Management Guide

VoltDB, Inc.

Abstract

This books explains how to set up, monitor, and manage VoltDB databases and the clusters that run them.

V2.8.4

Management Guide

VoltDB, Inc.

V2.8.4

Copyright © 2010-2012 VoltDB Inc.

The text and illustrations in this document are licensed under the terms of the GNU General Public License Version 3 as published by the Free Software Foundation. See the GNU General Public License (`http://www.gnu.org/licenses/`) for more details.

Some of the features described in this documentation are specific to the VoltDB Enterprise Edition, which is distributed by VoltDB, Inc. under a commercial license. Many of the core VoltDB database features described herein are also part of the VoltDB Community Edition, which is licensed under the GNU Public License 3 as published by the Free Software Foundation. Wherever possible, the documentation distinguishes between features that are unique to the Enterprise Edition, and those available in both the Enterprise and Community editions. Your rights to access and use VoltDB features described herein are defined by the license you received when you acquired the software.

This document was generated on November 01, 2012.

Table of Contents

List of Figures

List of Tables

Preface

This book explains how to set up, monitor, and manage VoltDB databases and the clusters that host them.

It is possible to perform the functions described in this book using the VoltDB Community Edition database software and other open source software. However, many of these functions are automated and simplified by using the Enterprise Manager, which is a feature of the VoltDB Enterprise Edition. Therefore, this book describes the Enterprise Manager as the primary interface. Notes are provided at the end of each section for those performing the functions without the assistance of the management tool.

This book assumes the reader is already familiar with VoltDB and the basic operation of a VoltDB database. If you are not familiar with VoltDB, we recommend reading *Getting Started With VoltDB* before reading this book.

1. Structure of This Book

This book is divided into eight chapters and three appendices:

- Chapter 1, *Managing VoltDB Databases*
- Chapter 2, *Installing and Starting the VoltDB Enterprise Manager*
- Chapter 3, *Getting Started with the Enterprise Manager*
- Chapter 4, *Adding a Database*
- Chapter 5, *Starting and Stopping the Database*
- Chapter 6, *Monitoring the Cluster*
- Chapter 7, *Updating the Database*
- Chapter 8, *Maintaining and Repairing the Cluster*
- Appendix A, *Server Configuration Options*
- Appendix B, *VoltDB Management REST Interface*
- Appendix C, *Snapshot Utilities*

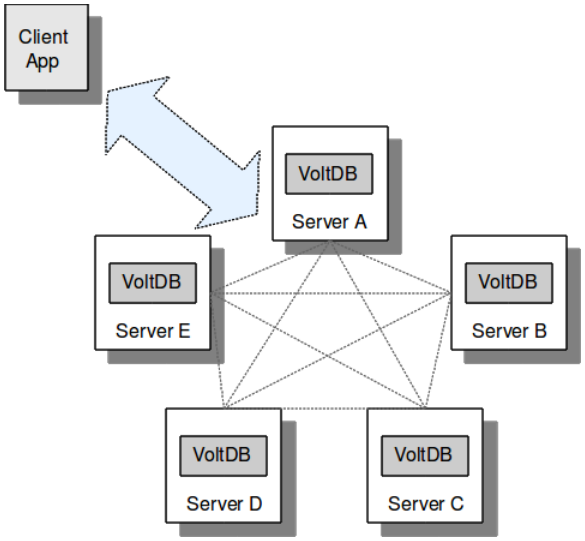
2. Related Documents

This book does not describe how to design or develop VoltDB databases. For a complete description of the development process for VoltDB and all of its features, please see the accompanying manual *Using VoltDB*, available on the web from http://www.voltdb.com/.

Chapter 1. Managing VoltDB Databases

VoltDB database applications use a standard client-server model, with the database running on one or more servers and the calling application(s) running as one or more clients. The servers form a cluster that manages the data and distribution of work internally.

When you first learn about VoltDB, or while developing your VoltDB application, it is easiest to run both the client and the server on the same machine. However, the goal of VoltDB is to achieve the best, most scalable throughput possible for OLTP applications. To do this in a production environment, you normally run the VoltDB database server on a cluster of nodes, with one or more client applications running on separate machines.



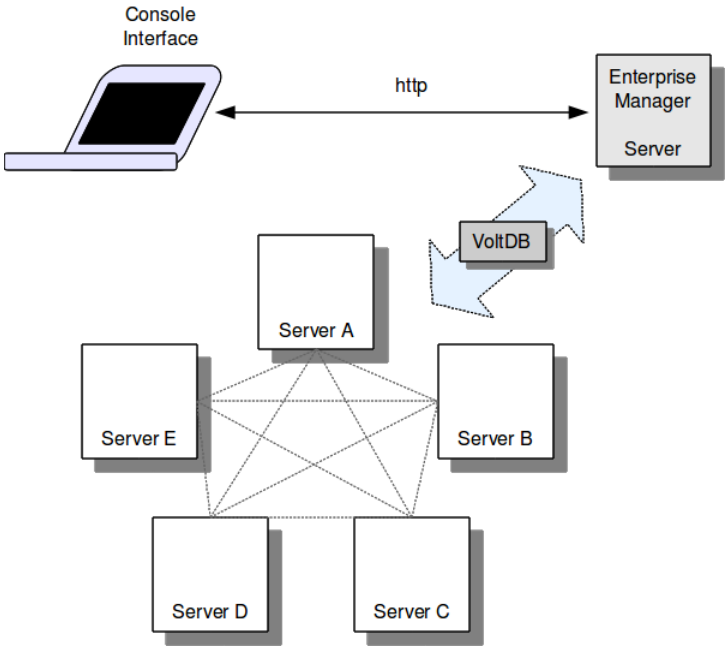
Starting and stopping a VoltDB database server on a single node is easy. The more nodes there are on the cluster, the more tedious it becomes to start and manage the database server by hand.

VoltDB provides a management console to simplify this process. For the most part, this book explains how to perform these tasks using the VoltDB Enterprise Manager, which is a VoltDB Enterprise Edition feature. However, it is possible to perform the same tasks manually or through scripts and other open source tools and technology. At the end of each chapter is a summary of other approaches for those who choose not to use the VoltDB Enterprise Manager.

1.1. Using the VoltDB Enterprise Manager

The VoltDB Enterprise Manager is a tool for simplifying the work of the VoltDB database administrator and operator. The VoltDB Enterprise Manager consists of server software (which provides the management functions and controls the database cluster) and a web-based management console that the database administrator uses to issue commands and evaluate the state of the cluster.

When you run the VoltDB Enterprise Manager, it creates its own dedicated web server. You can then connect to that server from any web browser. The web interface, or console, allows you to manage your VoltDB clusters from virtually anywhere.



Using the VoltDB Enterprise Manager, you do not need to pre-install VoltDB on each node. The management console handles distributing both the VoltDB software and the database catalogs to the cluster nodes. The console also helps by providing a common interface for:

- Initiating and collecting snapshots
- Providing live statistics on database volume and performance
- Comparing and updating the database catalog and schema
- Dropping and rejoining nodes to a high availability cluster (using K-Safety)

In addition to a web-based graphical interface, the Enterprise Manager contains a REST programming interface, allowing database managers to script any of these functions for further customization and simplification of the management tasks. The VoltDB REST interface is described later in this book in Appendix B, *VoltDB Management REST Interface*.

1.2. Managing Databases

There are a number of management tasks that a database administrator needs to perform. These responsibilities fall into three main categories:

Table 1.1. Database Management Tasks

Basic Operations	Administrators are often responsible for configuring the hardware, software, and the database that runs on them. The management console helps by providing a single interface from which you can define the configuration of a VoltDB database, verify the setup of the operating systems, and distribute the VoltDB software and database catalog to all nodes of the cluster in a single step. The console also lets you start and stop clusters or individual nodes from one location.

Performance Monitoring	Another important role of the database administrator is monitoring database performance. Monitoring is important for several reasons: • Performance • Load Balancing • Fault Detection The management console provides tools to help in all three cases.
Maintenance and Upgrades	Over time, both a cluster and the database may require maintenance — either planned or emergency — and possibly upgrades. The ability for a VoltDB cluster with K-safety enabled to bring down a node for repair and bring it (or a replacement) back into the cluster addresses the need for hardware and operating system maintenance. VoltDB Enterprise Edition also provides a mechanism for updating the database catalog itself, on the fly, with a single command from within the management console.

The chapters of this book describe each of these tasks and how to accomplish them with VoltDB in more detail. But first you need to know how to set up the management console, which is described in Chapter 2, *Installing and Starting the VoltDB Enterprise Manager*.

Chapter 2. Installing and Starting the VoltDB Enterprise Manager

The VoltDB Enterprise Manager simplifies the process of managing VoltDB database clusters. However, you must first make sure the management server and database servers that will be managed are set up properly to allow the Enterprise Manager to do its work. This chapter explains how to set up and start the Enterprise Manager, including:

- Operating System and Software Requirements
- Installing the VoltDB Enterprise Manager
- Preparing the Cluster Nodes for Management
- Starting the Enterprise Manager
- Configuring Access to the Enterprise Manager
- Applying Your Enterprise License

2.1. Operating System and Software Requirements

The requirements for managing a database cluster are identical to the requirements for running a VoltDB database (see the *Using VoltDB* manual for details), with the addition that each node must be running SSH for communication with the management server. Table 2.1, "Operating System and Software Requirements: Database Cluster Nodes" summarizes these requirements.

Table 2.1. Operating System and Software Requirements: Database Cluster Nodes

Operating System	CentOS version 5.6 or later or Ubuntu (10.04 or later)
CPU	• Dual core x86_64 processor • 64 bit • 1.6 GHz
Memory	4 Gbytes
Java	Sun JDK 6 update 20 or later
Required Software	NTP SSH

The server running the VoltDB Enterprise Manager itself does not have the same strict requirements for memory and performance, but is assumed to be running Linux and SSH. Table 2.2, "Operating System and Software Requirements: Management Tool Host" summarizes these requirements.

Table 2.2. Operating System and Software Requirements: Management Tool Host

Operating System	CentOS (5.6 or later), Ubuntu (10.04 or later), or Macintosh OSX (10.6)
Java	Sun JDK 6 update 20
Required Software	SSH

2.2. Installing the VoltDB Enterprise Manager

The VoltDB Enterprise Manager is part of the VoltDB Enterprise Edition. You install the Enterprise Edition the same way you install the VoltDB Community Edition (as described in the *Getting Started* manual) by unpacking a tar archive onto your system. When you install the Enterprise Edition, there is an additional folder, `management`, that contains the Enterprise Manager software and configuration files.

You can run the Enterprise Manager directly from the `management` folder where it is installed. However, you might choose to run the Manager from a server other than the one you use for application development. In that case, you can either install the full VoltDB Enterprise Edition kit or you can simply copy the contents of the `management` folder to the server where you want to run the management suite.

2.3. Preparing the Cluster Nodes for Management

The Enterprise Manager comes complete with all of the software needed to run VoltDB. When you start a VoltDB cluster using the management console, all of the necessary files including the VoltDB software and the database catalog and deployment files are copied to all nodes automatically. So you do not need to manually install any VoltDB software on the cluster nodes themselves

However, you must prepare the cluster nodes so that the management console can communicate with them. The Enterprise Manager uses SSH to perform many functions, including copying files to the cluster nodes and starting the database process. (The Manager also uses native VoltDB system procedures to interact with the database once it is up and running.)

Before you start the Enterprise Manager, you must make sure the server can reach the cluster nodes using SSH. SSH should be set up with public keys for enhanced security and password-less access. If you are using SSH with public keys, you need to create an SSH key file on the server where the Manager will run. You then copy the appropriate key files to an account on each of the cluster nodes.

Once you believe you have SSH set up properly, it is a good idea to test the configuration before starting the Enterprise Manager. From the command line on the server that will run the Enterprise Manager process, use the ssh command to connect to each of the database servers that will be used. If the connection succeeds, the configuration is correct. If the connection requests a password before continuing (or generates a permission error), the configuration is not correct. In the following example, the user tests the connection to three servers: first using the default account on Athens, then using the account romulus on Rome, and finally the account napolean and the keyfile `.ssh/mykey.key` on Paris:

```
$ ssh athens
 [...]
$ ssh romulus@rome
 [...]
$ ssh napolean@paris -i .ssh/mykey.key
```

If you are unfamiliar with SSH, there are instructions for setting it up available on the web. For example `http://www.debian-administration.org/articles/152`.

2.4. Starting the Enterprise Manager

Once you install the management software and set up SSH, you are ready to start the Enterprise Manager. The Enterprise Manager comes with three files:

- `enterprise_manager.jar` — This is the Enterprise Manager software. When you run this Java JAR file, it creates the Enterprise Manager process and web server.
- `users.xml` — This is the security configuration file for the Enterprise Manager web interface. The Enterprise Manager is protected by a username and password. The configuration file defines what username and password pairs are allowed to access the console interface. See Section 2.5, “Configuring Access to the Enterprise Manager” for information on modifying the security settings.
- `enterprise_manager.sh` — is a shell script for starting the Enterprise Manager. Use this script to start the Enterprise Manager process.

To use the default settings, change directory to the folder where you installed the software and invoke the Enterprise Manager process using the included shell script. For example:

```
$ cd ~/voltdb-ent-2.8.4/management
$ ./enterprise_manager.sh
```

When you start the Enterprise Manager, it creates a dedicated web server on the current system (using port 9000). To access the management console interface, you connect to the console server from a web browser.

2.4.1. Using a Different HTTP Port

If you wish to change the HTTP port that the Enterprise Manager uses, you can specify a different port number as an argument to the script (using `-p`). This then becomes the port number you specify when accessing the console from your web browser. For example, the following command starts the Enterprise Manager on port 8181:

```
$ ./enterprise_manager.sh -p 8181
```

2.4.2. Using a Different Configuration Directory

You can also specify what directory the Enterprise Manager uses to store configuration information. By default, it creates the hidden directory `~/.voltdb` to store its configuration information. If you want to a different configuration directory, use the `-m` argument. For example, the following command starts the Enterprise Manager using the directory `/etc/voltdb/` as the configuration directory:

```
$ ./enterprise_manager.sh -m /etc/voltdb/
```

2.5. Configuring Access to the Enterprise Manager

By default, the Enterprise Manager is configured to allow access to the user *admin* with the password *voltdb*. You can change these defaults or add access for other users by editing the `management/users.xml` provided in the distribution kit.

Add `<user>` tags to add new accounts. The accounts must have the `role` attribute set to "admin" to get access to the console interface, For example, the following modified `users.xml` file removes the default account and adds two others, *root* and *ops*:

```
<?xml version="1.0"?>
<tomcat-users>
  <user username="root" password="No1GetsIn" roles="admin"/>
  <user username="ops"  password="CcureET" roles="admin"/>
</tomcat-users>
```

2.6. Applying Your Enterprise License

The VoltDB Enterprise Edition ships with a trial license, letting you try the Enterprise Manager for free on a limited time basis. When you purchase the Enterprise Edition you will want to replace the trial license with the commercial license provided by VoltDB.

The enterprise license is an XML file (license.xml). When you start the Enterprise Manager, the software looks for the license first in the subfolder (voltdb) where the software JAR files are installed, then in the VoltDB configuration directory.

When you receive a commercial license from VoltDB, you can replace the trial license, or copy the commercial license to the VoltDB configuration directory (`~/.voltdb` by default). If you don't replace the trial license, be sure to delete it so it does not override the commercial license.

Note that a license in an installation directory or in the configuration directory may be overwritten or deleted as part of an upgrade. Therefore, it is a good idea to keep a copy of the license file outside the VoltDB directories in case you need to reinstate it following an upgrade.

It is possible to store your license in an alternate directory and specify its location on the command line when you start the Enterprise Manager. If you store the license in a directory other than one of the two default locations, use the `-l` argument (lowercase "L") to specify the location when starting the Enterprise Manager. For example, the following command starts the Enterprise Manager using a license stored in a system-wide directory as `voltdb-license.xml`:

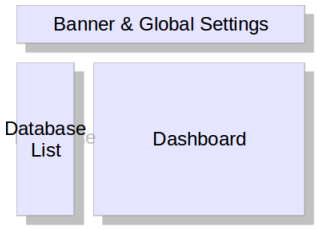
```
$ ./enterprise_manager.sh -l /etc/voltdb-license.xml
```

Chapter 3. Getting Started with the Enterprise Manager

The VoltDB Enterprise Manager lets you define, administer, and monitor VoltDB databases from a single easy-to-use web interface. You can manage multiple databases and multiple servers all from a single instance of the Enterprise Manager.[1]

To do this, the Enterprise Manager presents you with a dashboard for viewing the activity within a database, modifying its settings, and starting and stopping either the database or individual servers. But before you get started, it is a good idea to understand how the dashboard operates. This chapter introduces you to the components of the dashboard and how to use them. The following chapters explain how to use the dashboard to perform specific activities.

3.1. The Components of the Interface



The Enterprise Manager interface is made up of three main components:

- The dashboard
- A list of the databases controlled by the Enterprise Manager
- A banner and global server list

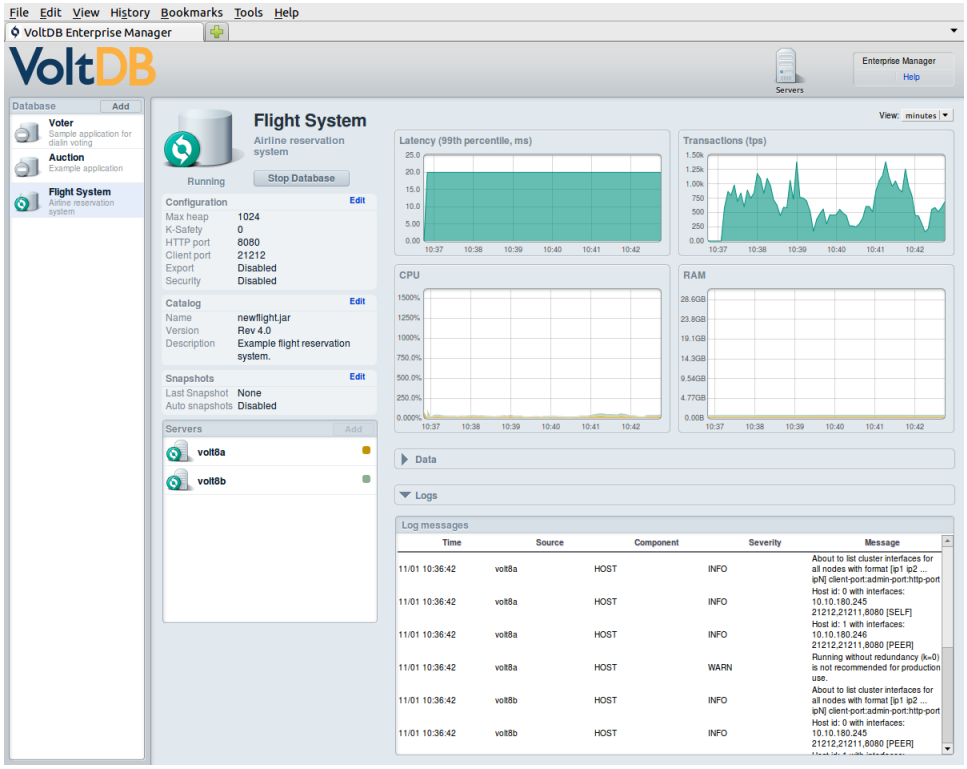
3.1.1. Database Dashboard

The main component of the Enterprise Manager interface is the dashboard. The dashboard lets you examine the configuration and performance of one database at a time.

[1]The Enterprise Manager manages databases that are defined and started using the web console or the REST interface only. It cannot correctly discover or manage databases started manually, and it cannot detect management actions performed manually through system procedures such as @UpdateLogging or @UpdateApplicationCatalog.

Mixing use of the Enterprise Manager and manual invocations of system procedures that modify the database is neither recommended nor supported. For each database (and its associated servers), use either the Enterprise Manager or manual processes for starting and managing the database.

Note that this restriction applies to procedures that modify the database schema or server configuration only. The use of read-only or non schema modifying system procedures does not impact the management of the database.



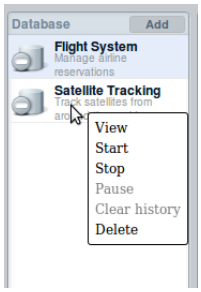
The left side of the dashboard summarizes the configuration of the database, including security settings, K-safety, snapshotting, and export. There are separate panels for database configuration, catalog selection, and snapshot management. Click on the **Edit** button in each panel to modify the settings.

The configuration information also includes a list of the servers assigned to the database. Click on the **Add** button to add servers to the database. By clicking on the name of a server in the list you can choose to start or stop it (depending on the current state of the database and the server). You can also remove the server from the list or replace it with a different server.

The right side of the dashboard provides real-time information about the performance of the database, including summaries of latency, throughput (transactions per second), memory, and CPU usage, as well as detailed tables showing the volume of data per table and invocations per procedure. The right side of the dashboard also includes a log viewer for reviewing any log messages generated by the individual servers or the management tool itself.

Depending on what information you want to review, you can switch the data table between views of data volume or procedure invocations. You can also expand/collapse the display of the data tables or the log viewer — or both — by clicking on the heading above each section of the dashboard.

3.1.2. List of Databases



The list of databases on the left side of the dashboard lets you select which database to view in the dashboard. It also gives you a quick view of the status of all your databases.

To view a database in the dashboard, click on its name in the global list of databases and select **view** from the popup menu. There are additional actions you can take directly from the list, including starting and stopping the database, putting it in admin mode (pause), or removing it from the list (as long as it is not running).

Each database in the list displays an icon indicating its current state. You can use these icons to tell quickly whether there are any issues with your database infrastructure. Table 3.1, “Database Status Icons” explains the meaning of each icon.

Table 3.1. Database Status Icons

	Database is not running.
	Database is running properly.
	Database is running in admin mode (paused). The database is available for administrative access, but other clients cannot queue transactions until the database "resumes" and leaves admin mode.
	Database is running, but not as configured. This means one or more servers are not running and the database is not at its originally specified K-safety value. This may be caused by an error or intentional action (for example, if the operator stops a server).
	An unexpected error has occurred and the database is no longer running. Note that if you stop the database explicitly, the icon changes to gray (stopped). The red icon only occurs if the database stops unexpectedly due to errors of some kind.

3.1.3. Banner and Global Server List

The section of the interface above the dashboard includes the VoltDB banner, the global server list, and the Help menu. The global server list icon and the help menu are to the far right of the banner.

The help menu brings up the About dialog box. The About box lists the currently installed version of the Enterprise Manager, information about the active license, and a link to the online documentation.

Clicking on the Servers icon brings up a list of all of the servers known to the Enterprise Manager. When you define a database in the Enterprise Manager, you must assign servers to run the database before the database can be started. When you add servers to a database, those servers are automatically added to the global server list.

When you remove a server from its assignment to a database in the dashboard, it does not remove the server from the Enterprise Manager. It stays so it can be assigned to another database later. If you want to remove the server definition entirely, open the global server list, click on the server name and select **Delete server** from the popup menu. (The server must be removed from all databases before it can be deleted.)

Another use of the global servers list is to show you the databases to which a server is assigned and their current status. Click on a server name to see its properties and the status of the databases in which it participates.

3.2. Starting the Enterprise Manager

After you install and start the Enterprise Manager, it establishes a web server that you can connect to from a web browser[2] to display the management interface. So, for example, if you start the Enterprise Manager on the server zeus, you connect to the url `http://zeus:9000/` to access the dashboard.

When you first connect, the Enterprise Manager asks for your username and password. The Enterprise Manager is secured against anonymous access. If you have not modified the default security settings, use the username *admin* and the password *voltdb* to access the management console. You will be asked for your credentials every time you connect to the Enterprise Manager from a new browser session.

Once you have logged in you are ready to use the dashboard. The first time you start the Enterprise Manager, there are no databases defined. So your next step, after logging in, is to create your first database. The next chapter explains how to do that.

[2]VoltDB Enterprise Manager is tested against and supports the Chrome, Firefox, and Safari browsers. Use of other browsers for accessing the web interface may work, but is not recommended.

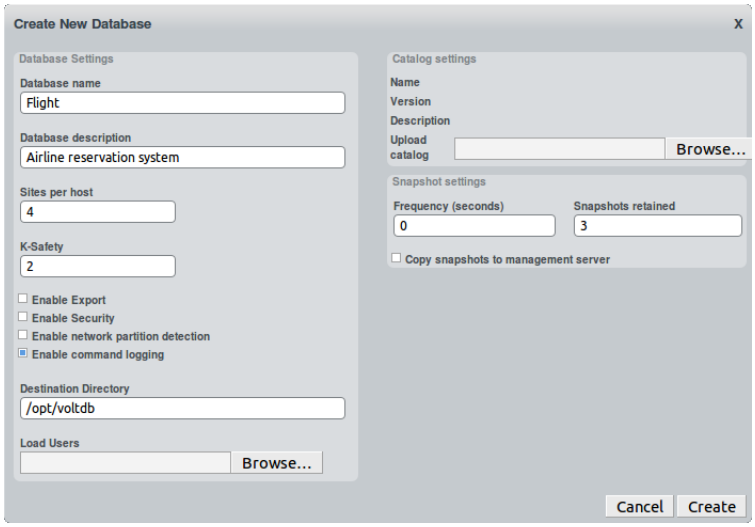
Chapter 4. Adding a Database

The Enterprise Manager makes it easy to manage databases. The first step is to add the database to the management interface, specifying how you want the database configured, in terms of sites per host, K-safety, and so on.

4.1. Configuring the Database

To add a new database to the Enterprise Manager, click on the **Add** button at the top of the database list, which brings up the Add Database dialog box. (If this is your first time using Enterprise Manager or if you have no databases defined, the Add Database dialog is displayed automatically.)

Figure 4.1. Add Database Dialog



The Add Database dialog box lets you set the basic properties for the database, including the application catalog, the number of sites per host, the K-safety value, and so on. You can also load user definitions and set up automated snapshots.

Table 4.1, “Database Configuration Settings” describes each of the fields in the dialog box in detail. In some cases you can take the default settings or leave the field blank. But the four areas of the configuration screen you want to make sure to fill out are the database name, the application catalog, the partitioning and K-safety settings, and the destination directory.

- The *name and description* are what the Enterprise Manager uses to identify your database. The name is shown in the list of databases on the left and the description is included on the dashboard. Provide a meaningful name that helps you identify your database.

- The *application catalog* defines the schema and stored procedures that your database will use. You must load an existing catalog before you can complete the Add Database dialog.

- The *sites per host* and *K-safety* fields are important aspects of the database configuration. Sites per host indicates how many partitions are created on each node of the cluster and K-safety specifies how many replicates are created (to ensure availability in case of node failure). See the *Using VoltDB* guide for more information about these features. However, it is important to make sure these settings are appropriate for your hardware configuration and the needs of your database application.

- The *destination directory* specifies where the Enterprise Manager puts files it needs to run VoltDB on the individual servers. The Enterprise Manager will create this directory if it doesn't already exist. But make sure that the SSH user account has permissions to create and write into the specified directory.

Table 4.1. Database Configuration Settings

Database Settings	
Field	**Description**
Database name	A short, meaningful name identifying the database. The Enterprise Manager uses this name to identify the database in lists and dialog boxes.
Database description	A longer description of the purpose of the database. The description is shown in the dashboard only.
Sites per host	The number of sites to create on each cluster node.
K-safety	The K-safety value you wish to set for the database cluster. See the chapter on "Availability" in *Using VoltDB* for more information about K-safety.
Enable export	Check this box if you want to enable export. To use export, you must have tables marked for export in the catalog and have an appropriate export client to receive the export data. (Note that you must run the export client manually. The Enterprise Manager does not run the client for you.) See the chapter on "Exporting Live Data" in *Using VoltDB* for more information
Enable security	Check this box if you want to enable security. If you enable security, your catalog must include the definition of one or more roles and you must import a set of user definitions using the "Load Users" function described below. See the chapter on "Security" in *Using VoltDB* for more information
Enable network partition detection	Check this box if you want to enable network partition detection. This feature is only necessary if you are running with K-safety enabled in an environment susceptible to intermittent network failures. See the chapter on "Availability" in *Using VoltDB* for more information
Enable command logging	Check this box if you want to enable command logging. Command logging enhances availability by creating a record of all the stored procedure invocations as they occur. Then, in case of unexpected failure, the log can be "replayed" automatically on startup, recreating the database contents. See the chapter on "Command Logging and Recovery" in *Using VoltDB* for more information.
Destination directory	The directory on the remote nodes that the Enterprise Manager uses for storing files. This directory must be valid on all nodes of the cluster and should be a complete (absolute) path. Use of a relative path for the destination directory is not recommended, since the default directory for the remote process is not guaranteed to be the same each time.
Load users	If security is enabled, you must include user definitions as part of your configuration. (If security is not enabled, user definitions are optional.) Rather than define each user separately, the Enterprise Manager lets you define users in a VoltDB deployment file and then upload that deployment file into the configuration. This way you can reuse user definitions for multiple databases. To load users, click on the **Browse...**[1] button and select a valid VoltDB deployment file. The Enterprise Manager imports the user definitions from that file. (Note that only the user definitions are loaded; the other settings within the deployment file are ignored.)

Catalog Settings	
Field	**Description**
Upload catalog	You must specify the schema for your database by loading an existing VoltDB catalog. Click on the **Browse...**[1] button and select a catalog JAR file to load. The Enterprise Manager then opens and validates the contents of the catalog. Once you load a catalog, the fields above the Upload catalog field display a summary of information about the catalog.
Snapshot Settings	
Field	**Description**
Frequency	Snapshot frequency indicates how often (in seconds) an automated snapshot will be taken of the database. By default, the frequency is zero, or no automated snapshots. If you want periodic snapshots of the database, specify the frequency for the snapshots in this field as a whole number of seconds. (For example, 300 seconds for snapshots every 5 minutes.)
Snapshots retained	If you are using automated snapshots, the snapshots will use up more and more disk space over time. To avoid this situation, you must specify a maximum number of snapshots to keep on the database servers. When an automated snapshot is taken, if there are more snapshots than specified, the oldest snapshot is deleted.
Copy snapshots to management server	By selecting this checkbox, the automated snapshots are copied from the database servers to the management server. Snapshots *must* be copied to the management server if you want to use them later in the Enterprise Manager to restore a snapshot when you start the database. When snapshots are copied to the management server, they are not deleted from the database server(s). It is a good idea to both enable the copying of automated snapshots and set a small value for the number of snapshots retained, to avoid excessive disk usage. Note that the retention setting affects snapshots on the database servers only. It does not purge snapshots on the management server. It is a good idea to periodically review and delete old snapshots on the management server. (See Section 8.3, “Preventative Maintenance Using Snapshots” for more information on managing snapshots.)

[1]The exact wording of buttons and controls may vary from browser to browser. Use the appropriate interface widgets for your environment to complete the described actions.

Once you complete filling out the dialog box, click the **Create** button to add the database to the dashboard.

4.2. Changing the Database Configuration

After you add a database to the dashboard, you can still change the configuration settings, as necessary. Make sure the database is being displayed in the dashboard. (If not, click on the database name in the global list of databases to the left and select **View** from the popup menu.) Then click on **Edit** next to the settings you want to modify:

- The basic database settings you defined when creating database are displayed in the Configuration panel. Click on **Edit** in the Configuration panel to change these settings or to access the advanced settings.

- The database schema is defined by the runtime catalog listed in the Catalog panel. Click on **Edit** in the Catalog panel to load a new catalog.

- The settings for automated snapshots are defined in the Snapshots panel. Click on **Edit** in the Snapshots panel to change the settings for automated snapshots or to manage existing snapshots.

The catalog and snapshot settings match the settings available to you when you created the database. The configuration settings include both the basic settings you specified when creating the database, and advanced settings that let you customize the ports the database servers use at runtime, paths for runtime functions, and more. Table 4.2, “Advanced Configuration Settings” describes the additional fields on the Edit Database dialog box.

Table 4.2. Advanced Configuration Settings

Port Settings	
Field	**Description**
Client Port	The port that client applications use to connect to the VoltDB database cluster. The default client port is 21212. You can specify a different value. However, if you do, all client applications must specify that port number when creating a connection to the database.
Admin Port	The port used to enable and disable admin mode and issue commands while admin mode is in effect. The default admin port is 21211. See the section "Admin Mode" in *Using VoltDB* for more information about admin mode.
HTTP Port	The port that the JSON interface uses to connect to the VoltDB database cluster. This port also provides basic information about the database (including version number and memory usage) if accessed by an HTTP request not directed at the JSON URL.
Enable JSON	If your client applications use the JSON interface to access the database, you must check the box to enable JSON. The JSON interface uses the port identified in the **HTTP Port** field.
Internal Port	The port that VoltDB servers use to communicate among themselves.
Log Port	The port that the Enterprise Manager uses to collect Log4J log messages from the individual servers.
JMX Port	The port that the VoltDB Enterprise Manager uses to retrieve status and performance statistics from the individual servers (using JMX, the Java Management Extensions).
Zookeeper Port	The port that VoltDB uses to interact with Zookeeper locally. The default port is 2181.
Path Settings	
Field	**Description**
Snapshot Directory	The path where snapshots are stored on the individual database servers, including manual, automated, and network partition snapshots. This path is relative to the VoltDB root directory on the remote nodes. The VoltDB root directory for remote servers controlled by the Enterprise Manager is a subfolder of the destination directory. The name of the subfolder is the database ID. So, for example, if the destination directory is `/opt/voltdb` and the database ID is 12345, the VoltDB root directory is `/opt/voltdb/12345/`. If you specify an absolute path for the snapshot directory, the snapshot files are stored in that directory. If you specify a relative path, the files are stored in that path relative to the VoltDB root directory. If you do not specify a snapshot directory path, snapshots are stored in a subfolder named `snapshots` under the VoltDB root directory.

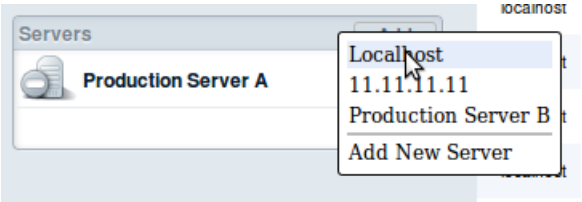
Path Settings	
Field	**Description**
Export Overflow Directory	The path where export overflow information is stored if the export queue backs up. This path is relative to the VoltDB root directory on the remote nodes, as outlined in the description of the snapshot directory above. If you specify an absolute path, export overflow information is stored in that directory. If you specify a relative path, the data is stored in that path relative to the VoltDB root directory, If you do not specify an export overflow path, export overflow data is stored in a subfolder named `export_overflow` under the VoltDB root directory.
Command Log Directory	The path where the command logs are stored if command logging is enabled. For maximum performance while logging, the command logs should be written to a disk with good response time. (For example, a disk with battery-backed caching.) It is also best to dedicate a disk to the command logs, rather than share a device for logs and other I/O such as snapshots or export overflow. If you specify an absolute path, the command logs are stored in that directory. If you specify a relative path, the data is stored in that path relative to the VoltDB root directory, By default the command logs are stored in a subfolder named `command_log` under the VoltDB root directory.
Command Log Snapshot Directory	The path where the snapshots associated with command logs are stored, if command logging is enabled. Again, for maximum performance, it is best to write the logs and the snapshots to separate devices. Note that these are snapshots specific to command logging. If you have automated snapshots turned on, those snapshots are written to the snapshot directory (described earlier) instead. If you specify an absolute path, the snapshots are stored in that directory. If you specify a relative path, the snapshots are stored in that path relative to the VoltDB root directory, By default the command log snapshots are stored in a subfolder named `command_log_snapshot` under the VoltDB root directory.
Feature Configuration	
Field	**Description**
Command Logging: Log Frequency	There are two settings available for configuring the frequency of command logging. These settings define the frequency with which the record of procedure invocations are written to the command log. The more frequently the logs are written, the less danger of losing data there is in case of a complete cluster failure. However, the more frequently logs are written, the more likely it is that disk I/O may impact your application's throughput or latency. There are two frequency settings that can be set separately: • Time (in milliseconds) • Number of transactions Whichever milestone is reached first initiates a write to the logs. See the chapter on "Command Logging and Recovery" in *Using VoltDB* for more information.
Command Logging: Synchronous Logging	In addition to specifying the frequency of logging, you can also specify whether logging occurs synchronously or asynchronously. Normally, logging is asynchronous to the database transactions; the transactions continue while the log is being written. If you select synchronous logging (by checking the checkbox), the

Feature Configuration	
Field	**Description**
	results of all transactions are held until the next batch of transactions are written to the log. Synchronous logging is recommended only if you have a fast, dedicated device for command logging (such as a disk with a battery-backed cache). Otherwise logging will have a negative impact on the performance of your transactions as they wait for the disk I/O to complete. At the same time, if you *do* use synchronous logging, it is important to set the command logging frequency to a very small increment (1-10 milliseconds), to keep the impact on latency to a minimum. See *Using VoltDB* for more information on command logging.
Command Logging: Log Size	The size of the command logs defines how much disk space is preallocated for storing the logs. For most workloads, the default log size of one gigabyte is sufficient. However, if your workload writes large volumes of data or uses large strings for queries (so the command invocations include large parameter values), the log segments fill up very quickly. To avoid this, you can increase the total log size. Specify the desired log size as an integer number of megabytes. The minimum log size is three megabytes.
Server Settings	
Field	**Description**
Max Java heap	The maximum heap size for the Java process running the database software on each cluster node.
Heartbeat Timeout	The database servers keep track of each other by periodically checking for a "heartbeat" from the other nodes in the cluster. If a heartbeat is not received from a server within a specified time limit, that server is assumed to be down and the cluster reconfigures itself with the remaining nodes (assuming it is running with K-safety). This time limit is called the "heartbeat timeout" , which is specified in seconds. For most situations, the default value for the timeout (10 seconds) is appropriate. However, if your cluster is operating in an environment that is susceptible to network fluctuations or unpredictable latency, you may want to increase the heartbeat timeout period.
Max temp table memory	The maximum size for the temporary tables that VoltDB uses to store table data while processing transactions. The default temp table size is 100 megabytes. This setting is appropriate for most applications. If you need to increase the temp table size, you can specify an alternate size as an integer number of megabytes. Note, however, increasing the temp table limit may result in out-of-memory errors on memory-constrained systems.
Snapshot Priority	Snapshot priority specifies the priority (as a value between 0 and 10) for snapshots as opposed to other database work. The lower the priority value, the more priority snapshot work receives. Zero specifies that no prioritizing is applied and snapshot work is done immediately. The closer to 10 the priority value, the longer a snapshot can take to complete. The closer to 1 the value, the quicker the snapshots complete but the more likely snapshots could impact latency and/or throughput of database transactions on a loaded database.

Note that certain settings cannot be changed while the database is running (for example, port numbers or the maximum heap size). If the database is running and a setting cannot be modified, that particular configuration field is grayed out in the interface. To change these settings, you must stop the database first.

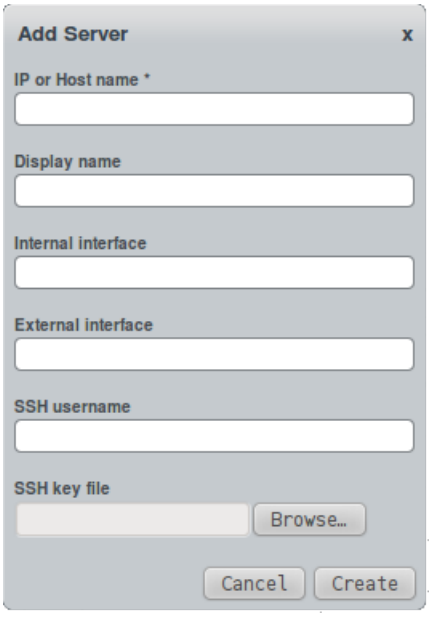
4.3. Adding Servers to Your Database Configuration

Once you create the database in the Enterprise Manager, you need to assign servers that will form the cluster that runs the database. Make sure the database is showing in the dashboard. (If not, click on your new database's name in the database list and select **View**.) You can then click on the **Add** button at the top of the server list in the dashboard.



If you defined servers earlier (when creating another database), the names of those servers appear on the Add Server list. Simply select the appropriate server name from the list to assign it to the current database.

If the server you want is not already defined, choose **Add New Server...** to add it to the list. The Add Server dialog box comes up to let you define the new server.



Enter the IP address or hostname of the server in the first field. This field is required. However, hostnames and IP addresses are not always very meaningful. So you can also enter a display name in the second field. If you provide a display name, this is the name the Enterprise Manager uses in lists and displays. If not, it displays the hostname or IP address.

If the server has multiple IP addresses, you can specify which interface to use for internal ports and which to use for external ports in the next two fields. See Section A.1.1, “Network Configuration (DNS)” for more information about specifying internal and external interfaces.

You can also provide SSH credentials for the server as part of the Add Server dialog. Normally, you will want to use the same SSH credentials for all servers, so you can accept the default for the SSH username and keyfile. However, if a server has unique credentials, you can enter them here.

Once you complete the dialog box, click **Create** to add the server to the list and assign it to the current database.

4.4. Configuring a Database Manually

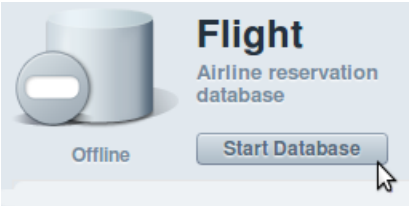
The Enterprise Manager simplifies the process of configuring and starting a database. But all of the settings available through the Enterprise Manager can also be set using the VoltDB Community Edition. The difference is, using the Community Edition you adjust the configuration through syntax in the deployment file or command line arguments when you start the database. See the Chapter 5 on "Running Your VoltDB Application" and the appendix on the deployment configuration file in *Using VoltDB* for details on setting configuration options using the command line interface.

Chapter 5. Starting and Stopping the Database

Once you add a database and servers to the Enterprise Manager, you are ready to get things rolling. This chapter explains how to start, stop, and pause a database using the Enterprise Manager.

5.1. Starting the Database

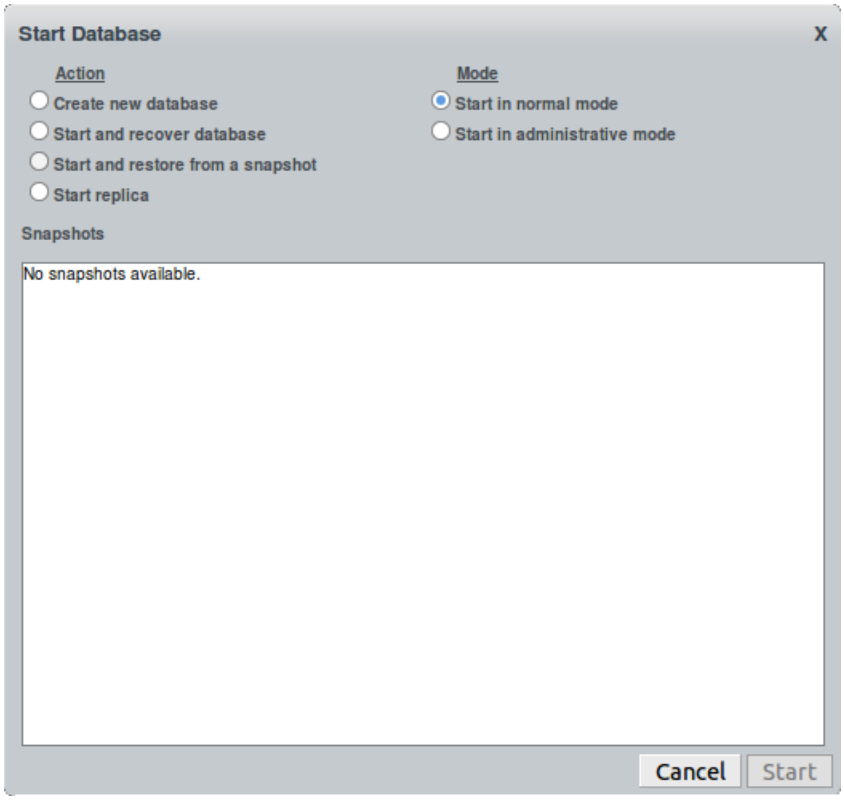
Once you configure the database and add servers, you are ready to start the cluster. Click on the **Start Database** button next to the database name in the dashboard to start the database.



If your current configuration is not complete (for example, if you specify a K-safety value of 1 but only have one server assigned to the database), the **Start Database** button is grayed out. Use the dashboard to correct the configuration before starting the database.

When you click on **Start Database**, the Enterprise Manager presents you with four possible actions:

- Create a new database.
- Start and recover the database from the command logs of a previous run.
- Start and restore a snapshot that was copied to the management server.
- Create a replica database.



If you are starting a database for the first time, you must select the first option (creating a new database) because there are no command logs to recover or snapshots to restore. If, during normal operations, you need to restart a database and want to restore a previous known state, you can choose either of the latter two options:

- **Start and recover** — starts the database and replays the command logs from the last session, reinstating the database contents to their last known state. To recover a database, the previous database session must have been run with command logging turned on. (If not, the operation fails and you must start using the `create new database` action.) See Section 4.1, “Configuring the Database” for information on enabling command logging.

- **Start and restore** — starts the database and restores the selected snapshot. Restoring a snapshot restores the contents of the database at a known point in time (when the snapshot was created). You can only restore snapshots that have been copied to the management server. Select the appropriate snapshot to restore from the list in the dialog box. See Section 8.3.1, “Collecting Snapshots” for more information on creating and copying snapshots.

The last option, starting a replica, is part of the database replication process. See the *Using VoltDB* manual for more information about database replication.

You can also select a startup mode: either normal or administrative mode. To start the database and allow client interactions as soon as the database startup is complete, choose *normal mode* (the default). If you want to perform administrative functions, or want to restore a snapshot manually (for example, if you are restoring a snapshot created outside of the Enterprise Manager), choose the second option, starting in admin mode.

Admin mode stops the database from accepting transactions on the usual client port; only requests issued over the admin port are accepted. Starting in admin mode prevents clients from accessing the database prematurely and lets you perform administrative tasks over the admin port before initiating client activity.

Once you select an startup option and click **Start**, the Enterprise Manager performs the following actions:

1. Generates the appropriate deployment file based on the database configuration.

2. Copies the necessary files (that is, the VoltDB software, the deployment file, and the runtime catalog) to each of the cluster servers. If you chose to restore a snapshot, the snapshot files are also copied to the cluster.

3. Starts the VoltDB database on all of the servers and performs the requested actions (such as recover or restore).

While the database is starting, the icon next to its name will "spin". Once the cluster is successfully started, the icon turns green. Or, if you chose to start in admin mode, blue to indicate that the database is paused. If there are any problems, the console will display the associated messages in the log message area in the lower right of the dashboard and the database icon returns to its stopped state.

You can also start a database that is not currently showing in the dashboard. Simply click on the name of the database in the list on the left and select **Start** from the popup context menu that appears.

Finally, if you choose to start the database in admin mode, be sure to "unpause" the database once you complete your administrative activities. Click on the name of the database in the global list and select **Resume** from the popup menu to exit admin mode and resume normal operations.

5.2. Stopping the Database

When the database is running, the button on the dashboard changes to **Stop Database**. You can perform an orderly shutdown of the cluster by clicking on the **Stop Database** button. When you stop the cluster, the Enterprise Manager first verifies that you indeed mean to stop the database. If you confirm the shutdown, the console stops each node in the cluster and the icons on the dashboard turn grey to indicate the database is no longer running.

You can also stop a database not currently displayed in the dashboard. Just as you can start a database from the list of databases, you can stop a running database by clicking on the database name and selecting **Stop** from the popup context menu.

5.2.1. Pausing the Database (Admin Mode)

Sometimes you don't want to completely stop the database, but do need to "pause" the database while performing administrative functions. For example, you might want to stop incoming transaction requests while you rejoin a server to a K-Safe cluster.

Click on the database name in the full list of databases on the left and select **Pause** from the popup menu to place the database in admin mode. While the database is paused, you can still perform all administrative actions through the Enterprise Manager, such as updating the catalog, taking snapshots, stopping and rejoining servers (when running with K-Safety), or shutting down the database.

When you are done with your administrative tasks, you can either resume the database (using the popup menu) to return to normal operation or shutdown the database as described in Section 5.2, “Stopping the Database”.

5.2.2. Starting and Stopping Individual Servers

If you start a cluster with K-Safety, it is possible for individual nodes in the cluster to stop — either due to failure or by intent (when you want to perform system maintenance, for example) — without stopping the

cluster as a whole. It is then possible to correct the problem with the node (such as replacing hardware) and then restart the server, bringing it back into the running cluster.

To stop an individual node manually, click on its name in the list of servers. Then click **Stop** on the popup menu that appears. This stops VoltDB on the node and the icon next to the server name turns gray. To restart the node and have it rejoin the cluster, simply select **Rejoin** or Live Rejoin from the popup menu. See Section 8.2.2, “Choosing a Rejoin Option (Live Rejoin)” for more information about the difference between live rejoin and regular, blocking rejoins.

5.3. Starting and Stopping a VoltDB Database Manually

If you are using the VoltDB Community Edition, you can still start a database cluster relatively easily. Using SSH it is possible to create shell scripts to automate the process of copying the application catalog to a predefined location and then starting the VoltDB server process on each node in the cluster. It is also possible to use NFS-mounted disks to distribute the VoltDB software and catalogs to the cluster from a single location. However, you should be careful not to use NFS-mounted disks for saving automated snapshots, because the individual nodes will encounter a conflict when trying to write their files to the same location.

It is also possible to start a VoltDB cluster in admin mode manually, specifying `<admin-mode adminstartup="true"/>` in the deployment file. However, to start from a snapshot manually, you must first start the cluster in admin mode and then issue a @SnapshotRestore system procedure invocation through the admin port.

To stop or pause a VoltDB cluster manually, use the system procedures @Shutdown, @Pause, and @Resume as described in *Using VoltDB*.

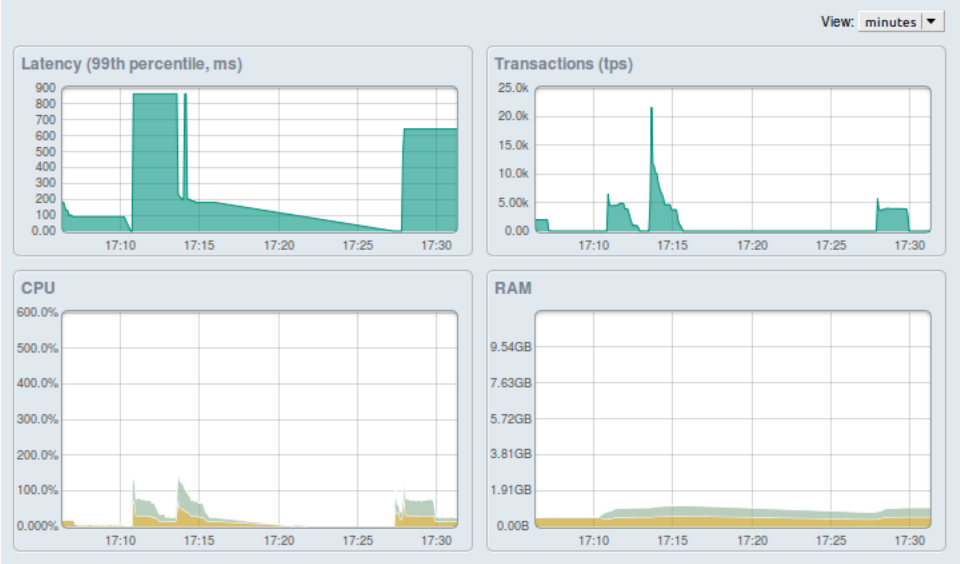
Chapter 6. Monitoring the Cluster

The goal of the VoltDB Enterprise Manager is not only to simplify basic administrative functions such as stopping and starting the database. The dashboard also helps you understand the performance characteristics of your application so you can identify issues and make informed decisions about configuration changes and tuning. Once a database is running, the Enterprise Manager dashboard helps you monitor:

- Database activity and performance
- Cluster health

6.1. Monitoring Database Activity

The right side of the dashboard provides real-time statistics on the currently selected database. There are four graphs showing you key aspects of database performance.



Latency

The latency graph shows you the latency for transactions being processed by the database. The graph shows latency for the 99th percentile of the transactions. (That is, 99% of the transactions complete within the time indicated.)

Latency measures the length of time (in milliseconds) between when the stored procedure request is received by the server and when the response is queued for return to the client. (Note that latency measurements do not include network latency between the client and the VoltDB server). Latency tends to go up when the servers receive more requests than they can process in a given amount of time.

Transactions

The transactions graph shows the number of transactions processed per second (TPS), a common measurement of database throughput. The goal for most OLTP applications is to maximize the number of TPS. There are a number of conditions that might make the transactions graph go down, including increased latency from too many multi-partition stored procedures, topping out system resources such as CPU and memory, or simply reduced load from the clients. Therefore, the transactions graph is always most informative when viewed in conjunction with the other graphs.

CPU — The CPU graph shows the percentage of the system's computational power being utilized by VoltDB on each server. As with any application, it is important to keep CPU usage below the total available CPU to ensure peak performance. (Each graph is cumulative for all cores on a processor. So, for example, if there are four cores, the total CPU available for that server is 400%.) The CPU graph provides an overall view of the amount of raw processing power that is being used by the VoltDB database process.

Memory — The memory graph shows the resident set size (RSS) of the VoltDB process on each server. Since VoltDB is an in-memory database, memory capacity is critical. The memory graph helps you understand current usage and trending of memory consumption.

Depending on your application, you may wish to see more or less data at a time. For example, during development you may be interested in short runs for testing, whereas for long-running applications, you will want to see extended graphs. You can change the horizontal scale of the graphs using the pulldown **View** menu on the top right. Set the view to "minutes" to see up to 30 minutes of data in the graph or to "day" to see a maximum of 24 hours displayed.

The graphs give you an overview of the database performance and resource utilization, which can be very helpful in detecting issues or performance regression in an application. The latency and transactions graphs show average statistics across the entire database cluster. The CPU and memory graphs show statistics per server, with multiple server statistics "stacked" in the graph. The list of servers on the left side of the dashboard include a color swatch to the right of each server name that acts as a legend to the color coding of the CPU and memory graphs.

However, the graphs alone are not necessarily sufficient for identifying the root cause of any issues you detect. To help you further diagnose problems or discrepancies, the dashboard provides detailed tabular statistics below the graphs. (The data charts are collapsed by default. Click on the label **Data** or the triangle next to it to expand the data tables.)

There are two types of tables provided: volume and invocation. Click on the headings in the data section to switch between the two types of data table.

- The *volume table* shows you the number of rows in each table, the type of table (partitioned, replicated, or view), and the maximum and minimum number of rows per partition. Large discrepancies between the minimum and maximum volume usually indicates a problem with the partitioning, either due to not enough partitions or a partitioning key value that is not well distributed. This could result in serious differences in memory usage between servers.
- The *invocation table* shows the total number of invocations for each stored procedure. In other words, the table shows you how often each stored procedure is called as well as the maximum, minimum, and average execution time in each case. This table is useful in determining if a particular stored procedure is taking longer than expected to execute or creating a bottleneck for the application. The invocation table is also useful in validating the expected distribution of transactions during normal operations.

6.2. Monitoring Overall Cluster Health

In addition to information about database activity, the Enterprise Manager also provides a quick view of the overall health of your database clusters. In the list of databases to the left of the dashboard, each database is represented by a colored icon, with different colors indicating the health of that entity. When the database is not running, the icon is gray. When the database is running properly, the icon is green.

If communication with a server fails while the database is running (for example, if the network fails or the VoltDB process stops on that node) the icon turns yellow or red, depending upon the consequences. If it is

a "K-safe" database (that is, the K-safety value allows for the remaining nodes of the cluster to continue), the database's icon turns yellow, indicating there is a problem but the database is still operational. If there are insufficient nodes to continue, the database stops and the icon turns red.

By clicking on the icon or name of the database in the list and choosing **View** from the popup menu, you can switch the dashboard to show that database and examine the situation more closely. In the dashboard, not only is the database icon color coded, but the servers in the server list are as well. So you can determine which server is in trouble.

Within the dashboard, if a server's icon is gray, it indicates that the server is stopped. After determining and fixing the problem, you can choose **Live ReJoin** or **ReJoin** from the server's popup menu to have the server restart and rejoin the cluster. If the problem is hardware related, you can choose **Replace** to replace the current server with another server from the Enterprise Manager's list of servers.

Once the problems are resolved and the database cluster is back to its full complement of nodes, the database icon will turn green again. See Chapter 8, *Maintaining and Repairing the Cluster* for more information on handling error conditions and performing maintenance activities on a running VoltDB database using the Enterprise Manager.

6.3. Integrating VoltDB with Other Monitoring Systems

The VoltDB Enterprise Manager provides a complete environment for managing and monitoring VoltDB databases. However, you may have an existing management framework you would like to use for monitoring all of your resources, including VoltDB. For these situations, the Enterprise Manager provides two alternatives for integrating VoltDB statistics with your larger monitoring needs.

6.3.1. Integrating with Ganglia

If you use Ganglia as your monitoring tool, VoltDB integrates seamlessly with Ganglia to provide VoltDB performance data to the Ganglia monitoring interface. Ganglia is a distributed monitoring system that provides a graphical interface to distributed clusters of systems. If Ganglia is present, VoltDB acts as a data source for the Ganglia system.

To use the VoltDB Enterprise Manager with Ganglia, make sure:

- The Ganglia Monitoring Daemon (gmond) is installed and configured on each VoltDB cluster node.
- The Ganglia Meta Daemon (gmetad) is installed and configured on the server running the VoltDB Enterprise Manager.

Having completed these steps, the VoltDB Enterprise Manager will automatically generate data for the Ganglia monitoring system. See the Ganglia web site (`http://ganglia.sourceforge.net/`) for more information about Ganglia.

6.3.2. Integrating Through JMX

The Enterprise Manager uses the Java Management Extensions (JMX) to send statistics from the database nodes to the management server. The VoltDB JMX interface is also available to other monitoring frameworks that want to query for VoltDB statistics.

VoltDB Enterprise Edition servers open the JMX interface on port 9090 by default (although you can change this port as part of the database configuration, described in Section 4.2, “Changing the Database

Configuration"). Clients can either poll on this port for specific information or subscribe to messages that are sent approximately every second.

The information sent over the JMX interface is the same as that available through existing VoltDB system procedures, such as @SystemInformation, @Statitstics, and @SnapshotStatus. The difference is that the JMX interface is a lightweight process and is not transactional, as system procedures are. Therefore the JMX interface produces less load on the servers than repeated calls to the system procedures would.

The easiest way to become familiar with the JMX interface for the Enterprise Manager is to connect to a running database using the Java Monitoring and Management Console (also known as Jconsole) and browse through the structures returned by the VoltDB servers.

6.4. Monitoring a Cluster Manually

If you are using the VoltDB Community Edition, you cannot use the Enterprise Manager to monitor your cluster. However, you can still get statistics on database activity and the performance of your cluster nodes.

The VoltDB system procedures provide valuable information about the status of the database. In particular, the @SystemInformation and @Statistics system procedures return information about the cluster configuration and database activity, respectively. (See the Appendix on system procedures in the *Using VoltDB* manual for details.)

Even without the Enterprise Manager, it is possible to use Ganglia for a consolidated view of the performance characteristics of your VoltDB database nodes. Ganglia can report on CPU usage, memory, disk storage, and other system statistics. See the Ganglia website (`http://ganglia.sourceforge.net/`) for more information.

Chapter 7. Updating the Database

It is not uncommon, after running a database application in development or production, to realize that there are improvements that could be made to the database schema, the stored procedures, the configuration, and so on. This sort of incremental improvement is a natural part of database development.

VoltDB lets you make many incremental improvements "on the fly" — without having to shutdown or restart the database. The Enterprise Manager makes this process even easier by providing a simple interface for changing the software and hardware configuration, and for loading and deploying updates to the runtime catalog. On the dashboard, click on the **Edit** button on the appropriate panel to modify:

- The database configuration
- The application catalog
- The snapshot settings

7.1. Updating the Database Configuration

To modify the database configuration, simply click **Edit** on the configuration panel and make the appropriate changes in the Edit Database dialog box. Some attributes (such as name and description) are modifiable at any time. Others require that the database be stopped before they can be changed.

If an attribute is not modifiable because the database is currently running or paused (for example, K-Safety or any of the port numbers), that field will be grayed out. To change these attributes, you must stop the database first and then make the changes before restarting.

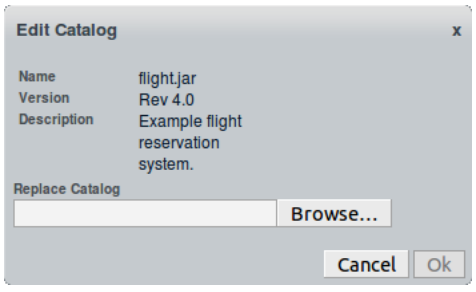
7.2. Updating the Application Catalog

Many changes to the database structure — including adding and removing columns or tables and modifying stored procedures — can be made while the database is running. You do this by updating the application catalog.

Once you rebuild the catalog, the Enterprise Manager not only helps you do the update, it lets you compare the two catalogs (the current and the new catalog) to see what changes they contain before finalizing the change. The following sections explain how to update the catalog using the Enterprise Manager dashboard.

7.2.1. Loading a New Catalog

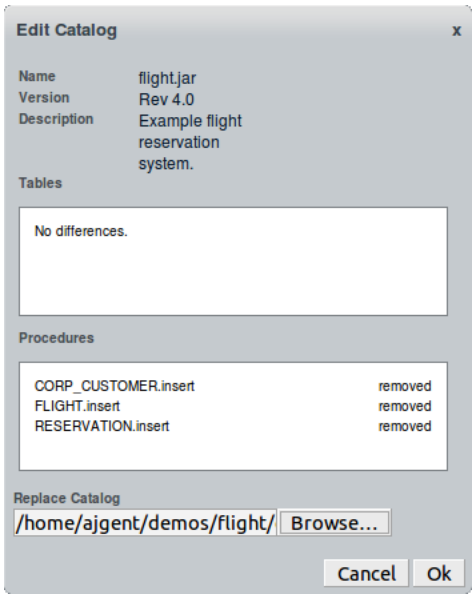
Click on **Edit** in the catalog settings panel to bring up the Edit Catalog dialog box.



When you first bring up the dialog box, it shows you information about the current catalog. To update to a new catalog, click on the **Browse...** button, select a new catalog from your system, and then click **OK**.

7.2.2. Comparing Differences Between the Catalogs

Once the Enterprise manager loads the new catalog, the Edit Catalog dialog changes to show you the changes between the current catalog and the new catalog. In particular, the dialog shows you any differences (whether additions, deletions, or modifications) in the schema tables or stored procedures.



When you update your project catalog, you can keep track of the changes using metadata that is shown in the dashboard. The name, version, and description of the catalog shown in the dashboard are taken from the <info> tag in your project definition file itself. For example, the following project definition file specifies the name, description, and version number for the current revision of the project:

```
<?xml version="1.0"?>
<project>
   <info>
      <name>Flight Reservation System</name>
      <version>Rev 1.3b</version>
      <description>
         Database definition for the flight reservation system.
      </description>
   </info>
         .
         .
         .
```

7.2.3. Confirming the Update

Once you compare the changes between the new catalog and the current catalog and are satisfied that they are changes you want to make, click the **OK** button one more time to commit the update.

If the database is running and the changes in the catalog are consistent with updating the database "on the fly", the Enterprise Manager will update the catalog information on the dashboard and update the database as well. If the catalog contains changes that require a restart, you will receive a warning that the catalog cannot be updated while the database is running. In this case, you will need to cancel the update, stop

the database (taking the appropriate snapshot if you wish to save the data beforehand), then update the catalog, and restart the database.

If the database is stopped, any catalog changes are possible. The catalog is updated as soon as you click **OK**.

7.3. Updating Snapshot Settings

There are three snapshot settings that you can modify: the frequency of snapshots, how many are retained on the database servers, and whether to copy new snapshots to the management server. When the database is stopped, you can change any of these settings. If the database is running, you can only change whether the snapshots are copied to the management server or not.

To modify the snapshot settings, click **Edit** on the Snapshots panel. The changes you make on the Edit Snapshots dialog go into effect as soon as you click **OK**. So, if the database is running, changes to the copying of snapshots to the management server are communicated the the cluster nodes immediately. If the database is not currently running, all changes go into effect the next time the database starts.

7.4. Updating the Database Manually

If you are using the VoltDB Community Edition, you can update the application catalog manually using the @UpdateApplicationCatalog system procedure. For other changes, you may need to shutdown the database, distribute new versions of the catalog and deployment file, and restart the cluster manually. See the chapter on "Updating Your VoltDB Application" in the *Using VoltDB* manual for details.

Chapter 8. Maintaining and Repairing the Cluster

The VoltDB Enterprise Manager helps you monitor and maintain your database by displaying statistics about the state of the database. In addition to the graphs described in Chapter 6, *Monitoring the Cluster*, the dashboard provides access to log messages of any errors or warnings reported by the individual servers or the cluster as a whole.

This chapter explains how to:

- Detect and evaluate error conditions
- Remove and rejoin individual nodes for repair
- Perform regular maintenance tasks for the database using snapshots

8.1. Detecting and Evaluating Error Conditions

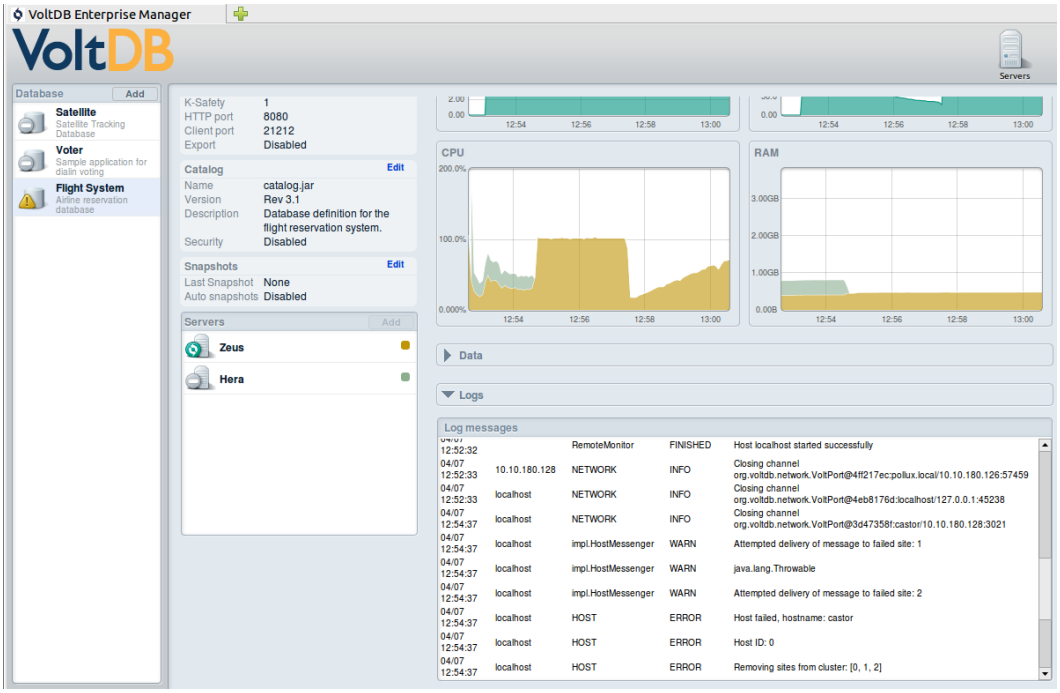
Performance issues can be diagnosed using the graphs and data tables that the dashboard displays. However, if an error occurs, it may not be easy to identify by the performance graphs alone. Instead, you want to see the logs of any errors or warnings that the database generates.

At a high level, the dashboard provides a visual indicator of serious problems through the icons next to the database and server names. If an icon changes from green to yellow or red, you know there is a problem. For example, if a server fails on a cluster running with K-safety, the server icon turns red and the cluster icon changes to yellow, showing that the cluster is still running, but not with its full complement of nodes.

The Enterprise Manager helps you find out exactly what happened by displaying the console logs from all the nodes in the cluster.

- Click on the name of the database on the list of databases and choose **View** from the popup menu to display that database in the dashboard.
- If the log message area in the lower right of the dashboard are collapsed, Click on the Logs heading or the triangle next to it to expand the display and show the log messages.

In the following example, the server Hera suffered a hardware failure and stopped. Since the VoltDB process stopped abruptly, there are no messages from the node itself. But if you look at the log messages, you see an error indicating that the other node sees the failure and identifies the troubled node. At the same time, you can see the icon for Hera has turned gray, indicating it has stopped.



8.2. Rejoining and Replacing Servers

Once you determine what happened, you often need to take action to correct the problem. If, for example, the VoltDB server process stops on a node because of a hardware failure, you will need to fix the hardware before bringing the node back into the cluster.

8.2.1. Rejoining a Node to the Database Cluster

Once the problem is diagnosed and any necessary repairs made, you can bring the node back into the running cluster directly from the dashboard. Click on the name of the stopped node in the server list and select one of the two **Rejoin** options from the popup menu. The Enterprise Manager copies the necessary files to the node, issues a rejoin request, and rejoins the node to the cluster. Once this procedure is complete, the indicators for the node and the cluster return to green.

Note that if two or more nodes are removed from the running cluster, you should have the nodes rejoin the cluster one at a time. Wait for each node to complete the rejoin process before starting the next node. Attempting to rejoin multiple nodes at the same time can result in some nodes timing out and failing to rejoin.

8.2.2. Choosing a Rejoin Option (Live Rejoin)

When rejoining a node to the cluster, you have two choices: regular rejoin and *live* rejoin. Which type of rejoin you choose depends on the specific needs of your application. To rejoin a busy database without negatively impacting the throughput or latency of client applications, use **Live Rejoin**. If your database can be paused temporarily, you can use a regular **Rejoin**.

Normally, when rejoining, the other nodes in the cluster send copies of the appropriate partitions to the rejoining node as part of their transactional work queue. This provides the quickest way to restore a rejoining node.

However, the downside is that the copying process ties up that partition until the copy is complete. No other transactions are executed by the partition in the meantime. If there is a multi-partition transaction in the queue, the rejoin can block all partitions as they wait for that partition to complete its part of the multi-partition transaction.

To avoid blocking transactions on an active database, you can request a "live" rejoin. A live rejoin is performed as a separate workload and does not block the existing partitions. Live rejoin is preferable because the database remains available throughout the procedure with a minimum impact on throughput and performance. The deficit of a live rejoin is that, for large datasets, the live process can take longer to complete than with a blocking rejoin.

In rare cases, if the database is near capacity in terms of throughput, a live rejoin cannot keep up with the ongoing changes made to the data. If this happens, VoltDB reports that the live rejoin cannot complete and you must wait until database activity subsides or you can safely perform a regular, blocking, rejoin to reconnect the server.

8.2.3. Replacing a Node in the Database Cluster

If the problem is hardware related and cannot be fixed quickly, you may wish to replace that server in the cluster. Similarly, if you want to deliberately remove a specific server from the cluster (for maintenance, for example), you can click on its name and select **Stop** from the popup menu.

Once the server is stopped, you can tell the Enterprise Manager to replace it with another server by doing the following:

1. Click on the server name in the list of servers on the dashboard.
2. Select **Replace** from the popup menu.
3. A secondary menu appears to the right, listing all of the available servers. Click on the name of the server you want to use as a replacement, or click on **Add...** to enter a new server.
4. If you choose **Add...** the Add Server dialog box pops up and lets you enter the new server information. Click **Replace** to add the server and use it as the replacement.
5. At this point the menus disappear and the stopped node is replaced in the server list with the name of its replacement.
6. Once the replacement is complete, you can click on the new server's name and select **Rejoin** or **Live Rejoin** to have it join the cluster and restore the database to its full complement of servers.

8.2.4. Restarting the Cluster

Note that the process for rejoining and replacing nodes to the cluster described above depends on a high availability cluster. That is, a cluster with the K-safety value set to one or more. If the cluster is running without K-safety, or more nodes fail than the cluster can support, the entire cluster will stop and the database icon will turn red.

In this situation, you can still restart the database from within the Enterprise Manager. Simply click on the **Start Database** button next to the database name or select **Start** from the popup context menu in the global list of databases.

However, when the entire database stops, the data in memory is lost. If you restart the database using the **create** option, you create a fresh, empty database. If you want to restore the database's previous state, be sure to use the **recover** option when restarting the database. This is why use of command logging or automated snapshots is recommended, especially if you are not using K-safety for durability.

8.3. Preventative Maintenance Using Snapshots

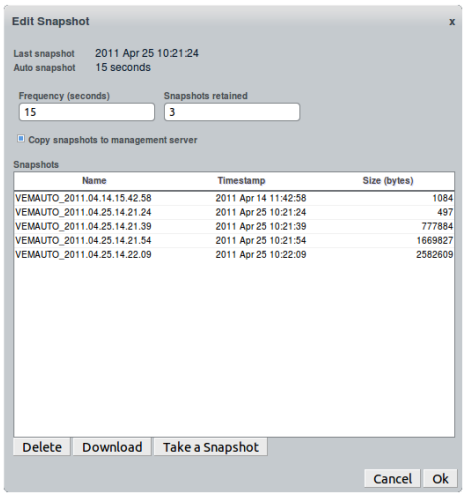
K-safety is one way to ensure the durability of the database by protecting the cluster against individual node failures. Another important feature for ensuring the viability of the cluster is the use of snapshots to save and restore the data from disk.

If you have automatic snapshotting turned on, the Enterprise Manager helps you manage your snapshots by optionally collecting the created snapshots and copying them to the management server. These features are available from the **Edit** button on the Snapshot Settings panel of the dashboard.

The following sections explain how to use the Enterprise Manager to restore a database and manage your database snapshots.

8.3.1. Collecting Snapshots

When you create the database, you get to choose whether to create automated snapshots, and whether those snapshots are copied to the management server. Once the database exists, you can revisit these settings by clicking on the **Edit** button in the Snapshot Settings panel of the dashboard.



The Edit Snapshots dialog box lets you change the settings for automated snapshots. You can modify the frequency of automated snapshots, how many snapshots are retained, and whether automated snapshots are copied to the management server. Once you make the desired changes, click **OK** to have the changes take effect or click **Cancel** to discard the changes and close the dialog box.

The Edit Snapshots dialog also lets you take a manual snapshot. Whether automated snapshots are enabled or not or whether *copy to management server* is enabled or not, a manual snapshot is always copied to the management server and made available to you. When you click on the **Take a Snapshot** button, the manual snapshot is initiated. After the snapshot is complete, the Enterprise Manager will copy the snapshot to the management server and display it in the list of available snapshots.

8.3.2. Restoring the Database from a Snapshot

Collecting the snapshot files provides assurance against failure. If the database fails for any reason, you can use the snapshots to restore the database's contents to a known state. You can also use the snapshots to save and restore the database in cases where you know you must take the cluster down for maintenance.

If you know that the database cluster must shutdown, you can use the following procedure to capture a known state of the database.

1. Pause the database, putting it in admin mode and stopping further client activity, by clicking on the database name in the global list of databases and selecting **Pause** from the popup menu.

2. If automated snapshots are not currently being copied to the management server, click on **Edit** in the Snapshot Settings panel to bring up the Edit Snapshots dialog box. Check the box for copying snapshots, then click **OK**. Or, if automated snapshots are disabled, use the Edit Snapshot Settings dialog to take a manual snapshot.

3. Wait for the snapshot to complete and be copied to the management server. At this point all client activity should have stopped and you can be assured that this last snapshot is complete.

4. Shutdown the database by clicking the **Stop Database** button.

At this point the cluster can be shutdown, repaired, replaced, or other maintenance completed. Once the servers are ready to start again, use the following procedure to restore the last valid snapshot of the database.

1. In the Enterprise Manager dashboard, click on the **Start Database** button.

2. In the Start Database dialog box that appears, select the option for "Start and restore from a snapshot".

3. Next, select the snapshot you want to restore. (The snapshots are listed with the most recent first.)

4. Click **Start**.

When you follow these steps, the Enterprise Manager will start the database and restore the selected snapshot to return the database contents to their previous state. Note that the snapshots must be copied to the management server before they become available for use in restoring the database on start.

8.3.3. Managing Snapshots

Over time, the number of snapshots that the Enterprise Manager collects can grow. You should periodically clean up the list be removing old and outdated snapshots.

- For snapshots you no longer need, you can prune the list by selecting one or more snapshots from the list and clicking the **Delete** button. The selected snapshots are deleted from the management server and removed from the list.

- For snapshots you want to save but not necessarily reuse in the Enterprise Manager, you can select those snapshots and click on **Download**. The selected snapshots are zipped into a single compressed file each and downloaded to your current system. (That is, the computer whose web browser you are using to access the management server.)

While cleaning up, be sure to keep any snapshots you may want to use later to restore the database. The Enterprise Manager can only restore snapshots that are on the management server.

8.4. Maintaining and Repairing Your Cluster Manually

If you are using the VoltDB Community Edition, you can maintain, repair, and recover your database cluster manually using the VoltDB system procedures and command line options. If a node fails when a cluster is running with K-safety, it is possible to have the node rejoin the cluster using the `rejoin` action

on the command line. See the section on "Recovering from System Failures" in the *Using VoltDB* manual for information on using `rejoin`.

You can also initiate manual snapshots using the @SnapshotSave system procedure. See the section on "Performing a Manual Save and Restore of a VoltDB Cluster" in the *Using VoltDB* manual for details.

Appendix A. Server Configuration Options

There are a number of system, process, and application options that can impact the performance or behavior of your VoltDB database. You can control some of these options when starting VoltDB. Some of the most important, such as sites per host and K-safety, are described in detail in the *Using VoltDB* manual. Other aspects of the server and process configuration can also impact database performance and reliability.

The various configuration options fall into five main categories:

- Server configuration
- Process configuration
- Database configuration
- Path configuration
- Network ports used by the database cluster

This appendix describes each of the configuration options, how to set them, and their impact on the resulting VoltDB database and application environment.

A.1. Server Configuration Options

VoltDB provides mechanisms for setting a number of options. However, it also relies on the base operating system and network infrastructure for many of its core functions. There are operating system configuration options that you can adjust to to maximize your performance and reliability, including:

- Network configuration
- Time configuration

A.1.1. Network Configuration (DNS)

VoltDB creates a network mesh among the database cluster nodes. To do that, all nodes must be able to resolve the IP address and hostnames of the other server nodes. Make sure all nodes of the cluster have valid DNS entries or entries in the local hosts files.

For servers that have two or more network interfaces — and consequently two or more IP addresses — it is possible to assign different functions to each interface. VoltDB defines two sets of ports:

- External ports, including the client and admin ports. These are the ports used by external applications to connect to and communicate with the database.
- Internal ports, including all other ports. These are the ports used by the database nodes to communicate among themselves. These include the internal port, the zookeeper port, and so on. (See Section A.5, “Network Ports”for a complete listing of ports.)

You can specify which network interface the server expects to use for each set of ports by specifying the internal and external interface when starting the database. When using the VoltDB community edition, use the internalinterface and externalinterface arguments on the command line. For example:

```
$ voltdb create deployment deployment.xml \
  catalog catalog.jar host serverA \
  license license.xml \
  externalinterface 10.11.169.10 \
  internalinterface 10.12.171.14
```

When using the REST API the internal and external interfaces are defined as part of the server resource.

A.1.2. Time Configuration (NTP)

Keeping VoltDB cluster nodes in close synchronization is critical to the performance of your database. At a minimum, use of NTP to synchronize time across the cluster to a single NTP server is recommended. If the time difference between nodes is too large (greater than three seconds) VoltDB refuses to start. It is also important to avoid having nodes adjust time backwards, or VoltDB will pause while it waits for time to "catch up" to its previous setting.

A.2. Process Configuration Options

In addition to system settings, there are configuration options pertaining to the VoltDB server process itself that can impact performance. Runtime configuration options are set as command line options when starting the VoltDB server process.

The key process configuration for VoltDB is the Java maximum heap size. It is also possible to pass other arguments to the Java Virtual Machine directly.

A.2.1. Maximum Heap Size

The heap size is a parameter associated with the Java runtime environment. Certain portions of the VoltDB server software use the Java heap. In particular, the part of the server that receives and responds to stored procedure requests uses the Java heap.

Depending upon how many transactions your application executes a second, you may need additional heap space. The higher the throughput, the larger the maximum heap needed to avoid running out of memory.

In general, a maximum heap size of a gigabyte (1024) is recommended. More than 2 gigabytes tends to use up space that is needed for data storage without providing any corresponding benefit. Less than 1 gigabyte and there is a risk that Java will run out of memory for throughput-intense applications.

It is important to remember that the heap size is not directly related to data storage capacity. Increasing the maximum heap size does not provide additional data storage space. In fact, quite the opposite. Needlessly increasing the maximum heap size reduces the amount of memory available for storage.

To set the maximum heap size when starting VoltDB, define the environment variable VOLTDB_HEAPSIZE as an integer value (in megabytes) before issuing the `voltdb` command. For example, the following commands start VoltDB with the recommended 1 gigabyte limit:

```
$ VOLTDB_HEAPSIZE="1024"
$ voltdb create host localhost \
         catalog mycatalog.jar deployment deployment.xml
```

When using the Enterprise Manager, there is an input field on the database configuration dialog for specifying the maximum heap size. The Enterprise Manager uses this value to set both the maximum and

the minimum heap size to avoid any issues at runtime should the heap need to grow and additional memory is no longer available.

Note that the Enterprise Manager sets the maximum heap size to the recommended 1024 value by default, as does the `voltdb` shell command. If you are using the java command line to start VoltDB manually, the default for the maximum heap size is much lower, so use of the java -Xmx argument is strongly recommended.

A.2.2. Other Java Runtime Options (VOLTDB_OPTS)

VoltDB sets the Java options — such as heap size and classpath — that directly impact VoltDB. There are a number of other configuration options available in the Java Virtual machine (JVM).

VoltDB provides a mechanism for passing arbitrary options directly to the JVM. If the environment variable VOLTDB_OPTS is defined, its value is passed as arguments to the Java command line.

- When starting VoltDB using the voltdb command, the contents of VOLTDB_OPTS are added to the Java command line on the current server.
- When starting the VoltDB Enterprise Manager, the contents of VOLTDB_OPTS are added to the invocation that the Enterprise Manager uses to start the VoltDB process on all remote servers.

In other words, when starting a cluster manually, you must define VOLTDB_OPTS on each server to have it take effect for all servers. When using the Enterprise Manager, you only need to define VOLTDB_OPTS once, before starting the Enterprise Manager, to have these JVM options take effect on all nodes that the Enterprise Manager starts. For example:

```
$ cd opt/voltdb/management/
$ VOLTDB_OPTS="-DmyApp.DebugFlag=true"
$ ./enterprise_manager.sh
```

Warning

VoltDB does not validate the correctness of the arguments you specify using VOLTDB_OPTS or their appropriateness for use with VoltDB. This feature is intended for experienced users only and should be used with extreme caution.

A.3. Database Configuration Options

Runtime configuration options are set either as part of the deployment configuration file or as command line options when starting the VoltDB server process. These database configuration options are only summarized here. See the *Using VoltDB* manual for a more detailed explanation. The configuration options include:

- Sites per host
- K-Safety
- Network partition detection
- Automated snapshots
- Export

- Command logging
- Heartbeat
- Temp table size

A.3.1. Sites per Host

Sites per host specifies the number of unique VoltDB "sites" that are created on each physical database server. The section on "Determining How Many Partitions to Use" in the *Using VoltDB* manual explains how to choose a value for sites per host.

You set the value of sites per host using the `sitesperhost` attribute of the `<cluster>` tag in the deployment file. When using the Enterprise Manager, the number of sites per host is set in the dialog box when you create or edit the database.

A.3.2. K-Safety

K-safety defines the level of availability or durability that the database can sustain, by replicating individual partitions to multiple servers. K-safety is described in detail in the "Availability" chapter of the *Using VoltDB* manual.

You specify the level of K-safety that you want in the deployment file using the `kfactor` attribute of the `<cluster>` tag. If you are using the Enterprise Manager, there is an input field ("kfactor") for specifying the K-safety value when you create or edit the database.

A.3.3. Network Partition Detection

Network partition detection protects a VoltDB cluster in environments where the network is susceptible to partial or intermittent failure among the server nodes. Partition detection is described in detail in the "Availability" chapter of the *Using VoltDB* manual.

You can enable or disable network partition detection in the deployment file using the <partition-detection> tag. If you are using the Enterprise Manager, there is a check box for enabling partition detection when you create or edit the database.

A.3.4. Automated Snapshots

Automated snapshots provide ongoing protection against possible database failure (due to hardware or software issues) by taking periodic snapshots of the database's contents. Automated snapshots are described in detail in the section on "Scheduling Automated Snapshots" in the *Using VoltDB* manual.

You enable and configure automated snapshots with the <snapshot> tag in the deployment file. If you are using the Enterprise Manager, you can enable automated snapshots when you create the database (using fields on the Create Database dialog box) or after the database is created from the Snapshot settings panel using the Edit Snapshots dialog.

Snapshot activity involves both processing and disk I/O and so may have a noticeable impact on performance (in terms of throughput and/or latency) on a very busy database. You can control the priority of snapshots activity using the <snapshot/> tag within the <systemsettings> element of the deployment file. The snapshot priority is an integer value between 0 and 10, with 0 being the highest priority and 10 being the lowest. The closer to 10, the longer snapshots take to complete, but the less they can affect ongoing database work.

Note that snapshot priority affects all snapshot activity, including automated snapshots, manual snapshots, and command logging snapshots. If you are using the Enterprise Manager, there is an input field for specifying the snapshot priority on the Edit Configuration dialog box.

A.3.5. Export

The export function lets you automatically export selected data from your VoltDB database to an export client using SQL INSERT statements to special export tables at runtime. This feature is described in detail in the chapter on "Exporting Live Data" in the *Using VoltDB* manual.

You enable and disable export using the <export> tag in the deployment file. If you are using the Enterprise Manager, there is a check box for enabling export when you create or edit the database.

A.3.6. Command Logging

The command logging function saves a record of each transaction as it is initiated. These logs can then be "replayed" to recreate the database's last known state in case of intentional or accidental shutdown. This feature is described in detail in the chapter on "Command Logging and Recovery" in the *Using VoltDB* manual.

To enable and disable command logging, use the <commandlog> tag in the deployment file. If you are using the Enterprise Manager, there is a check box for enabling command logging when you create the database and additional fields for configuring command logging on the Edit Database dialog box.

A.3.7. Heartbeat

The database servers use a "heartbeat" to verify the presence of other nodes in the cluster. If a heartbeat is not received within a specified time limit, that server is assumed to be down and the cluster reconfigures itself with the remaining nodes (assuming it is running with K-safety). This time limit is called the "heartbeat timeout" and is specified as a integer number of seconds.

For most situations, the default value for the timeout (10 seconds) is appropriate. However, if your cluster is operating in an environment that is susceptible to network fluctuations or unpredictable latency, you may want to increase the heartbeat timeout period.

You can set an alternate heartbeat timeout using the `<heartbeat>` tag in the deployment file. If you are using the Enterprise Manager, there is a input field for specifying the timeout on the Edit Database dialog box.

A.3.8. Temp Table Size

VoltDB uses temporary tables to store intermediate table data while processing transactions. The default temp table size is 100 megabytes. This setting is appropriate for most applications. However, extremely complex queries or many updates to large records could cause the temporary space to exceed the maximum size, resulting in the transaction failing with an error.

In these unusual cases, you may need to increase the temp table size. You can specify a different size for the temp tables using the `<temptables>` tag in the deployment file. Note: since the temp tables are allocated as needed, increasing the maximum size can result in a Java out-of-memory error at runtime if the system is memory-constrained. Modifying the temp table size should be done with caution.

If you are using the Enterprise Manager, there is a input field for specifying the maximum temp table limit on the Edit Database dialog box.

A.4. Path Configuration Options

The running database uses a number of disk locations to store information associated with runtime features, such as export, network partition detection, and snapshots. You can control which paths are used for these disk-based activities. The path configuration options include:

- VoltDB root
- Snapshots path
- Export overflow path
- Command log path
- Command log snapshots path

A.4.1. VoltDB Root

VoltDB defines a root directory for any disk-based activity which is required at runtime. This directory also serves as a root for all other path definitions that take the default or use a relative path specification.

By default, the VoltDB root is the directory `voltdbroot` created as a subfolder in the current working directory for the process that starts the VoltDB server process. (If the subfolder does not exist, VoltDB creates it on startup.) You can specify an alternate root in the deployment file using the `<voltdbroot>` element. However, if you explicitly name the root directory in the deployment file, the directory must exist or the database server cannot start. See the section on "Configuring Paths for Runtime Features" in the *Using VoltDB* manual for details.

When using the VoltDB Enterprise Manager, the VoltDB root directory is defined implicitly by the destination directory. You define the destination directory when you create the database. (See Section 4.1, “Configuring the Database” for details.) The VoltDB root becomes a subfolder of the destination directory, where the subfolder name is the same as the database ID of the current database. So, for example, if the destination directory is /`opt/voltdb` and the database ID is 12345, the resulting VoltDB root directory is `/opt/voltdb/12345`.

A.4.2. Snapshots Path

The snapshots path specifies where automated and network partition snapshots are stored. The default snapshots path is the "`snapshots`" subfolder of the VoltDB root directory. You can specify an alternate path for snapshots using the `<snapshots>` child element of the `<paths>` tag in the deployment file. When using the VoltDB Enterprise Manager, you specify an alternate snapshots path in the **Edit Database** dialog box (as described in Section 4.2, “Changing the Database Configuration”).

A.4.3. Export Overflow Path

The export overflow path specifies where overflow data is stored if the export queue gets too large. The default export overflow path is the "`export_overflow`" subfolder of the VoltDB root directory. You can specify an alternate path using the `<exportoverflow>` child element of the `<paths>` tag in the deployment file. When using the VoltDB Enterprise Manager, you specify an alternate path in the **Edit Database** dialog box (as described in Section 4.2, “Changing the Database Configuration”).

See the chapter on "Exporting Live Data" in the *Using VoltDB* manual for more information on export overflow.

A.4.4. Command Log Path

The command log path specifies where the command logs are stored when command logging is enabled. The default command log path is the "`command_log`" subfolder of the VoltDB root directory. However, for production use, it is strongly recommended that the command logs be written to a dedicated device, not the same device used for snapshotting or export overflow. You can specify an alternate path using the `<commandlog>` child element of the `<paths>` tag in the deployment file. When using the VoltDB Enterprise Manager, you specify an alternate path in the **Edit Database** dialog box (as described in Section 4.2, “Changing the Database Configuration”).

See the chapter on "Command Logging and Recovery" in the *Using VoltDB* manual for more information on command logging.

A.4.5. Command Log Snapshots Path

The command log snapshots path specifies where the snapshots created by command logging are stored. The default path is the "`command_log_snapshot`" subfolder of the VoltDB root directory. (Note that command log snapshots are stored separately from automated snapshots.) You can specify an alternate path using the `<commandlogsnapshot>` child element of the `<paths>` tag in the deployment file. When using the VoltDB Enterprise Manager, you specify an alternate path in the **Edit Database** dialog box (as described in Section 4.2, “Changing the Database Configuration”).

See the chapter on "Command Logging and Recovery" in the *Using VoltDB* manual for more information on command logging.

A.5. Network Ports

A VoltDB cluster opens network ports to manage its own operation and to provide services to client applications. When using the Enterprise Manager, most ports are configurable as part of the database definition. Many of the network ports are configurable as part of the command line that starts the VoltDB database process or through the deployment file. Table A.1, “VoltDB Port Usage” summarizes the ports that VoltDB uses, their default value, and how to change the default. The following sections describe each port in more detail.

Table A.1. VoltDB Port Usage

Port	Default Value	Where to Set (Community Edition)
Client Port	21212	VoltDB command line
Admin Port	21211	Deployment file
Web Interface Port (httpd)	8080	Deployment file
Internal Server Port	3021	VoltDB command line
Log Port	4560	(Enterprise Edition feature)
JMX Port	9090	(Enterprise Editiion feature)
Zookeeper port	2181	VoltDB command line

A.5.1. Client Port

The client port is the port VoltDB client applications use to communicate with the database cluster nodes. By default, VoltDB uses port 21212 as the client port. You can change the client port. However, all client applications must then use the specified port when creating connections to the cluster nodes.

To specify a different client port on the command line, use the `port` parameter when starting the VoltDB database. For example, the following command starts the database using port 12345 as the client port:

```
$ voltdb create host localhost \
         catalog mycatalog.jar deployment deployment.xml \
         port 12345
```

When using the Enterprise Manager, use the Edit Configuration dialog box to specify the port to use.

If you change the default client port, all client applications must also connect to the new port. The client interfaces for Java and C++ accept an additional, optional argument to the createConnection method for this purpose. The following examples demonstrate how to connect to an alternate port using the Java and C++ client interfaces.

Java

```
org.voltdb.client.Client voltclient;
voltclient = ClientFactory.createClient();
voltclient.createConnection("myserver",12345);
```

C++

```
boost::shared_ptr<voltdb::Client> client = voltdb::Client::create();
client->createConnection("myserver", 12345);
```

A.5.2. Admin Port

The admin port is similar to the client port, it accepts and processes requests from applications. However, the admin port has the special feature that it continues to accept requests when the database enters admin mode.

By default, VoltDB uses port 21211 as the admin port. You can change the port assignment in the deployment file using the <admin-mode> tag. For example, the following deployment file sets the admin port to 2222:

```
<deployment>
   ...
   <admin-mode port="2222" />
</deployment>
```

When using the Enterprise Manager, use the Edit Configuration dialog box to specify a different admin port.

A.5.3. Web Interface Port (httpd)

The web interface port is the port that VoltDB listens to for web-based connections from the JSON interface. There are two related settings associated with the JSON interface. The first setting is whether the port is enabled; the second is which port to use, if the interface is enabled.

When starting a VoltDB database manually, the web interface port (and the JSON interface) or disabled by default. When using the Enterprise Manager, both the web interface port and the JSON interface are enabled by default. The default httpd port is 8080.

If you plan on using the JSON interface from the community edition, be sure to include the <httpd> tag in the deployment file.

- To enable the httpd port but disable the JSON interface, specify the attribute `enabled="false"` in the `<jsonapi>` tag in the deployment file when starting VoltDB. If you are using the Enterprise Manager, there is a check box for enabling and disabling the JSON interface in the Edit Configuration dialog box.
- To change the web interface port, specify the alternate port using the `port` attribute to the `<httpd>` tag in the deployment file. Or, if you are using the Enterprise Manager use the httpd port field in the Edit Configuration dialog.

For example, the following deployment file fragment enables the web interface and the JSON interface, specifying the alternate port 8083.

```
<httpd port='8083'>
        <jsonapi enabled='true'/>
</httpd>
```

If you change the port number, be sure to use the new port number when connecting to the cluster using the JSON interface. For example, the following URL connects to the port 8083, instead of 8080:

```
http://athena.mycompany.com:8083/api/1.0/?Procedure=@SystemInformation
```

For more information about the JSON interface and specifying the appropriate port when connecting to the VoltDB cluster, see the section on "How the JSON Interface Works" in the *Using VoltDB* manual.

A.5.4. Internal Server Port

A VoltDB cluster uses ports to communicate among the cluster nodes. This port is internal to VoltDB and should not be used by other applications.

By default, the internal server port is port 3021 for all nodes in the cluster[1]. You can specify an alternate port using the `internalport` parameter when starting the VoltDB process. For example, the following command starts the VoltDB process using an internal port of 4000:

```
$ voltdb create host localhost \
         catalog mycatalog.jar deployment deployment.xml \
         internalport 4000
```

When using the Enterprise Manager, you can change the starting internal port number by using the Edit Database dialog, as described in Section 4.2, “Changing the Database Configuration”.

A.5.5. Log Port (Enterprise Edition Feature)

When using the Enterprise Manager to configure and run VoltDB, the resulting VoltDB cluster nodes open a port as an output stream for log4J messages. The Enterprise Manager uses the port to fetch log4J messages from the cluster nodes and display them in the management console.

By default, port 4560 is assigned as the log port. You can change this port using the Edit Database dialog (described in Section 4.2, “Changing the Database Configuration”).

A.5.6. JMX Port (Enterprise Edition Feature)

The VoltDB Enterprise Manager uses JMX to collect statistics from the cluster nodes at runtime. It does this using JMX. The default JMX port is 9090. However, you can change this port in the Edit Database dialog box as described in Section 4.2, “Changing the Database Configuration”.

[1] In the special circumstance where multiple VoltDB processes are started for one database, all on the same server, the internal server port is incremented from the initial value for each process.

There is no JMX port when using the VoltDB Community Edition.

A.5.7. Zookeeper Port

VoltDB uses a version of Apache Zookeeper to communicate among supplementary functions that require coordination but are not directly tied to database transactions. Zookeeper provides reliable synchronization for functions such as command logging without interfering with the database's own internal communications.

VoltDB uses a network port bound to the local interface (127.0.0.1) to interact with Zookeeper. By default, 2181 is assigned as the Zookeeper port. You can specify a different port number using the `zkport` parameter when starting the VoltDB process. For example:

```
$ voltdb create host localhost \
         catalog mycatalog.jar deployment deployment.xml \
         zkport 2288
```

When using the Enterprise Manager, you can change the Zookeeper port by using the Edit Database dialog, as described in Section 4.2, “Changing the Database Configuration”.

Appendix B. VoltDB Management REST Interface

The Enterprise Manager provides a complete, interactive user interface described earlier in this book. However, there are times when it is desirable to automate frequently repeated actions or to script a series of actions into a single function.

To allow for these sorts of customizations, the VoltDB Enterprise Manager also provides a programmable interface based on the REST ("Representational State Transfer") architecture. If you are familiar with REST interfaces, you can skip to the reference section at the end of this chapter. If you are new to REST interfaces, the following sections provide a quick overview of:

- How to program against the VoltDB management interface
- What resources are available to you
- A sample application using the REST API

B.1. Introduction to the REST Interface

The VoltDB Management interface uses the REST architecture to provide programmable services. At its simplest, the REST architecture allows an application to use HTTP requests to operate on a set of defined resources. For example, the following HTTP request adds a server to the Enterprise Manager's list of resources:

```
POST❶  /man/api/1.0/mgmt/servers❷  HTTP/1.1
User-Agent: curl/7.21.0 (x86_64-pc-linux-gnu)
Host: voltdbmgr:8080
Accept: */*
Content-Length: 71
Content-Type: application/x-www-form-urlencoded

{"Server":{"name":"Test Server","host":"localhost",""sshuser":"admin"}}❸
```

The key to understanding how the REST architecture works is to understand the components of the HTTP request.

❶ The request *method* specifies the action to take.
❷ The *URL* specifies the resource to operate on.
❸ The *body of the request* specifies any arguments, values, or content needed to complete the action.

So in the previous example, the request adds (POST) a server resource (/man/api/1.0/mgmt/servers) using the arguments in the request body (name="Test Server", host="localhost", and sshuser="admin"). It is important to note that for the VoltDB REST interface, with only one or two exceptions, all of the input and output (that is, the body of the requests and responses) is encoded as JSON strings. This is not required by the REST architecture per se, but provides a consistent standards-based mechanism for communicating through the VoltDB API.

The advantage of a REST interface is that many requests can be accomplished using a web browser or command line utility such as curl. For example, the following Linux shell script issues two requests to add servers and then requests a list of the servers currently defined within the Enterprise Manager.

```
curl --data "{'Server':{'name':'Server #1','host':'MyServer1',}}" \
            http://voltdbmgr:9000/man/api/1.0/mgmt/servers
curl --data "{'Server':{'name':'Server #2','host':'MyServer2',}}" \
            http://voltdbmgr:9000/man/api/1.0/mgmt/servers
curl http://voltdbmgr:9000/man/api/1.0/mgmt/servers
```

B.1.1. Resources and Methods

Once you understand the components of the REST request, the key to making the interface useful is understanding what resources are available and what actions you can perform. For the VoltDB Management interface, there are five types of resources you can operate on:

- Catalogs
- Databases
- Deployments
- Servers
- Snapshots

You specify the type of resource as part of the URL. For example, `/man/api/1.0/mgmt/databases` refers to database resources. To act on a specific resource, you include the resource's internal ID in the URL. So, for example, the following command deletes the database with the internal ID of "154":

```
curl -X DELETE http://voltdbmgr:8080/man/api/1.0/mgmt/databases/154
```

For any given resource, there can be up to four methods you can apply to the resource:

- GET — fetch information about the resource
- POST — add a new resource
- PUT — update an existing resource
- DELETE — remove an existing resource

Section B.2, “VoltDB Management Interface: Reference Section” explains which methods are possible for each resource type and what arguments are required to perform a POST or PUT.

In addition, there are special actions possible for database resources only. For the database itself, the actions are START, STOP, PAUSE, RESUME, and SNAPSHOT. You can also perform a START and STOP on individual servers assigned to a database.

Since there are no equivalent HTTP methods for these actions, you invoke them by specifying the action as part of the URL and use the HTTP PUT method. For example, the following command starts the database identified by the ID 236:

```
curl -X PUT http://voltdbmgr:9000/man/api/1.0/mgmt/databases/236/start
```

B.1.2. Interpreting Status Codes and Return Values

When you use the VoltDB Management Interface, the REST services can return information to your program or script in several different ways. The three most important pieces of information the REST interface returns are the status code, any error messages if the request failed, and the return data when the request succeeds.

The REST interface uses standard HTTP response status codes to indicate the success or failure of requests. The specific return code depends upon the action requested. Table B.1, "REST Success Codes" describes the status codes that are returned when a call to the REST interface succeeds.

Table B.1. REST Success Codes

Method	HTTP Status	Description
GET	200	The information requested is returned in the body of the response as a JSON-encoded string.
POST	303	When adding a new object, the URL of the created resource is returned as the redirect location in the HTTP header.
POST	200	When requesting an action via a POST other than creating a new object (such as requesting a comparison of two catalogs), success is reported as code 200.
PUT	200	When updating a resource, success is reported as code 200.
PUT	202	When requesting an extended action (such as start, stop, or update) which is performed asynchronously, the successful initiation of that action is reported as code 202. The application should query the object periodically to check the status to see if and when the action completes.
DELETE	200	Successful deletion of a resource is reported as code 200.

As noted in the tables, the information returned by the response also depends upon the action requested. For GET methods, the information is returned in the body of the response as a JSON-encoded string. For successful POST actions, the URL for the created resource is returned in the redirect location in the HTTP header.

If a request fails, the VoltDB interface returns both an error status using the HTTP code and a detailed description of the error in the body of the response. The return status depends on the nature of the failure. Table B.2, "REST Error Codes" describes the status codes that are returned when a call to the REST interface fails.

Table B.2. REST Error Codes

Error	HTTP Status	Description
Invalid request	400	The HTTP request URL or JSON contents are invalid. See the body of the response for more information on the specific error.
Invalid action	403	The action you requested is not currently permitted. This usually involves attempts to update or delete an object (such as a database) that is currently in use. Check the status of the object to determine what actions (such as stopping the database or server) are needed before repeating this request.
Object not found	404	A valid request was made but the referenced object does not exist. This can happen if you try to update a server or database that has already been deleted or if you specify an invalid object ID.
Internal error	500	An unexpected error occurred while attempting to execute your request. This condition should not occur during normal operation. See the body of the response for more information about the specific error.

Note that the header, including the status code, is separate from the standard content of the response. You may need to use special arguments to capture the header in the output. For example, the following curl command uses the -w (write-out) argument to include the HTTP code in the output:

```
curl -w "Code: %{http_code}\n" http://castor:9000/man/api/1.0/mgmt/servers
```

B.1.3. Examples of Using the Management Interface

The best way to understand an interface is often to see it in action. The following are some examples of Linux shell scripts that use the VoltDB Management interface to create and manage databases.

The following script creates the necessary servers, deployment, and catalog resources to define and start a database. In this case, it uses a script routine (RestAdd) to create each of the resources and return the resulting resource ID.

Note the use of the curl -w argument to write the redirect location as part of the output so the script can capture the resource ID created by the REST service. This is necessary since the ID is not in the body of the HTTP resource. An alternative approach is to use -L to have curl follow the redirection. In which case the full JSON representation of the new resource, including the ID, will be in the body of the response.

```
#
# Sample script to create database.
#
MGTHOST="http://voltdbmgr:9000"
RestAdd () {
  ID=$( curl -sw "%{redirect_url}" -d "$2" \
          "$MGTHOST/man/api/1.0$1" \
          | grep -oP "(?<=$1/)[0-9]+" )
  echo "$ID"
}

# Define Three servers
JSON='{"Server":{"host":"goneril","name":"goneril"}}'
S1=$( RestAdd "/mgmt/servers" "$JSON" )
JSON='{"Server":{"host":"regan","name":"regan"}}'
S2=$( RestAdd "/mgmt/servers" "$JSON" )
JSON='{"Server":{"host":"cordelia","name":"cordelia"}}'
S3=$( RestAdd "/mgmt/servers" "$JSON" )

# Define a deployment
JSON='{"Deployment":{"name":"Flight deployment"}}'
D1=$( RestAdd "/mgmt/deployments" "$JSON" )

# Load a catalog (use base64-encoding)
CATALOG=$( base64 flight.jar )
JSON="{\"Catalog\":{\"name\":\"flight\",\
       \"catalogbytes\":\"$CATALOG\"}}"
C1=$( RestAdd "/mgmt/catalogs" "$JSON" )

# Define and start database
JSON="{\"Database\":{\"name\":\"Flight DB\",\
      \"catalog\":\"$C1\",\"deployment\":\"$D1\",\
      \"remotedir\":\"flightdb\",\
      \"Servers\":[{\"server\":\"$S1\"},\
      {\"server\":\"$S2\"},{\"server\":\"$S3\"}]}}"
```

```
DB=$( RestAdd "/mgmt/databases" "$JSON" )
curl -X PUT "$MGTHOST/man/api/1.0/mgmt/databases/$DB/start"
```

B.2. VoltDB Management Interface: Reference Section

The VoltDB management interface provides a set of management functions that let you create, update, and remove resources within the Enterprise Manager. These management services let you define databases, add servers and catalogs, and stop and start the databases, all programmatically using HTTP calls. These services are available from URLs starting with /man/api/1.0/mgmt/.

The management interface also provides a set of informational services, that return statistical data about running databases, the differences between two catalogs, and log messages from database servers and the Enterprise Manager itself. The URLs for information services start with /man/api/1.0/info or /man/api/1.0/log.

Table B.3, "Summary of the VoltDB Management REST Interface" summarizes the resources and methods available through the VoltDB management interface.

Table B.3. Summary of the VoltDB Management REST Interface

URI	GET	POST	PUT	DELETE
/mgmt/catalogs	List catalogs	Add catalog		
/mgmt/catalogs/{id}	Catalog object			Delete catalog
/mgmt/databases	List databases	Add database		
/mgmt/databases/{id}	Database object		Update database properties	Delete database
/mgmt/databases/{id}/start			Creates database	
/mgmt/databases/{id}/recover			Recovers database	
/mgmt/databases/{id}/stop			Stops database	
/mgmt/databases/{id}/pause			Pauses database	
/mgmt/databases/{id}/resume			Resumes database	
/mgmt/databases/{id}/snapshot			Initiates a snapshot	
/mgmt/databases/{id}/servers/{id}/start			Starts the server	
/mgmt/databases/{id}/servers/{id}/stop			Stops the server	
/mgmt/deployments	List deployments	Add deployment		
/mgmt/deployments/{id}	Deployment object		Update deployment properties	Delete deployment

URI	GET	POST	PUT	DELETE
/mgmt/servers	List servers	Add a server		
/mgmt/servers/{id}	Server object		Update server properties	Delete server
/mgmt/snapshots	List snapshots			
/mgmt/snapshots/{id}	Snapshot object			Delete snapshot
/info/catalogdiffs	List differences between two catalog objects			
/log/messages/	List messages from the database servers			
/log/managements/	List messages from the management server			

The following pages provide detailed information concerning each service.

/mgmt/catalogs

/mgmt/catalogs — Manage catalog resources

Syntax

```
/mgmt/catalogs

/mgmt/catalogs/{catalog-id}
```

Methods

GET — Fetches information about the current catalogs defined within the Enterprise Manager. A call without a catalog ID returns a list of the known catalogs. A call with a catalog ID returns information about that catalog specifically.

POST — Adds a catalog to the Enterprise Manager. Since you must provide the contents of the catalog itself, this is the only call that does not use JSON to encode the body of the request. Instead, you must provide the catalog JAR file as a multipart-form element.

DELETE — Deletes the specified catalog from the Enterprise Manager. You must specify a valid catalog ID in the URL.

Arguments

For the POST method, you must provide the catalog JAR file in the body of the call. Since the catalog is a binary file, you must Base64-encode the file's contents to pass it to the service as part of the JSON input. The valid input arguments are the following:

Argument Name	Type	Description
name	text	The name of the catalog
catalogbytes*	file data	The binary contents of a VoltDB catalog JAR file

*Required

When performing a GET, the REST interface returns information about the catalog rather than the catalog file itself. The structure of the JSON object returned for catalogs is as follows:

Attribute Name	Type	Description
name	text	The name of the catalog
version	text (read-only)	The version specified in the catalog
description	text (read-only)	The description specified in the catalog

Examples

The following example uses curl to create a new catalog object, encoding the JAR file `examples/voter/voter.jar`:

```
CATALOG =$( base64 examples/voter/voter.jar )
JSON="{\"Catalog\":{\"name\":\"Voter\",\"catalogbytes\":\"$CATALOG\"}}"
```

```
curl --data "$JSON" http://voltdbmgr:9000/man/api/1.0/mgmt/catalogs
```

/mgmt/databases

/mgmt/databases — Manage database resources

Syntax

```
/mgmt/databases
/mgmt/databases/{database-id}
/mgmt/databases/{database-id}/start
/mgmt/databases/{database-id}/startreplica
/mgmt/databases/{database-id}/recover
/mgmt/databases/{database-id}/stop
/mgmt/databases/{database-id}/pause
/mgmt/databases/{database-id}/resume
/mgmt/databases/{database-id}/snapshot
```

Methods

GET	Fetches information about the current databases defined within the Enterprise Manager. A call without a database ID returns a list of the known databases. A call with a database ID returns information about that specific database — including the IDs for the deployment, catalog, and server(s).
POST	Adds a database to the Enterprise Manager. You must provide valid IDs for an existing deployment resource, catalog, and one or more servers as part of the request.
PUT	Replaces the database specified by ID in the URL with the database properties defined in the body of the request. It is always best to specify all of the properties when performing a replace, even if only one or two properties are changing. (For example, when changing the name of the database.) Any properties not specified are replaced with the default value, *not* the current value. The easiest way to perform an update is to fetch the current properties by doing a GET on the database ID, changing those properties that need changing, and then doing a PUT with the updated JSON structure, ensuring all unchanged properties retain their current value.
START	Starts the database specified by ID in the URL, creating a new, empty database. You specify "start" as part of the URL and send the request using the PUT method. Optionally, you can add the argument "?pause=1" to the start method URL to start the database in admin mode.
STARTREPLICA	Starts the database specified by ID in the URL, creating a new replica (read-only) database. You specify "startreplica" as part of the URL and send the request using the PUT method. See the *Using VoltDB* manual for more information about replica databases and the database replication process.

	Optionally, you can add the argument "?pause=1" to the start method URL to start the database in admin mode.
RECOVER	Starts the database specified by ID in the URL and recovers the database contents from the command logs of the last session. You specify "recover" as part of the URL and send the request using the PUT method. Optionally, you can add the argument "?pause=1" to the recover method URL to start the database in admin mode.
STOP	Stops the database specified by ID in the URL. You must specify "stop" as part of the URL and send the request using the PUT method.
PAUSE	Pauses the database specified by ID in the URL. That is, initiates admin mode on the specified cluster. You must specify "pause" as part of the URL and send the request using the PUT method.
RESUME	Resumes, or unpauses, the database specified by ID in the URL, returning the cluster to normal operation. You must specify "resume" as part of the URL and send the request using the PUT method.
SNAPSHOT	Initiates a manual snapshot for the database specified by ID in the URL. Once the snapshot is complete, the snapshot files are copied to the management server and added as a snapshot resource, also available through the REST interface. You must specify "snapshot" as part of the URL and send the request using the PUT method.
DELETE	Deletes the specified database from the Enterprise Manager. You must specify a valid database ID in the URL. The database must be stopped before it can be deleted.

Arguments

To create or update a database using the POST and PUT methods, you must provide information about the database as arguments in the body of the request. The arguments must be encoded as a JSON string. The structure of the database object is as follows:

Attribute	Type	Description
name*	Text	The name of the database
status	Text (read-only)	Current state of the database (running, stopped, or paused)
description	Text	Description of the database
maxheap	Integer	The max heap size (in megabytes) to use when starting the server(s)
port	Integer	The client port
internalport	Integer	The internal server port
jmxport	Integer	The JMX port
logport	Integer	The log4j port
zookeeperport	Integer	The zookeeper port
catalog*	Integer	ID of the catalog to use
deployment*	Integer	ID of the deployment attributes to use
collectsnapshots	Boolean	Whether to copy snapshots to the management server or not
servers	List	Collection of servers assigned to the database. On input, the collection is specified as an array of "server":{server-ID} pairs. On output, each

Attribute	Type	Description
		server object includes read-only attributes specifying the name, IP address, and status of the server.
snapshots	List (read-only)	Collection of snapshots that the management server has copied. The collection is an array of "snapshot":{snapshot-ID} pairs.

*Required

Examples

The following example uses curl to create a new database from existing catalog and deployment resources:

```
curl --data '{"Database":{"name":"Cars",
                          "catalog":"16777261",
                          "deployment":"33554476"}}' \
     http://voltdbmgr:9000/man/api/1.0/mgmt/databases
```

The next example uses GET and PUT to update a database and assign an existing server resource. The script uses GET to retrieve the full database object, then uses sed to replace the Servers property before using PUT to update the database with the modified JSON string.

```
JSON=$( curl http://voltdbmgr:9000/man/api/1.0/mgmt/databases/83886132 )
JSON=$( echo $JSON | \
        sed 's/"Servers":\[\]/"Servers":\[{"server":67108911}\]/g' )
curl -X PUT --data "$JSON" \
http://voltdbmgr:9000/man/api/1.0/mgmt/databases/83886132
```

/mgmt/databases/{id}/servers

/mgmt/databases/{id}/servers — Manage servers assigned to a database

Syntax

```
/mgmt/databases/{database-id}/servers/{server-id}/start

/mgmt/databases/{database-id}/servers/{server-id}/stop
```

Methods

START Starts the VoltDB database process for the database and on the server specified by the IDs in the URL. You must specify "start" as part of the URL and send the request using the PUT method.

You can only start a server under the following conditions:

- The server is already assigned to the specified database
- The database has started and is running
- The server has stopped, either due to an error condition or a previous STOP action

STOP Stops the VoltDB database process for the database and on the server specified by the IDs in the URL. You must specify "stop" as part of the URL and send the request using the PUT method.

You can only stop a server under the following conditions:

- The server is already assigned to the specified database
- The database has started and is running
- The VoltDB process for the specified database is running on the server

Arguments

There are no arguments to the START or STOP actions. Also START and STOP are the only actions allowed. To get a list of the servers currently assigned to a database, you must perform a GET on the database object itself. Similarly, to change the assignment of servers to the database, you must perform a PUT on the database object directly.

Examples

The following example stops the VoltDB process for database 83886132 on server 67108911:

```
curl -X PUT \
http://voltdbmgr:9000/man/api/1.0/mgmt/databases/83886132\
/servers/67108911/stop
```

/mgmt/deployments

/mgmt/deployments — Manage deployment resources

Syntax

```
/mgmt/deployments

/mgmt/deployments/{deployment-id}
```

Methods

GET Fetches information about the deployments defined within the Enterprise Manager. A call without a deployment ID returns a list of the known deployments. A call with a deployment ID returns the attributes for that deployment.

POST Adds a deployment to the Enterprise Manager. You must provide the attributes of the deployment as arguments in the body of the request.

PUT Updates the deployment specified by ID in the URL with the attribute values in the body of the request.

DELETE Deletes the specified deployment from the Enterprise Manager. You must specify a valid deployment ID in the URL and the deployment cannot be currently included in any active database object.

Arguments

A deployment is a set of attributes used for configuring the database servers at startup. These attributes are similar to the arguments you specify in a deployment file when starting a VoltDB database manually.

For the POST and PUT methods, you must provide the attribute values in the body of the call. You specify the attributes as a JSON-encoded string. The structure of the deployment object is as follows:

Argument Name	Type	Description
name	Text	The name of the deployment
voltroot	Text	The destination directory where files are stored on the cluster nodes
kfactor	Integer	The K-safety value for the cluster
sitesperhost	Integer	The number of sites per host to create on each node
httpport	Integer	The HTTP port
jsonenabled	Boolean	Whether the JSON API is to be enabled or not
snapshotpath	String	The path where automated snapshots are stored on the cluster nodes. Paths are relative to the VoltDB root path, which is a subfolder of voltroot, using the database ID as the subfolder name.
snapshotfrequency	Integer	The frequency (in seconds) for automated snapshots. A frequency of zero disables automated snapshots.

Argument Name	Type	Description
snapshotretain	Integer	The number of automated snapshots to keep on the server(s). If more automated snapshots are created, any older snapshots above the retention limit are purged.
export	Boolean	Whether export is enabled or not
exportoverflowpath	String	The path where export overflow data is stored on the cluster nodes. Paths are relative to the VoltDB root path, which is a subfolder of voltroot, using the database ID as the subfolder name.
security	Boolean	Whether security is enabled or not
partitiondetection	Boolean	Whether network partition detection is enabled or not
adminport	Integer	The admin port number
userbytes	Byte String	The contents of a VoltDB deployment containing user definitions to import
heartbeattimeout	Integer	The cluster heartbeat timeout interval (in seconds)
commandlogenabled	Boolean	Whether command logging is enabled or not
commandlogsync	Boolean	Whether command logging is synchronous (true) or not (false)
commandlogfreqtime	Integer	The frequency of command logging (in milliseconds)
commandlogfreqtxns	Integer	The frequency of command logging (in number of transactions)
commandlogpath	String	The path where command logs are written. Paths are relative to the VoltDB root path, which is a subfolder of voltroot, using the database ID as the subfolder name.
commandlogsnapshot path	String	The path where command log snapshots are stored. Paths are relative to the VoltDB root path, which is a subfolder of voltroot, using the database ID as the subfolder name.
commandlogsize	Integer	The initial space allocated for the command logs (in megabytes)
snapshotpriority	Integer	The priority of snapshot activity (from 0 to 10)
maxtemptablememory	Integer	The maximum size of temp table memory usage (in megabytes)

Examples

Since no arguments are required, it is possible to specify an empty top-level JSON object, to create a deployment with the default settings:

```
curl --data '{"Deployment":{}}' \
http://localhost:9000/man/api/1.0/mgmt/deployments/
```

The next example uses several attributes to create a deployment useful for local testing (in this case, without K-safety and with a destination directory in /tmp):

```
curl --data '{"Deployment":{"name":"test",
                            "kfactor":"0",
                            "voltroot":"/tmp/test"}}' \
http://localhost:9000/man/api/1.0/mgmt/deployments/
```

The last example demonstrates how to load user names and passwords into the deployment object. Security credentials are imported from an existing VoltDB deployment file. In this example, two user definitions are written to a file, Base64 encoded, then inserted using the userbytes argument:

```
echo '<deployment><users>
   <user name="spade" password="hammett" groups="users"/>
   <user name="marlowe" password="chandler" groups="users"/>
      </users></deployment>' > userdef.xml
BYTES=$( base64 userdef.xml )
curl --data "{\"Deployment\":{\"name\":\"test\", \
            \"userbytes\":\"$BYTES\"}}" \
http://localhost:9000/man/api/1.0/mgmt/deployments/
```

/mgmt/servers

/mgmt/servers — Manage server resources

Syntax

```
/mgmt/servers

/mgmt/servers/{server-id}
```

Methods

GET Fetches information about the current servers defined within the Enterprise Manager. A call without a server ID returns a list of the known servers. A call with a server ID returns information about that server specifically.

POST Adds a server to the Enterprise Manager. You specify the IP address and other information about the server as arguments in the body of the request.

PUT Update the server specified by ID in the URL with the attributes in the body of the request.

DELETE Deletes the specified server from the Enterprise Manager. You must specify a valid server ID in the URL. Also, the server must not be currently assigned to any databases or the DELETE action will fail.

Arguments

For the POST and PUT methods, you must provide attributes of the server in the body of the call. You specify the attributes as a JSON-encoded string. The structure of the deployment object used for input and output is as follows

Argument Name	Type	Description
name	Text	The name of the server
host*	Text	The IP address or host name of the server
sshuser	Text	The username for the management server to use when accessing the server via ssh
sshkey	Text	The ssh key to use when accessing the server via ssh
interfaces	List	The internal and external network interfaces used by the server

*Required

Examples

This example adds a server to the management server:

```
curl --data '{"Server":{"host":"zeus"}}' \
http://voltdbmgr:9000/man/api/1.0/mgmt/servers/
```

The next example attempts to delete a server. The response will indicate whether the action succeeded or not.

```
curl -X DELETE  \
http://voltdbmgr:9000/man/api/1.0/mgmt/servers/67108917
```

Because the SSH keyfile is binary, you must upload the key as a Base64-encoded text string. The next example loads a specific username and keyfile.

```
KEY=$( base64 keyfile )
curl --data "{\"Server\":{\"host\":\"hera\",\
              \"sshuser\":\"juno\",\
              \"sshkey\":\"$KEY\"}}"  \
http://voltdbmgr:9000/man/api/1.0/mgmt/servers/
```

/mgmt/snapshots

/mgmt/snapshots — Manage snapshot resources

Syntax

```
/mgmt/snapshots

/mgmt/snpashots/{snapshot-id}
```

Methods

GET — Fetches information about the current snapshots stored on the management server. A call without a snapshot ID returns a list of snapshots. A call with a snapshot ID returns information about that snapshot specifically.

DELETE — Deletes the specified snapshot from the Enterprise Manager. You must specify a valid snapshot ID in the URL.

Arguments

There are no input arguments to the snapshot methods, since the snapshot objects themselves are immutable. However, the structure of the snapshot object returned on a GET is as follows:

Argument Name	Type	Description
nonce	Text (read-only)	The unique identifier used as a prefix for snapshot files
timestamp	Integer (read-only)	The timestamp identifying when the snapshot was taken
size	Integer (read-only)	The size of the snapshot (in bytes)

Examples

The following example gets a list of all of the snapshots on the management server:

```
curl http://voltdbmgr:9000/man/api/1.0/mgmt/snapshots/
```

/info/catalogdiffs

/info/catalogdiffs — compares two catalogs and reports on the differences

Syntax

```
/info/catalogdiffs
```

Methods

POST When you specify the IDs for two existing catalog objects as arguments to this METHOD, VoltDB returns information about the difference between, including any tables or stored procedures that have been added, removed, or modified.

Arguments

You must specify the IDs of the two catalogs to compare in the attributes `lhs_id` and `rhs_id`. The complete JSON structure is returned in the results. The structure of the JSON object is as follows:

Argument Name	Type	Description
lhs_id*	Text	The ID of one of the catalogs to compare
rhs_id*	Text	The ID of the other catalog to compare
restart	Boolean (read-only)	Whether a database must be restarted to update from the one catalog to the other.
ProjectDiffs	List (read-only)	A collection of objects identifying changes to features within the catalog, such as enabling or disabling security
TableDiffs	List (read-only)	A collection of objects identifying changes to schema, one entry per table where tables have been added, removed, or modified.
ProcedureDiffs	List (read-only)	A collection of objects identifying changes to stored procedures in the catalog, one entry per procedure where procedures have been added, removed, or modified.

*Required

Examples

The following example compares two catalogs:

```
curl -d '{"Catalogdiff":{"lhs_id":16777280,"rhs_id":16777265}}'\
     http://voltdbmgr:9000/man/api/1.0/info/catalogdiffs
```

/log/messages

/log/messages — Returns log messages from all database and servers

Syntax

```
/log/messages
```

Methods

GET Fetches log messages from all of the databases defined within the Enterprise Manager. The /log/messages method returns the messages from the database servers.

Arguments

No arguments.

Examples

The following example fetches a collection of log messages for all databases defined in the management server:

```
curl http://voltdbmgr:9000/man/api/1.0/log/messages
```

/log/managements

/log/managements — Returns log messages from the Enterprise Manager

Syntax

```
/log/managements
```

Methods

GET Fetches log information from the Enterprise Manager itself. The combination of the /log/messages and /log/managements methods returns all of the log messages from the entire system.

Arguments

No arguments.

Examples

The following example fetches a collection of log messages from the Enterprise Manager:

```
curl http://voltdbmgr:9000/man/api/1.0/log/managements
```

Appendix C. Snapshot Utilities

VoltDB provides two utilities for managing snapshot files. These utilities verify that a snapshot is complete and usable and convert the snapshot contents to a text representation that can be useful for uploading or reloading data in case of severe errors.

It is possible, as the result of a design flaw or failed program logic, for a database application to become unusable. However, the data is still of value. In such emergency cases, it is desirable to extract the data from the database and possibly reload it. This is the function that save and restore performs within VoltDB.

But there may be cases when you want to use the data created by a VoltDB snapshot elsewhere. The goal of the utilities is to assist in that process. The snapshot utilities are:

- *SnapshotConverter* converts a snapshot (or part of a snapshot) into text files, creating one file for each table in the snapshot.
- *SnapshotVerifier* verifies that a VoltDB snapshot is complete and usable.

Unlike system procedures, that must be run within the context of an existing database connection, the snapshot utilities can be run from the Linux shell without a running database present. However, the utilities are still dependent on the Java classes and library for VoltDB. So you must be sure to define your Java classpath and library path appropriately to invoke the classes.

To run either SnapshotVerifier or SnapshotConverter, use the Java command to invoke the class, specifying the appropriate classpath and library path based on where your VoltDB software is installed. For example, if VoltDB is installed in /opt/voltdb, the command to invoke SnapshotVerifier is as follows:

```
$ java -classpath "/opt/voltdb/voltdb/*" \
       -Djava.library.path=/opt/voltdb/voltdb \
       org.voltdb.utils.SnapshotVerifier
```

You may find it easier to add an alias to your shell script startup file to abbreviate these commands:

```
$ alias snapshotverify="java -cp \"/opt/voltdb/voltdb/*\"  \
        -Djava.library.path=/opt/voltdb/voltdb \
         org.voltdb.utils.SnapshotVerifier "
$ alias snapshotconvert="java -cp \"/opt/voltdb/voltdb/*\"  \
        -Djava.library.path=/opt/voltdb/voltdb \
        org.voltdb.utils.SnapshotConverter "
```

The following sections describing each command assume these aliases have been defined.

Each command accepts a different set of arguments. Use the `--help` argument to display a list the allowable arguments and qualifiers. For example:

```
$ snapshotverify --help
```

snapshotconvert

snapshotconvert — Converts the tables in a VoltDB snapshot into text files.

Syntax

```
snapshotconvert {snapshot-id} --type {csv|tsv} \
        --table {table} [...] [--dir {directory}]... \
        [--outdir {directory}]

snapshotconvert --help
```

Description

SnapshotConverter converts one or more tables in a valid snapshot into either comma-separated (csv) or tab-separated (tsv) text files, creating one file per table.

Where:

{snapshot-id} is the unique identifier specified when the snapshot was created. (It is also the name of the .digest file that is part of the snapshot.) You must specify a snapshot ID.

{csv|tsv} is either "csv" or "tsv" and specifies whether the output file is comma-separated or tab-separated. This argument is also used as the filetype of the output files.

{table} is the name of the database table that you want to export to text file. You can specify the `--table` argument multiple times to convert multiple tables with a single command.

{directory} is the directory to search for the snapshot (`--dir`) or where to create the resulting output files (`--outdir`). You can specify the `--dir` argument multiple times to search multiple directories for the snapshot files. Both `--dir` and `--outdir` are optional; they default to the current directory path.

Example

The following command exports two tables from a snapshot of the flight reservation example used in the *Using VoltDB* manual. The utility searches for the snapshot files in the current directory (the default) and creates one file per table in the user's home directory:

```
$ snapshotconvert flightsnap --table CUSTOMER --table RESERVATION \
                  --type csv -- outdir ~/
```

snapshotverify

snapshotverify — Verifies that the contents of one or more snapshot files are complete and usable.

Syntax

```
snapshotverify [snapshot-id] [--dir {directory}] ...

snapshotverify --help
```

Description

SnapshotVerifier verifies one or more snapshots in the specified directories.

Where:

[snapshot-id] is the unique identifier specified when the snapshot was created. (It is also the name of the .digest file that is part of the snapshot.) If you specify a snapshot ID, only snapshots matching that ID are verified. If you do not specify an ID, all snapshots found will be verified.

{directory} is the directory to search for the snapshot. You can specify the `--dir` argument multiple times to search multiple directories for snapshot files. If you do not specify a directory, the default is to search the current directory.

Examples

The following command verifies all of the snapshots in the current directory:

```
$ snapshotverify
```

This example verifies a snapshot with the unique identifier "flight" in either the directory `/etc/voltdb/save` or `~/mysaves`:

```
$ snapshotverify flight --dir /etc/voltdb/save/ --dir ~/mysaves
```