



Release Notes

Product	VoltDB Community Edition
Version	2.6
Release Date	April 26, 2012

This document provides information about known issues and limitations to the current release of the VoltDB Community Edition. If you encounter any problems not listed below, please be sure to report them using the VoltDB forums at <http://community.voltdb.com/>. Thank you.

Important Base Platform Considerations

The recommended platform for production use of VoltDB is CentOS 5.6 or later, Ubuntu 10.4 or later, and Sun JDK 6 Update 20 or later. Macintosh OSX 10.6 is supported as a development platform. However, there are certain configuration options in the base platforms that are important when running VoltDB.

1.1. Disable Swapping

Swapping is an operating system feature that optimizes memory usage when running multiple processes. However, memory is a critical component of the VoltDB server process. Any contention for memory, including swapping, will have a very negative impact on performance and functionality.

We recommend using dedicated servers and disabling swapping when running the VoltDB database server process. Use the `swapoff` command to disable swapping on Linux systems. If swapping cannot be disabled for any reason, you can reduce the likelihood of VoltDB being swapped out by setting the kernel parameter `vm.swappiness` to zero.

1.2. Disable Card-Marking Optimization when using JDK 6 Update 18 through 20

Recent JDKs introduced an issue related to performance optimization and garbage collection. (See <http://java.sun.com/javase/6/webnotes/6u18.html> for details.) To use VoltDB (or any memory intensive application) with JDK 6 update 18 through 20, you should disable the card marking optimization, using the following command line argument:

```
java -XX:-ReduceInitialCardMarks
```

Note that this bug has been fixed in a more recent release, Sun JDK 6 Update 21. So the preceding workaround is only needed for Sun JDK and OpenJDK versions 6 update 18 through 20.

1.3. Turn off TCP window scaling in older Linux kernels.

There is a bug in older Linux versions (prior to 2.6.25) that can cause TCP messaging queues to stall and time out. (See <http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commitdiff;h=607bfbf2d55dd1cfe5368b41c2a81a8c9ccf4723> for details.)

If you are using CentOS or RedHat Enterprise Linux (RHEL) versions 5.4 or 5.5, you must either apply the preceding patch or turn off window scaling before running VoltDB in production mode. To turn off window scaling, issue the following shell command (or add the line `"net.ipv4.tcp_window_scaling=0"` to the file `/etc/sysctl.conf` to make the change persistent):

```
$ sudo /sbin/sysctl -w net.ipv4.tcp_window_scaling=0
```

Note that this bug is fixed for Linux releases after 2.6.25, including Ubuntu 9.10 and later, and backported to CentOS/RedHat 5.7 and 5.8. So no changes are needed for recent releases on those platforms or on the Macintosh OS.

- 1.4.** Turn off TCP segmentation offload and generic receive offload if cluster stability is a problem.

There is an issue where, under certain conditions, the use of TCP segmentation offload (TSO) and generic receive offload (GRO) can cause nodes to randomly drop out of a cluster. The symptoms of this problem are that nodes timeout — that is, the rest of the cluster thinks they have failed — although the node is still running and no other network issues (such as a network partition) are the cause.

Disabling TSO and GRO is recommended for any VoltDB clusters that experience such instability. The commands to disable offloading are the following, where *N* is replaced by the number of the ethernet card:

```
ethtool -K ethN tso off
ethtool -K ethN gro off
```

Note that these commands disable offloading temporarily. You must issue these commands every time the node reboots.

Upgrading From Older Versions

When upgrading from a previous version of VoltDB — especially with an existing database — there are a number of important notes that you should be aware of. Some changes to the structure and syntax of the VoltDB project and deployment files may make old application catalogs and configuration files incompatible with newer versions.

Although incompatible changes are avoided wherever possible, some changes are necessary to add new features. It is always recommended that applications catalogs be recompiled when upgrading the VoltDB version. It is also important to note that the catalog is saved as part of snapshots and command logging. As a consequence, you must be careful to ensure an incompatible catalog is not loaded accidentally by starting a database with the **start** or **recover** function after an upgrade.

The recommended process for upgrading VoltDB for a running database is as follows:

1. Place the database in admin mode using the @Pause system procedure (or VoltDB Enterprise Manager).
2. Perform a manual snapshot of the database (using @SnapShotSave).
3. Shutdown the database (using @Shutdown).
4. Upgrade VoltDB.
5. Recompile the application catalog using the new version of VoltDB.
6. Restart the database using the **create** option, the new catalog, and starting in admin mode (specified in the deployment file).
7. Restore the snapshot created in Step #2 (using @SnapshotRestore).
8. Return the database to normal operations (using @Resume).

When using the Enterprise Manager, it is also recommended that you delete the Enterprise Manager configuration files (stored by default in the `.voltdb` subfolder in the home directory of the current account) when performing an upgrade.

Changes Since the Last Release

Users of previous versions of VoltDB should take note of the following changes that will impact their existing applications. The following is a list of changes since the last release.

1. Release V2.6

1.1. Improved Messaging

The messages displayed on the console when compiling an application catalog or starting a VoltDB server have been improved. Fewer messages are displayed on the console by default, and their format has been simplified to make the messages clearer.

Note that a more thorough set of messages with an severity of INFO or above is still available and written, by default, to log files in the `/log` subfolder of the user's working directory. You can change both what is displayed and where logs are written by modifying the Log4J configuration file. See the chapter on logging in the *Using VoltDB* manual for details.

1.2. Improved JDBC Error Reporting

The messages returned by the VoltDB JDBC interface have been improved to provide more meaningful information concerning errors that occur at runtime.

1.3. Useful Warnings During Compilation

The VoltDB compiler now warns you if any of the SQL queries in your stored procedures (or defined as statements in the project definition file) could result in non-deterministic output. It is important that all SQL queries are deterministic, especially when using durability features such as K-safety, command logging, or database replication. In most cases, non-determinism is a result of an unsorted query; that is, a query without an appropriate index or ORDER BY clause.

If VoltDB detects such queries, a warning is issued, listing the query and the reason it is non-deterministic. When you receive such warnings, it is recommended that you add the appropriate index to the table or specify a sort order to resolve the non-determinism.

1.4. Simplified Command Line for Single Node Databases

The command for starting a single node VoltDB database has been simplified. If you are starting VoltDB on just one node (in other words, `hostcount=1` running on localhost), specifying the deployment file and leader node are now optional. For example, the following command starts a database using the catalog `myapp.jar` and default settings:

```
$ voltdb catalog myapp.jar
```

When you leave off the deployment file, VoltDB assumes one node, two sites per host, and a K-safety value of zero. If you want to enable other features (such as export or automatic snapshots), you must create the appropriate deployment file and specify it on the command line.

1.5. Changes to the Java Client API Statistics Methods

The statistics package within the Java client API has been modified to improve reporting capabilities. The following classes have been added:

```
org.voltdb.client.ClientStats  
org.voltdb.client.ClientStatsContext
```

These classes provide a more compact, complete, and consistent approach to statistics than in previous releases. See the javadoc (in the /doc subfolder) that accompanies the VoltDB software for more details. The new classes replace the following methods that have been removed from the API:

```
getIOStats();  
getIOStatsInterval();  
getProcedureStats();  
getProcedureStatsInterval();
```

Note, that these changes are specific to the Java Client API and do not affect other programming languages.

2. Release V2.5

2.1. Database Replication

The major addition for this release is support for database replication. Database replication is a commercial feature that is available through the VoltDB Enterprise Edition. See *Using VoltDB* for more information.

The community edition is being released simultaneously to ensure that the latest bug fixes and performance enhancements included in V2.5 are available to the open source community.

2.2. Updated Client APIs

Several of the client APIs — including C++, Java, PHP, and Python — have been updated to ensure their continued support of new features and capabilities.

3. Release V2.2.2

3.1. Java Version 7 Support

All known issues related to running VoltDB on Java version 7 have been resolved. Java 7 is a supported development and production environment for VoltDB.

3.2. Joining Partitioned Tables on Their Partitioning Column

VoltDB does not support joining two partitioned tables on an arbitrary column in multi-partitioned stored procedures. However, it is now possible to join two tables that are partitioned on the same column, provided the join is based on the equality of the partitioning column. For example, the following is now a valid SQL statement in either single-partitioned or multi-partitioned stored procedures, as long as the tables EMPLOYEE and MANAGER are both partitioned on DEPT_ID:

```
SELECT E.EMP_NAME, M.MGR_NAME  
       FROM EMPLOYEE AS E, MANAGER AS M  
       WHERE E.DEPT_ID = M.DEPT_ID;
```

4. Release V2.2.1

4.1. New Timeouts Added to the Java Client Library

Previously, client requests would remain pending until they completed or the connection to the database server was lost. If there was a delay caused by network issues or a procedure taking too long, the client application was never notified.

Starting with V2.2.1, both the client procedure call and the database connection itself can and will timeout. Timeouts help the client application recognize and respond to performance or network issues that are not

explicit errors. When a timeout occurs, the client application is notified through an error status returned by the procedure call and, optionally, invocation of a client status listener.

Note that the addition of timeouts is a change to previous behavior. Stored procedure calls that previously waited indefinitely will now time out after two minutes, by default. To emulate previous behavior, you can disable either or both timeout periods by setting the timeout value to zero using the `setConnectionResponseTimeout` and `setProcedureCallTimeout` methods on the `ClientConfig` object. See the section on "Handling Errors" in the *Using VoltDB* manual for more information about how to configure and identify time outs in procedure calls and client connections.

4.2. Improvements to the Java Client Status Listener

As part of the addition of timeouts to the Java client library, the client status listener has been improved. A new listener, `lateProcedureResponse`, has been added and the parameters to the `connectionLost` listener have been extended to provide additional information. To accommodate these changes, the original status listener `ClientStatusListener` has been deprecated and replaced with `ClientStatusListenerExt` (extended). This change allows existing applications that contain custom listeners to continue to work while providing extended capability to new or modified applications. Also, the replacement of the original Java interface with an abstract class will make future extensions easier.

5. Release V2.1.3

5.1. Performance Improvements to the SQL Statement Planner

The SQL statement planner is the VoltDB component that analyzes stored procedures when you compile the project definition file or at runtime when you issue an ad hoc query. Several changes have been made to the planner to improve performance. As a result, both compiling the application catalog and processing ad hoc queries should be significantly faster than in previous versions of VoltDB. Another improvement to the planner is a restructuring and simplification of the information it creates in the `/debugoutput` folder. An explanation of how to read and interpret the execution plans created by the planner is now available in a new book, the *VoltDB Performance Guide*.

5.2. Performance Optimization for VoltOne

Performance for the VoltOne product, which is a single server database, has been optimized. Certain communication overhead used in multi-node clusters has been removed for VoltOne. As a consequence, latency for transactions — particularly synchronous procedure invocations — on a single server database should be significantly improved over previous releases.

5.3. Support for IS NULL and NOT in SQL Syntax

Support for the boolean operators `IS NULL` and `NOT` has been added to VoltDB. It is now possible to use `WHERE` conditions that test whether a column is null or not or if a boolean expression is true or not. For example, the following `SELECT` statement selects records where the column `Email` is not null.

```
SELECT Username, Email FROM Users
WHERE NOT Email IS NULL;
```

6. Release V2.1.2

6.1. Improvements to Snapshot Restore

This release provides improved performance while restoring a snapshot on a K-safe cluster. For most databases using K-safety, restore times should be noticeably shorter than in earlier versions of VoltDB.

6.2. Bug Fix to Command Logging

There is a rare but critical bug in command logging that could result in the command log snapshot being erroneously deleted. This bug has been fixed. This release is recommended for anyone using command logging.

7. Release V2.1.1

7.1. Bug fixes.

The following limitations that existed in the previous release of VoltDB (V2.1) have been resolved:

- Previously, an attempt to restore a snapshot would fail if the specified file location was not present on all nodes. Now the restore operation continues as long as the location, and associated files, exist on at least one node of the cluster.
- In previous releases, output from the export clients generated timestamps where the fractional part of the timestamp was incorrect. This has been fixed so the timestamps are correctly reported as milliseconds.

8. Release V2.1

8.1. LIMIT ... OFFSET clause added to SELECT statement.

The LIMIT clause restricts the number of records that are returned by the SELECT statement. The new OFFSET option specifies the starting point within the selected record set. For example, in the set of alphabetic characters `SELECT ... LIMIT 3 OFFSET 5` returns the characters F, G, and H. See the *Using VoltDB* manual for details.

8.2. Additional control of command logging and snapshots

Two new controls have been added to the configuration options for VoltDB databases. Command log size lets you specify the initial size (in megabytes) of the command log. Snapshot priority lets you control the relative priority of snapshots with relation to other database functions, including transactions. Both options have default values suitable for the majority of applications. However, the controls are provided for specific situations and workloads that may require fine tuning for optimal performance. See the *VoltDB Management Guide* for details.

8.3. Bug fix.

The following limitation that existed in the previous release of VoltDB (V2.0) has been resolved:

- Previously, a restore request (@SnapshotRestore) would fail if the snapshot digest file was missing from the node processing the request. Now, the restore will succeed as long as the digest file is present on at least one node of the cluster.

9. Release V2.0

9.1. Lead node moved from deployment file to command line.

Previously, you specified the identity of the leader node in the deployment file. Starting with V2.0, you specify the lead node on the command line, not in the deployment file. For existing applications, you will need to remove the leader attribute from the deployment file and add it to the command line.

9.2. Application catalog is hosted by the cluster lead node.

In previous releases, the application catalog had to be available to every node in the cluster before the cluster could start. With V2.0, the application catalog only has to be available to the leader node. Once the cluster initializes, the lead node distributes the catalog to the rest of the cluster. In other words, you only need to specify the location of the catalog when starting the lead node. (However, specifying the catalog location to other nodes is not an error, the location is simply ignored.)

Similarly, you do not need to specify the catalog when rejoining a server to an existing database cluster or restarting a cluster with the **recover** startup option. Note, however, the deployment file still must exist and be available on all nodes when starting, rejoining, or recovering.

This change does not require any modification to existing applications.

9.3. Parameters to the @UpdateApplicationCatalog changed

Similar to the changes to the location of catalogs and deployment files on startup, the parameters to the @UpdateApplicationCatalog have changed. Previously, this stored procedure accepted two strings specifying the locations of the catalog and the deployment file. Starting with V2.0, the parameters to the stored procedure are the actual contents of the files, the catalog as a byte array and the deployment file as a string.

9.4. Bug fix.

The following limitation that existed in the previous release of VoltDB (V1.3.6) has been resolved:

- Previously, if the export timestamp format was less granular than the frequency of new files, the export client could overwrite existing files. This problem is resolved.

Known Limitations

The following are known limitations to the current release of VoltDB. Workarounds are suggested where applicable. However, it is important to note that these limitations are considered temporary and are likely to be corrected in future releases of the product.

1. SQL and Stored Procedures

1.1. Two identical aggregates in a SELECT statement result in only one value being returned.

If your SELECT statement includes two aggregates of the same column, VoltDB will return only one column value as part of the result set. For example, if your SQL statement is `SELECT COUNT(ColumnA), COUNT(ColumnA) FROM MyTable`, the resulting VoltTable will have only one column value per row. The workaround is to either not request the same value twice or aggregate on different columns (for example, `SELECT COUNT(Column10), COUNT(Column2)`).

1.2. Selection expressions involving arithmetic on aggregate functions are not allowed.

In SELECT statements, the selection expression can include columns, aggregate functions (such as COUNT), or arithmetic expressions involving columns (such as `Col1 + Col2`). However, they cannot include arithmetic involving aggregate functions (such as `SELECT SUM(Price) + 2`). Using aggregate functions in an arithmetic expression results in an error when you try to compile the project definition file.

The workaround is to use the aggregate in the SELECT statement and then perform the additional arithmetic on the result set either in the stored procedure or in the client application after the transaction completes.

1.3. SELECT DISTINCT using multiple columns or expressions is not supported.

Use of SELECT DISTINCT is supported for a single column (such as `SELECT DISTINCT Price FROM Inventory`). However, using DISTINCT with multiple columns or arithmetic expressions is not currently supported. For example, the following SELECT DISTINCT statements should not be used:

```
SELECT DISTINCT Price, Discount FROM Inventory
SELECT DISTINCT (Price - Discount) FROM Inventory
```

1.4. Joins of two arbitrarily partitioned tables in a multi-partitioned stored procedure are not supported

Attempts to join two (or more) distinctly partitioned tables in a multi-partitioned procedure will result in an error. You can join one partitioned table and multiple replicated tables, or you can join two partitioned tables where both tables have the same partitioning column and are joined on equality of that column. But joining two partitioned tables on non-partition columns or using a range of values is not supported.

1.5. SELECT ... FROM cannot join a table to itself.

If a SELECT statement lists the same table twice (such as SELECT ... FROM Employee AS A, Employee AS B) VoltDB will generate an error at compile time.

1.6. Do not use assertions in VoltDB stored procedures.

VoltDB currently intercepts assertions as part of its handling of stored procedures. Attempts to use assertions in stored procedures for debugging or to find programmatic errors will not work as expected.

1.7. It is possible to define an index on an export-only table, but no index is created.

As part of the database schema, it is possible to assign a primary key or index as part of a table definition, even if that table is then defined as export-only as part of the project definition file. Since export-only tables are write-only (you cannot use them in SELECT statements), the index has no purpose. VoltDB silently ignores the index definition at compile time.

2. VoltDB Compiler and Project Definition File

2.1. In the project definition file, the <database> element, if named, must be named "database".

When creating the project definition file, the database name attribute, if specified must be "database". For example <database name="database">. Note, however, that the name attribute is optional.

3. Client Interfaces

3.1. Avoid using decimal datatypes with the C++ client interface on 32-bit platforms.

There is a problem with how the math library used to build the C++ client library handles large decimal values on 32-bit operating systems. As a result, the C++ library cannot serialize and pass Decimal datatypes reliably on these systems.

Note that the C++ client interface *can* send and receive Decimal values properly on 64-bit platforms.

4. Runtime Issues

4.1. Partially removing snapshot files from the database servers can cause recovery to fail.

To ensure proper recovery on startup, either from command logs or the last database snapshot, make sure all snapshot files — or at least complete subsets of the snapshot files — are available on the nodes of the cluster. If you delete or move snapshot files (for example, copying all snapshot files to a single node) be sure to keep all of the files for each node together. Do not selectively delete or move individual files or else the recovery may fail.

Implementation Notes

The following notes provide details concerning how certain VoltDB features operate. The behavior is not considered incorrect. However, this information can be important when using specific components of the VoltDB product.

1. SQL

1.1. Do not use UPDATE to change the value of a partitioning column

For partitioned tables, the value of the column used to partition the table determines what partition the row belongs to. If you use UPDATE to change this value and the new value belongs in a different partition, the UPDATE request will fail and the stored procedure will be rolled back.

Updating the partition column value may or may not cause the record to be repartitioned (depending on the old and new values). However, since you cannot determine if the update will succeed or fail, you should not use UPDATE to change the value of partitioning columns.

The workaround, if you must change the value of the partitioning column, is to use both a DELETE and an INSERT statement to explicitly remove and then re-insert the desired rows.

- 1.2. Certain SQL syntax errors result in the error message *"user lacks privilege or object not found"* when compiling the runtime catalog.

If you refer to a table or column name that does not exist, the VoltDB compiler issues the error message *"user lacks privilege or object not found"*. This can happen, for example, if you misspell a table or column name.

Another situation where this occurs is if you mistakenly use double quotation marks to enclose a string literal (such as `WHERE ColumnA = "True"`). ANSI SQL requires single quotes for string literals and reserves double quotes for object names. In the preceding example, VoltDB interprets "True" as an object name, cannot resolve it, and issues the "user lacks privilege" error.

The workaround is, if you receive this error, to look for misspelled table or columns names or string literals delimited by double quotes in the offending SQL statement.

2. Runtime

- 2.1. File Descriptor Limits

VoltDB opens a file descriptor for every client connection to the database. In normal operation, this use of file descriptors is transparent to the user. However, if there are an inordinate number of concurrent client connections, or clients open and close many connections in rapid succession, it is possible for VoltDB to exceed the process limit on file descriptors. When this happens, new connections may be rejected or other disk-based activities (such as snapshotting) may be disrupted.

In environments where there are likely to be an extremely large number of connections, you should consider increasing the operating system's per-process limit on file descriptors.

3. Logging

- 3.1. All logging messages reported by the VoltDB server are timestamped using GMT (Greenwich Mean Time).

This is not a problem when looking at VoltDB logs separately. However, you should be aware of this distinction when integrating logging of VoltDB with logging of other system components that use the local time zone (rather than GMT). You may want to convert one or the other log streams so the time zones match.

- 3.2. To simplify logging, a file has been added to the distribution listing all of the VoltDB logging categories.

The file `voltddb/log4j.xml` lists all of the VoltDB-specific logging categories. It also serves as a useful logging schema. The sample applications and the VoltDB shell commands use this file to configure logging and it is recommended for new application development.