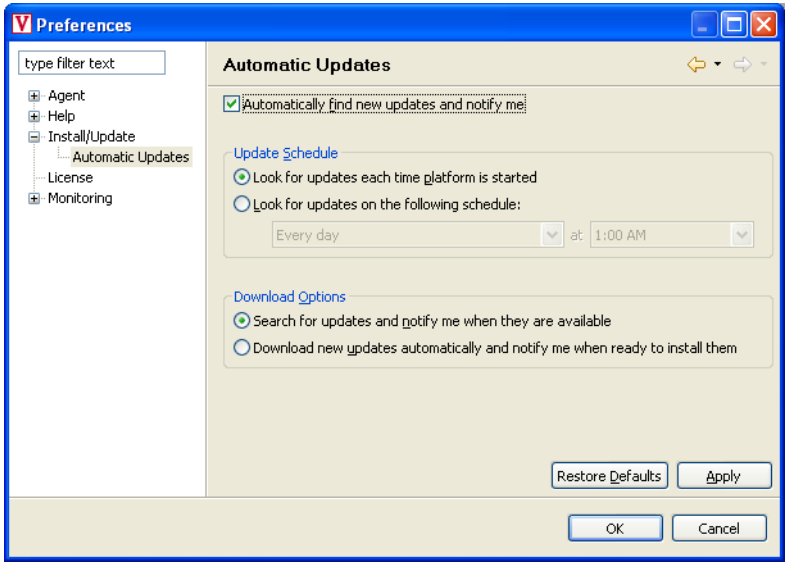


Click the **Install All** button to begin the actual installation.

The progress is displayed in a separate dialog. You can choose to close this progress dialog and run the update in the background. (Progress will continue to be displayed in the **Progress** pane.) Following the installation you will have the option to restart Vitness. The updates will not be in effect until you restart.

8.3. Automatic Update

Vitness can check periodically for updates. You can set automatic update in the global preferences. Select **Preferences** from the **Window** menu. Under **Install/Update** select **Automatic Updates**. The following dialog is displayed.



Check the box for **Automatically find new updates...** and select the desired **Update Schedule** and **Download Options**. Click the **Apply** or **OK** button.

Appendix A. Versant Object Database Statistics

This reference lists all of the statistics available in the Versant Object Database. Viewing statistics in Vitness is discussed in Section 2.6, “Statistics” [p. 20].

A.1. Backend Statistics

`be_base`

`be_begin`
DO NOT ADD BACKEND STATISTICS BEFORE HERE

`be_data_located`
Pages found in database page cache

`be_data_reads`
Pages read from sysvol + added volumes

`be_data_writes`
Pages written to sysvol + added volumes

`be_end`
DO NOT ADD BACKEND STATISTICS AFTER HERE

`be_ev_defined`
Events defined

`be_ev_sys_delivered`
System events delivered

`be_ev_sys_raised`
System events raised

`be_ev_user_delivered`
User events delivered

be_ev_user_raised
User events raised

be_latch_granted
Number of latches granted of any type

be_latch_granted_bf
Number of BF latches granted

be_latch_granted_bf_bkt
Number of BF_BKT latches granted

be_latch_granted_bf_dirty
Number of BF_DIRTY latches granted

be_latch_granted_bf_free
Number of BF_FREE latches granted

be_latch_granted_cp
Number of CP latches granted

be_latch_granted_cp_wait
Number of CP_WAIT latches granted

be_latch_granted_ev
Number of EV latches granted

be_latch_granted_heap
Number of HEAP latches granted

be_latch_granted_l2file
Number of L2FILE latches granted

be_latch_granted_l2file_da
Number of L2FILE_DA latches granted

be_latch_granted_llog
Number of LLOG latches granted

be_latch_granted_lock
Number of LOCK latches granted

be_latch_granted_log_unit
Number of LOG_UNIT latches granted

be_latch_granted_phy
Number of PHY latches granted

be_latch_granted_plog
Number of PLOG latches granted

be_latch_granted_ps
Number of PS latches granted

be_latch_granted_res10__
reserved for future latches

be_latch_granted_res3__
reserved for future latches

be_latch_granted_res4__
reserved for future latches

be_latch_granted_res5__
reserved for future latches

be_latch_granted_res6__
reserved for future latches

be_latch_granted_res7__
reserved for future latches

be_latch_granted_res8__
reserved for future latches

be_latch_granted_res9__
reserved for future latches

be_latch_granted_sc
Number of SC latches granted

be_latch_granted_sce
Number of SCE latches granted

be_latch_granted_sch
Number of SCH latches granted

be_latch_granted_sd
Number of SD latches granted

be_latch_granted_sda
Number of SDA latches granted

be_latch_granted_sdhs
Number of SDHS latches granted

be_latch_granted_sdhs_bkt
Number of SDHS_BKT latches granted

be_latch_granted_ss
Number of SS latches granted

be_latch_granted_st
Number of ST latches granted

be_latch_granted_tr
Number of TR latches granted

be_latch_granted_tre
Number of TRE (transaction entry) latches granted

be_latch_granted_voldev
Number of VOLDEV latches granted

be_latch_released
Number of latches released of any type

be_latch_wait_time
Seconds waiting for any latch

be_latch_wait_time_bf
Seconds waiting for BF latch

be_latch_wait_time_bf_bkt
Seconds waiting for BF_BKT latch

be_latch_wait_time_bf_dirty
Seconds waiting for BF_DIRTY latch

be_latch_wait_time_bf_free
Seconds waiting for BF_FREE latch

be_latch_wait_time_cp
Seconds waiting for CP latch

be_latch_wait_time_cp_wait
Seconds waiting for CP_WAIT latch

be_latch_wait_time_ev
Seconds waiting for EV latch

be_latch_wait_time_heap
Seconds waiting for HEAP latch

be_latch_wait_time_l2file
Seconds waiting for L2FILE latch

be_latch_wait_time_l2file_da
Seconds waiting for L2FILE_DA

be_latch_wait_time_llog
Seconds waiting for LLOG latch

be_latch_wait_time_lock
Seconds waiting for LOCK

be_latch_wait_time_log_unit
Seconds waiting for LOG_UNIT latch

be_latch_wait_time_phy
Seconds waiting for PHY latch

be_latch_wait_time_plog
Seconds waiting for PLOG latch

be_latch_wait_time_ps
Seconds waiting for PS latch

be_latch_wait_time_res10__
reserved for future statistics

be_latch_wait_time_res3__
reserved for future statistics

be_latch_wait_time_res4__
reserved for future statistics

be_latch_wait_time_res5__
reserved for future statistics

be_latch_wait_time_res6__
reserved for future statistics

be_latch_wait_time_res7__
reserved for future statistics

be_latch_wait_time_res8__
reserved for future statistics

be_latch_wait_time_res9__
reserved for future statistics

be_latch_wait_time_sc
Seconds waiting for SC latch

be_latch_wait_time_sce
Seconds waiting for SCE latch

be_latch_wait_time_sch
Seconds waiting for SCH latch

be_latch_wait_time_sd
Seconds waiting for SD latch

be_latch_wait_time_sda
Seconds waiting for SDA latch

be_latch_wait_time_sdhs
Seconds waiting for SDHS latch

be_latch_wait_time_sdhs_bkt
Seconds waiting for SDHS_BKT latch

be_latch_wait_time_ss
Seconds waiting for SS latch

be_latch_wait_time_st
Seconds waiting for ST latch

be_latch_wait_time_tr
Seconds waiting for TR latch

be_latch_wait_time_tre
Seconds waiting for TRE (transaction entry)

be_latch_wait_time_voldev
Seconds waiting for VOLDEV latch

be_latch_waits
Number of waits for any latch

be_latch_waits_bf
Number of waits for BF latch

be_latch_waits_bf_bkt
Number of waits for BF_BKT latch

be_latch_waits_bf_dirty
Number of waits for BF_DIRTY latch

be_latch_waits_bf_free
Number of waits for BF_FREE latch

be_latch_waits_cp
Number of waits for CP latch

be_latch_waits_cp_wait
Number of waits for CP_WAIT latch

be_latch_waits_ev
Number of waits for EV latch

be_latch_waits_heap
Number of waits for HEAP latch

be_latch_waits_l2file
Number of waits for L2FILE latch

be_latch_waits_l2file_da
Number of waits for L2FILE_DA latch

be_latch_waits_llog
Number of waits for LLOG latch

be_latch_waits_lock
Number of waits for LOCK latch

be_latch_waits_log_unit
Number of waits for LOG_UNIT latch

be_latch_waits_phy
Number of waits for PHY latch

be_latch_waits_plog
Number of waits for PLOG latch

be_latch_waits_ps
Number of waits for PS latch

be_latch_waits_res10__
reserved for future latches

be_latch_waits_res3__
reserved for future latches

be_latch_waits_res4__
reserved for future latches

be_latch_waits_res5__
reserved for future latches

be_latch_waits_res6__
reserved for future latches

be_latch_waits_res7__
reserved for future latches

be_latch_waits_res8__
reserved for future latches

be_latch_waits_res9__
reserved for future latches

be_latch_waits_sc
Number of waits for SC latch

be_latch_waits_sce
Number of waits for SCE latch

be_latch_waits_sch
Number of waits for SCH latch

be_latch_waits_sd
Number of waits for SD latch

be_latch_waits_sda
Number of waits for SDA latch

be_latch_waits_sdhs
Number of waits for SDHS latch

be_latch_waits_sdhs_bkt
Number of waits for SDHS_BKT latch

be_latch_waits_ss
Number of waits for SS latch

be_latch_waits_st
Number of waits for ST latch

be_latch_waits_tr
Number of waits for TR latch

be_latch_waits_tre
Number of waits for TRE (transaction entry) latch

be_latch_waits_voldev
Number of waits for VOLDEV latch

be_lock_deadlocks
Deadlocks occurred

be_lock_timeouts
Timeouts waiting for locks

be_lock_wait_time
Seconds clients spent waiting for locks

be_lock_waits
Lock waits which occurred

be_locks_granted
Locks requested and granted

be_net_bytes_read
Bytes read from front end

be_net_bytes_written
Bytes written to front end

be_net_read_time
Seconds reading from front end

be_net_reads
Reads from front end

be_net_rpcs
Database RPCs received from clients

be_net_write_time
Seconds writing to front end

be_net_writes
Writes to front end

be_obe_locks_waiting
1 if waiting for lock, 0 otherwise

be_obj_received
Objects received from front end

be_obj_sent
Objects sent to front end

be_qry_btree_objs
Objects read during B-tree query

be_qry_btree_time
Seconds spent in B-tree query

be_qry_hash_objs
Objects read during hash query

be_qry_hash_time
Seconds spent in hash query

be_qry_scan_objs
Objects read during sequential scan query

be_qry_scan_time
Seconds spent in sequential scan query

be_real_time
Seconds elapsed

be_system_time
Seconds in OS kernel functions

be_user_time
Seconds not in OS kernel functions

be_vm_maj_faults
Virtual memory major page faults

be_xact_active
Active transactions

be_xact_committed
Transactions committed

be_xact_rolled_back
Transactions rolled back

be_xact_started
Transactions started

A.2. Database Statistics

Many of the database statistics have a *delta* implementation. These are the the statistics names beginning with `delta_`. The delta statistics display the data change in the last sample period, as opposed to the normal cumulative statistics.

db_at_leaf_located
number of index and AT leaf pages located

db_at_leaf_read
number of index and AT leaf pages read

db_at_root_located
number of AT root and AT root bucket located

db_at_root_read
number of AT root and AT root bucket pages read

db_base

db_begin
DO NOT ADD DATABASE STATISTICS BEFORE HERE

db_bf_llog_bytes_written
delta_db_bf_llog_bytes_written
bytes written to logical log

db_bf_llog_end
number of logical log buffer end encountered

db_bf_llog_flushes
delta_db_bf_llog_flushes
number of writes to logical log

db_bf_llog_full
number of logical log buffer full encountered

db_bf_plog_bytes_written
delta_db_bf_plog_bytes_written
bytes written to physical log

db_bf_plog_end
number of physical log buffer end encountered

db_bf_plog_flushes
delta_db_bf_plog_flushes
number of writes to physical log

db_bf_plog_full
number of physical log buffer full encountered

db_checkpoints
delta_db_checkpoints
System checkpoints

db_data_located
delta_db_data_located
Pages found in database page cache

db_data_reads
delta_db_data_reads
Pages read from sysvol + added volumes

db_data_writes
delta_db_data_writes
Pages written to sysvol + added volumes

db_disk_free
Bytes of storage available

db_disk_reserved
Bytes of storage reserved by classes

db_end
DO NOT ADD DATABASE STATISTICS AFTER HERE

db_ev_defined
Events defined

db_ev_sys_delivered
System events delivered

db_ev_sys_raised
System events raised

db_ev_user_delivered
User events delivered

db_ev_user_raised
User events raised

db_latch_granted
Number of requests for any latch

db_latch_granted_bf
Number of BF latches granted

db_latch_granted_bf_bkt
Number of BF_BKT latches granted

db_latch_granted_bf_dirty
Number of BF_DIRTY latches granted

db_latch_granted_bf_free
Number of BF_FREE latches granted

db_latch_granted_cp
Number of CP latches granted

db_latch_granted_cp_wait
Number of CP_WAIT latches granted

db_latch_granted_ev
Number of EV latches granted

db_latch_granted_heap
Number of HEAP latches granted

db_latch_granted_l2file
Number of L2FILE latches granted

db_latch_granted_l2file_da
Number of L2FILE_DA latches granted

db_latch_granted_llog
Number of LLOG latches granted

db_latch_granted_lock
Number of LOCK latches granted

db_latch_granted_log_unit
Number of LOG_UNIT latches granted

db_latch_granted_phy
Number of PHY latches granted

db_latch_granted_plog
Number of PLOG latches granted

db_latch_granted_ps
Number of PS latches granted

db_latch_granted_res10__
reserved for future latches

db_latch_granted_res3__
reserved for future latches

db_latch_granted_res4__
reserved for future latches

db_latch_granted_res5__
reserved for future latches

db_latch_granted_res6__
reserved for future latches

db_latch_granted_res7__
reserved for future latches

db_latch_granted_res8__
reserved for future latches

db_latch_granted_res9__
reserved for future latches

db_latch_granted_sc
Number of SC latches granted

db_latch_granted_sce
Number of SCE latches granted

db_latch_granted_sch
Number of SCH latches granted

db_latch_granted_sd
Number of SD latches granted

db_latch_granted_sda
Number of SDA latches granted

db_latch_granted_sdhs
Number of SDHS latches granted

db_latch_granted_sdhs_bkt
Number of SDHS_BKT latches granted

db_latch_granted_ss
Number of SS latches granted

db_latch_granted_st
Number of ST latches granted

db_latch_granted_tr
Number of TR latches granted

db_latch_granted_tre
Number of TRE (transaction entry) latches granted

db_latch_granted_voldev
Number of VOLDEV latches granted

db_latch_released
Number of latches released of any type

db_latch_wait_time
Seconds waiting for any latch

db_latch_wait_time_bf
Seconds waiting for BF latch

db_latch_wait_time_bf_bkt
Seconds waiting for BF_BKT latch

db_latch_wait_time_bf_dirty
Seconds waiting for BF_DIRTY latch

db_latch_wait_time_bf_free
Seconds waiting for BF_FREE latch

db_latch_wait_time_cp
Seconds waiting for CP latch

db_latch_wait_time_cp_wait
Seconds waiting for CP_WAIT latch

db_latch_wait_time_ev
Seconds waiting for EV latch

db_latch_wait_time_heap
Seconds waiting for HEAP latch

db_latch_wait_time_l2file
Seconds waiting for L2FILE latch

db_latch_wait_time_l2file_da
Seconds waiting for L2FILE_DA

db_latch_wait_time_llog
Seconds waiting for LLOG latch

db_latch_wait_time_lock
Seconds waiting for LOCK

db_latch_wait_time_log_unit
Seconds waiting for LOG_UNIT latch

db_latch_wait_time_phy
Seconds waiting for PHY latch

db_latch_wait_time_plog
Seconds waiting for PLOG latch

db_latch_wait_time_ps
Seconds waiting for PS latch

db_latch_wait_time_res10__
reserved for future statistics

db_latch_wait_time_res3__
reserved for future statistics

db_latch_wait_time_res4__
reserved for future statistics

db_latch_wait_time_res5__
reserved for future statistics

db_latch_wait_time_res6__
reserved for future statistics

db_latch_wait_time_res7__
reserved for future statistics

db_latch_wait_time_res8__
reserved for future statistics

db_latch_wait_time_res9__
reserved for future statistics

db_latch_wait_time_sc
Seconds waiting for SC latch

db_latch_wait_time_sce
Seconds waiting for SCE latch

db_latch_wait_time_sch
Seconds waiting for SCH latch

db_latch_wait_time_sd
Seconds waiting for SD latch

db_latch_wait_time_sda
Seconds waiting for SDA latch

db_latch_wait_time_sdhs
Seconds waiting for SDHS latch

db_latch_wait_time_sdhs_bkt
Seconds waiting for SDHS_BKT latch

db_latch_wait_time_ss
Seconds waiting for SS latch

db_latch_wait_time_st
Seconds waiting for ST latch

db_latch_wait_time_tr
Seconds waiting for TR latch

db_latch_wait_time_tre
Seconds waiting for TRE (transaction entry)

db_latch_wait_time_voldev
Seconds waiting for VOLDEV latch

db_latch_waits
Number of waits for any latch

db_latch_waits_bf
Number of waits for BF latch

db_latch_waits_bf_bkt
Number of waits for BF_BKT latch

db_latch_waits_bf_dirty
Number of waits for BF_DIRTY latch

db_latch_waits_bf_free
Number of waits for BF_FREE latch

db_latch_waits_cp
Number of waits for CP latch

db_latch_waits_cp_wait
Number of waits for CP_WAIT latch

db_latch_waits_ev
Number of waits for EV latch

db_latch_waits_heap
Number of waits for HEAP latch

db_latch_waits_l2file
Number of waits for L2FILE latch

db_latch_waits_l2file_da
Number of waits for L2FILE_DA latch

db_latch_waits_llog
Number of waits for LLOG latch

db_latch_waits_lock
Number of waits for LOCK latch

db_latch_waits_log_unit
Number of waits for LOG_UNIT latch

db_latch_waits_phy
Number of waits for PHY latch

db_latch_waits_plog
Number of waits for PLOG latch

db_latch_waits_ps
Number of waits for PS latch

db_latch_waits_res10__
reserved for future latches

db_latch_waits_res3__
reserved for future latches

db_latch_waits_res4__
reserved for future latches

db_latch_waits_res5__
reserved for future latches

db_latch_waits_res6__
reserved for future latches

db_latch_waits_res7__
reserved for future latches

db_latch_waits_res8__
reserved for future latches

db_latch_waits_res9__
reserved for future latches

db_latch_waits_sc
Number of waits for SC latch

db_latch_waits_sce
Number of waits for SCE latch

db_latch_waits_sch
Number of waits for SCH latch

db_latch_waits_sd
Number of waits for SD latch

db_latch_waits_sda
Number of waits for SDA latch

db_latch_waits_sdhs
Number of waits for SDHS latch

db_latch_waits_sdhs_bkt
Number of waits for SDHS_BKT latch

db_latch_waits_ss
Number of waits for SS latch

db_latch_waits_st
Number of waits for ST latch

db_latch_waits_tr
Number of waits for TR latch

db_latch_waits_tre
Number of waits for TRE (transaction entry) latch

db_latch_waits_voldev
Number of waits for VOLDEV latch

db_lock_deadlocks
delta_db_lock_deadlocks
Deadlocks occurred

db_lock_timeouts
delta_db_lock_timeouts
Timeouts waiting for locks

db_lock_wait_time
delta_db_lock_wait_time
Seconds clients spent waiting for locks

db_lock_waits
Lock waits which occurred

db_locks_granted
Locks requested and granted

db_net_bytes_read
delta_db_net_bytes_read
Bytes read from front end

db_net_bytes_written
delta_db_net_bytes_written
Bytes written to front end

db_net_read_time
Seconds reading from front end

db_net_reads
delta_db_net_reads
Reads from front end

db_net_rpcs
delta_db_net_rpcs
Database RPCs received from clients

db_net_write_time
Seconds writing to front end

db_net_writes
delta_db_net_writes
Writes to front end

db_obe_locks_waiting
Connections waiting for locks

db_obj_received
delta_db_obj_received
Objects received from front end

db_obj_sent
delta_db_obj_sent
Objects sent to front end

db_qry_btree_objs
delta_db_qry_btree_objs
Objects read during B-tree query

db_qry_btree_time
delta_db_qry_btree_time
Seconds spent in B-tree query

db_qry_hash_objs
delta_db_qry_hash_objs
Objects read during hash query

db_qry_hash_time
delta_db_qry_hash_time
Seconds spent in hash query

db_qry_scan_objs
delta_db_qry_scan_objs
Objects read during sequential scan query

db_qry_scan_time
delta_db_qry_scan_time
Seconds spent in sequential scan query

db_xact_active
Active transactions

db_xact_committed
delta_db_xact_committed
Transactions committed

db_xact_rolled_back
delta_db_xact_rolled_back
Transactions rolled back

db_xact_started
Transactions started

A.3. Frontend Statistics

fe_base

fe_begin
DO NOT ADD FRONTEND STATISTICS BEFORE HERE

fe_end
DO NOT ADD FRONTEND STATISTICS AFTER HERE

fe_latch_wait_time
Seconds waiting for latch

fe_net_bytes_read
Bytes read from back end

fe_net_bytes_written
Bytes written to back end

fe_net_read_time
Seconds reading from back end

fe_net_reads
Reads from back end

fe_net_write_time
Seconds writing to back end

fe_net_writes
Writes to back end

fe_reads
Objects read into object cache

fe_real_time
Seconds of real time

fe_swapped
Dirty objects swapped out of object cache

fe_swapped_dirty
Objects written as a result of object swapping

fe_system_time
Seconds in OS kernel functions

fe_user_time
Seconds not in OS kernel functions

fe_vm_maj_faults
Virtual memory major page faults

fe_writes
Objects written from object cache

A.4. Session Statistics

se_base

se_begin
DO NOT ADD SESSION STATISTICS BEFORE HERE

se_cods
CODs in object cache

se_end
DO NOT ADD SESSION STATISTICS AFTER HERE

se_heap_allocates
Number of "allocate" operations on front-end heap

se_heap_empty_segments
Number of empty segments in front-end heap

se_heap_free
Bytes free in front-end heap

se_heap_frees
Number of "free" operations on front-end heap

se_heap_max_gap
Size of largest free area in front-end heap in bytes

se_heap_small_allocates
Number of "allocate" operations smaller than 1 page

se_heap_small_allocates_0
Allocates in size class 0 (1-8 bytes)

se_heap_small_allocates_1
Allocates in size class 1 (9-12 bytes)

se_heap_small_allocates_10
Allocates in size class 10 (81-96 bytes)

se_heap_small_allocates_11
Allocates in size class 11 (97-128 bytes)

se_heap_small_allocates_12
Allocates in size class 12 (129-160 bytes)

se_heap_small_allocates_13
Allocates in size class 13 (161-192 bytes)

se_heap_small_allocates_14
Allocates in size class 14 (193-256 bytes)

se_heap_small_allocates_15
Allocates in size class 15 (257-336 bytes)

se_heap_small_allocates_16
Allocates in size class 16 (337-408 bytes)

se_heap_small_allocates_17
Allocates in size class 17 (409-512 bytes)

se_heap_small_allocates_18
Allocates in size class 18 (513-680 bytes)

se_heap_small_allocates_19
Allocates in size class 19 (681-816 bytes)

se_heap_small_allocates_2
Allocates in size class 2 (13-16 bytes)

se_heap_small_allocates_20
Allocates in size class 20 (817-1024 bytes)

se_heap_small_allocates_21
Allocates in size class 21 (1025-1360 bytes)

se_heap_small_allocates_22
Allocates in size class 22 (1361-1632 bytes)

se_heap_small_allocates_23
Allocates in size class 23 (1633-2048 bytes)

se_heap_small_allocates_24
Allocates in size class 24 (2049-2720 bytes)

se_heap_small_allocates_25
Allocates in size class 25 (2721-3072 bytes)

se_heap_small_allocates_26
Allocates in size class 26 (3073-4096 bytes)

se_heap_small_allocates_27
Allocates in size class 27 (4097-5120 bytes)

se_heap_small_allocates_28
Allocates in size class 28 (5121-6144 bytes)

se_heap_small_allocates_29
Allocates in size class 29 (6145-10240 bytes)

se_heap_small_allocates_3
Allocates in size class 3 (17-20 bytes)

se_heap_small_allocates_30
Allocates in size class 30 (10241-12288 bytes)

se_heap_small_allocates_31
Allocates in size class 31 (12289-20480 bytes)

se_heap_small_allocates_4
Allocates in size class 4 (21-24 bytes)

se_heap_small_allocates_5
Allocates in size class 5 (25-32 bytes)

se_heap_small_allocates_6
Allocates in size class 6 (33-40 bytes)

se_heap_small_allocates_7
Allocates in size class 7 (41-48 bytes)

se_heap_small_allocates_8
Allocates in size class 8 (49-64 bytes)

se_heap_small_allocates_9
Allocates in size class 9 (65-80 bytes)

se_heap_small_free
Bytes free for allocations smaller than 1 page

se_heap_small_frees
Number of "free" operations smaller than 1 page

se_heap_small_frees_0
Number of frees in size class 0 (1-8 bytes)

se_heap_small_frees_1
Number of frees in size class 1 (9-12 bytes)

se_heap_small_frees_10
Number of frees in size class 10 (81-96 bytes)

se_heap_small_frees_11
Number of frees in size class 11 (97-128 bytes)

se_heap_small_frees_12
Number of frees in size class 12 (129-160 bytes)

se_heap_small_frees_13
Number of frees in size class 13 (161-192 bytes)

se_heap_small_frees_14
Number of frees in size class 14 (193-256 bytes)

se_heap_small_frees_15
Number of frees in size class 15 (257-336 bytes)

se_heap_small_frees_16
Number of frees in size class 16 (337-408 bytes)

se_heap_small_frees_17
Number of frees in size class 17 (409-512 bytes)

se_heap_small_frees_18
Number of frees in size class 18 (513-680 bytes)

se_heap_small_frees_19
Number of frees in size class 19 (681-816 bytes)

se_heap_small_frees_2
Number of frees in size class 2 (13-16 bytes)

se_heap_small_frees_20
Number of frees in size class 20 (817-1024 bytes)

se_heap_small_frees_21
Number of frees in size class 21 (1025-1360 bytes)

se_heap_small_frees_22
Number of frees in size class 22 (1361-1632 bytes)

se_heap_small_frees_23
Number of frees in size class 23 (1633-2048 bytes)

se_heap_small_frees_24
Number of frees in size class 24 (2049-2720 bytes)

se_heap_small_frees_25
Number of frees in size class 25 (2721-3072 bytes)

se_heap_small_frees_26
Number of frees in size class 26 (3073-4096 bytes)

se_heap_small_frees_27
Number of frees in size class 27 (4097-5120 bytes)

se_heap_small_frees_28
Number of frees in size class 28 (5121-6144 bytes)

se_heap_small_frees_29
Number of frees in size class 29 (6145-10240 bytes)

se_heap_small_frees_3
Number of frees in size class 3 (17-20 bytes)

se_heap_small_frees_30
Number of frees in size class 30 (10241-12288 bytes)

se_heap_small_frees_31
Number of frees in size class 31 (12289-20480 bytes)

se_heap_small_frees_4
Number of frees in size class 4 (21-24 bytes)

se_heap_small_frees_5
Number of frees in size class 5 (25-32 bytes)

se_heap_small_frees_6
Number of frees in size class 6 (33-40 bytes)

se_heap_small_frees_7
Number of frees in size class 7 (41-48 bytes)

se_heap_small_frees_8
Number of frees in size class 8 (49-64 bytes)

se_heap_small_frees_9
Number of frees in size class 9 (65-80 bytes)

se_heap_small_used
Bytes used for allocations smaller than 1 page

se_heap_total_segments
Number of segments in front-end heap

se_heap_used
Bytes used in front-end heap

se_latch_wait_time
Seconds waiting for latch

se_net_bytes_read
Bytes read from back end

se_net_bytes_written
Bytes written to back end

se_net_read_time
Seconds reading from back end

se_net_reads
Reads from back end

se_net_write_time
Seconds writing to back end

se_net_writes
Writes to back end

se_objs
Objects in object cache

se_objs_dirty
Dirty objects in cache

se_reads
Objects read into object cache

se_swapped
Objects swapped out of object cache

se_swapped_dirty
Objects written as a result of object swapping

se_writes
Objects written from object cache

A.5. Derived Statistics

The following are derived from two or more simple statistics.

be_cache_hit_ratio

formula: `be_data_located / (be_data_located + be_data_reads)`

hit ratio for page cache, cumulative

be_cpu_time

formula: `be_system_time + be_user_time`

be_inst_cache_hit_ratio

formula: `% be_data_located / (% be_data_located + % be_data_reads)`

hit ratio for page cache, instantaneous

be_latch_holds

formula: `be_latch_granted - be_latch_released`

Currently held latches

be_run_time

formula: `be_real_time - be_net_read_time - be_net_write_time - be_latch_wait_time`

db_at_cache_hit_ratio

formula: `( % db_at_root_located + % db_at_leaf_located ) / ( % db_at_root_located + % db_at_root_read + % db_at_leaf_located + % db_at_leaf_read)`

db_at_leaf_cache_hit_ratio

formula: `% db_at_leaf_located / ( % db_at_leaf_located + % db_at_leaf_read)`

db_at_root_cache_hit_ratio

formula: `% db_at_root_located / ( % db_at_root_located + % db_at_root_read)`

db_cache_hit_ratio

formula: `db_data_located / (db_data_located + db_data_reads)`

hit ratio for page cache, cumulative

db_inst_cache_hit_ratio

formula: `% db_data_located / (% db_data_located + % db_data_reads)`

hit ratio for page cache, instantaneous

db_latch_holds

formula: `db_latch_granted - db_latch_released`

Currently held latches

db_net_rpcs_per_db_obj_sent

formula: `db_net_rpcs / db_obj_sent`

Number of remote procedure calls for each object sent to the client.

A.6. Reporting and Error Codes

The following are various reporting and error codes associated with statistics collection.

autocollect_already_started

Statistics automatic collection is already turned on.

autocollect_not_started

Statistics automatic collection is not turned on.

autocollect_open

Could not open statistics automatic collection file.

bad_funcs_env

Could not parse VERSANT_STAT_FUNCS environment variable.

bad_stats_env

Could not parse VERSANT_STAT_STATS environment variable.

bad_time_env

Could not parse VERSANT_STAT_TIME environment variable.

eosstats

Unable to obtain operating system statistics

errortable_no_match
Error or message not found in error table.

file_write
Error writing to statistics automatic collection file.

nbuf
No buffer specified

invalid_statistic
Invalid statistic

missing_statistic
Missing one or more statistic

Appendix B. Versant MIB OID Reference

The complete list of managed objects in the Versant MIB is summarized in the following table.

Table B.1. Versant MIB OIDs (ordered by variable name)

Variable Name	OID	Type
mibObjectName	1.3.6.1.4.1.8884.100.1	S
mibObjectOid	1.3.6.1.4.1.8884.100.2	OI
trapDesc	1.3.6.1.4.1.8884.200.25	S
trapLogSeverity	1.3.6.1.4.1.8884.200.10	S
trapMonitorThreshold	1.3.6.1.4.1.8884.200.35	S
trapMonitorType	1.3.6.1.4.1.8884.200.30	S
trapMonitorValue	1.3.6.1.4.1.8884.200.40	S
trapTime	1.3.6.1.4.1.8884.200.5	S
vsntDataVol	1.3.6.1.4.1.8884.5.1.40.1	EN
vsntDataVolTable	1.3.6.1.4.1.8884.5.1.40	TA
vsntDbContact	1.3.6.1.4.1.8884.5.1.10.1.4	S
vsntDbCreateTime	1.3.6.1.4.1.8884.5.1.10.1.7	S
vsntDbEntry	1.3.6.1.4.1.8884.5.1.10.1	EN
vsntDbFreeSpace	1.3.6.1.4.1.8884.5.1.10.1.10	S
vsntDbFullThreshold	1.3.6.1.4.1.8884.5.1.10.1.11	I
vsntDbID	1.3.6.1.4.1.8884.5.1.10.1.1	I
vsntDbMode	1.3.6.1.4.1.8884.5.1.10.1.12	I
vsntDbName	1.3.6.1.4.1.8884.5.1.10.1.2	S
vsntDbOwner	1.3.6.1.4.1.8884.5.1.10.1.6	S
vsntDbPath	1.3.6.1.4.1.8884.5.1.10.1.8	S
vsntDbRelease	1.3.6.1.4.1.8884.5.1.10.1.3	S
vsntDbSize	1.3.6.1.4.1.8884.5.1.10.1.9	S
vsntDbTable	1.3.6.1.4.1.8884.5.1.10	TA
vsntDbType	1.3.6.1.4.1.8884.5.1.10.1.5	S
vsntEnvDbIDNode	1.3.6.1.4.1.8884.2.3	S
vsntEnvDbIDPath	1.3.6.1.4.1.8884.2.4	S
vsntEnvDbPath	1.3.6.1.4.1.8884.2.5	S
vsntEnvRootPath	1.3.6.1.4.1.8884.2.6	S

Variable Name	OID	Type
vsntEnvRuntimePath	1.3.6.1.4.1.8884.2.7	S
vsntLogActualSize	1.3.6.1.4.1.8884.5.1.20.1.5	I
vsntLogFreeKBytes	1.3.6.1.4.1.8884.5.1.20.1.6	I
vsntLogID	1.3.6.1.4.1.8884.5.1.20.1.1	I
vsntLogInitialSize	1.3.6.1.4.1.8884.5.1.20.1.4	I
vsntLogIsOnRawDev	1.3.6.1.4.1.8884.5.1.20.1.7	I
vsntLogName	1.3.6.1.4.1.8884.5.1.20.1.2	S
vsntLogPath	1.3.6.1.4.1.8884.5.1.20.1.3	S
vsntLogVol	1.3.6.1.4.1.8884.5.1.20.1	EN
vsntLogVolTable	1.3.6.1.4.1.8884.5.1.20	TA
vsntPrivateMibOID	1.3.6.1.4.1.8884.2.1	OI
vsntVendorName	1.3.6.1.4.1.8884.2.2	S
vsntVolExtSize	1.3.6.1.4.1.8884.5.1.40.1.5	I
vsntVolFreeExt	1.3.6.1.4.1.8884.5.1.40.1.7	I
vsntVolFreeKBytes	1.3.6.1.4.1.8884.5.1.40.1.8	I
vsntVolFullThreshold	1.3.6.1.4.1.8884.5.1.40.1.9	I
vsntVolID	1.3.6.1.4.1.8884.5.1.40.1.1	I
vsntVolName	1.3.6.1.4.1.8884.5.1.40.1.2prefaceContactInfo	S
vsntVolNumExt	1.3.6.1.4.1.8884.5.1.40.1.6	I
vsntVolPath	1.3.6.1.4.1.8884.5.1.40.1.3	prefaceContactInfo S
vsntVolSize	1.3.6.1.4.1.8884.5.1.40.1.4	I

The types in the table above are defined as follows.

I	Integer
S	Character String
OI	Object ID
EN	Enumeration
TA	Table

Appendix C. References

D. Perkins, E. McGinnis, *Understanding SNMP MIBs* (Prentice Hall PTR, 1997)

M. Sloman (Ed.), *Network and Distributed Systems Management* (Addison-Wesley, 1994)

RFC-1155, *The Structure and Identification of Management Information for TCP/IP-Based Internets*

RFC-1212, *Concise MIB Definitions*

RFC-1213, *Management Information Base for Network Management of TCP/IP-based Internets: MIB-II*

RFC-1215, *A Convention for Defining Traps for Use with SNMP*

Solaris Enterprise Agents http://www.sun.com/software/entagents/

For information about the Sun SNMP Toolkit refer to the Sun Microsystems - Solstice Enterprise Agents User Guide

Service Location Protocol (SLP) is formally defined in RFC 2608

Standardized APIs for the Service Location Protocol are discussed in RFC 2614.

The jSLP API reference can be found at http://jslp.sourceforge.net/apidocs/index.html

Additional information for the Service Location Protocol can be found at http://openslp.sourceforge.net/

C.1. Obtaining RFCs

The official Requests for Comments can be downloaded from many sites on the Internet where RFC repositories are maintained. One of the many sites where RFCs can be downloaded is the RFC repository maintained by the IETF Secretariat at http://www.ietf.org/rfc.html .