VoltDB Release Notes

V15.3.0

June 30, 2026 (updated July 21, 2026)

This document provides information about known issues and limitations to the current release of VoltDB. If you encounter any problems not listed below, please be sure to report them to support@voltactivedata.com. Thank you.

Upgrading From Older Versions

The process for upgrading from the recent versions of VoltDB is as follows:

1. Shutdown the database, creating a final snapshot (using **voltadmin shutdown --save**).

2. Upgrade the VoltDB software.

3. Restart the database (using **voltdb start**).

For Kubernetes, see the section on "Upgrading the VoltDB Software and Helm Charts" in the *VoltDB Kubernetes Administrator's Guide* . For DR clusters, see the section on "Upgrading VoltDB Software" in the *VoltDB Administrator's Guide* for special considerations related to DR upgrades. If you are upgrading from versions before V6.8, see the section on "Upgrading Older Versions of VoltDB Manually" in the same manual.

Finally, for all customers upgrading from earlier versions of VoltDB, please be sure to read the *Volt Upgrade Guide* , paying special attention to the notes for your current and subsequent releases, including *V8*, *V10*, *V11*, *V12*, *V13*, and V14.

Changes Since the Last Release

Users of previous versions of VoltDB should take note of the following changes that might impact their existing applications. See the *Volt Operator Release Notes* for changes specific to the use of VoltDB on the Kubernetes platform.

1. Release V15.3.0 (June 30, 2026, (updated July 21, 2026)

V15.3.x is the long-term support (LTS) release of VoltDB version 15. See the *Volt Upgrade Guide* for more information about the LTS release process and its benefits.

1.1. Recent Changes

- Previously, clock skew measurement was a one-way test that could be impacted by network latency, JVM garbage collection pauses, and scheduling delay. The resulting value could over-report the skew and occasionally trigger spurious "clock skew too high" warnings — or, in rare cases — fail a node join even when the

clocks were well synchronized. This measurement now corrects for any network round-trip delay. During an in-service upgrade to the new method, all nodes fall back to the previous measurement behavior until the upgrade is complete.

- You can now change the metrics collection interval on a running cluster without restarting it. Previously, any change to the metrics configuration required a cluster restart to take effect. Now, when you modify the interval attribute of the `metrics` element and apply the new configuration with `voltadmin update` or `set` command, the collection schedule is adjusted in place and reported by the internal metrics gauge immediately. Note that enabling or disabling metrics, or changing the buffer size, still requires a restart.
- In version 14.2.0 the snapshot buffer size was increased to 1.75MB to reduce the time required to complete a snapshot. However, the time each individual snapshot task took to process the larger buffer caused a noticeable increase in the latency of user transactions during the snapshot. Consequently, the buffer size has been reset to its original size (512K) to achieve a balance between transaction latency and overall snapshot duration.
- Queue prioritization has been deprecated and this feature will be removed from the product in a future major release. This includes per-transaction prioritization in the Client2 API as well as queue prioritization for features such as XCDR and snapshots. Note that two other methods for configuring the priority of snapshot tasks — snapshot priority and autotuning — are not affected by the deprecation of queue priorities.
- VoltDB now supports Multus and similar services that provide access to multiple network interfaces for Kubernetes pods. VoltDB can advertise separate addresses for client and XDCR traffic, configured per connection type (by interface name or CIDR) through the new `network` block in the Volt Operator's Helm chart — and optionally bind each listener to its interface. This feature requires Volt Operator 3.18.0 or later as well as support in the associated VoltDB software version.
- Messages reporting that a procedure or query is taking a long time to execute were previously always logged at the informational (INFO) level. These messages are now raised to warning (WARN) once the operation has been running for more than two seconds, making prolonged operations easier to detect with monitoring tools.
- The voltadmin DR DROP command has been deprecated and will be removed from the product in a future major release. The recommended replacement for removing a cluster from an XDCR relationship is to issue the DR RESET command from all of the other clusters in the XDCR quorum.
- Custom SQL functions (also known as user-defined functions or UDFs) have been deprecated and this feature will be removed in a future major release. Although functionally complete, UDFs never met their performance goal, especially in combination with other existing Volt capabilities. The deprecation affects the SQL DDL statements, CREATE FUNCTION, CREATE AGGREGATE FUNCTION, and DROP FUNCTION.

1.2. Security Updates

The following high and critical CVEs were resolved:

CVE-2024-31141
CVE-2025-27817
CVE-2026-2332
CVE-2026-34477
CVE-2026-34478
CVE-2026-34480
CVE-2026-35554
CVE-2026-42583
CVE-2026-42584
CVE-2026-42587
CVE-2026-44249

CVE-2026-45416
CVE-2026-54512
CVE-2026-54513
CVE-2026-54514
CVE-2026-54516
CVE-2026-54517
CVE-2026-54518

1.3. Additional Improvements

The following limitations in previous versions have been resolved:

- Previously, an application using the Volt Java client (Client2) could encounter a NullPointerException if it requested client statistics or attempted to open a connection after the client had been closed. This issue has been resolved and these operations now complete cleanly.

- Previously, after selecting a different server from the server list in the Volt Management Center (VMC), the CPU and memory monitoring charts could continue to show the previously selected server's data, making the per-server graphs appear identical. This issue has been resolved. VMC now keeps a separate, continuously updated history for every server, so switching servers immediately shows that server's own CPU and memory history without gaps.

- Previously, different nodes in a cluster could return rows in a different order if the query involved ordering by a non-unique index with duplicate key values. This issue affected physical row order only; no data was lost or returned incorrectly. This issue has been resolved and query results ordered by non-unique indexes are returned in a deterministic way by all nodes in the cluster.

- Previously, the Volt Management Center (VMC) did not always change the information displayed after selecting a different server from the server list. This issue has been resolved.

- In certain situations, performing a DR RESET would fail to restart the XDCR producer for the reset cluster. So the cluster could receive XDCR transactions from other clusters, but would not replicate its own transactions to the other participating clusters. This issue has been resolved.

- In certain rare cases, during a node shut down, the command log process could throw an alarming although ultimately harmless error that the Java printStackTrace method could not be invoked. This issue has been resolved.

- There is a rare case where, during an in-service upgrade if multi-partition transactions are being processed by an XDCR consumer while the partition leader is being moved from one node to another, the transaction might fail with the error NoSuchElementException, breaking replication. This issue has been resolved.

- Previously, collecting the IOSTATS and TLS statistics selectors required the network threads to assemble per-connection counters on demand, blocking the statistics request until each network thread responded. On a busy node with many client connections and frequent statistics polling, these requests could time out — reporting "timed out retrieving stats from network thread" in the log file and returning an error to the client. This issue has been resolved. The network threads now publish these counters to a cache at a regular interval, which the statistics request reads, eliminating the wait for the network threads. IOSTATS and TLS statistics collection no longer times out under load.

- In the database configuration, the memory compaction interval must be greater than zero. Previously, setting the compaction value to zero caused a fatal exception. This restriction is now enforced and reports a meaningful error.

- In certain rare cases where TLS/SSL is enabled, a race condition could cause a network thread to stall, resulting in a connection timeout when using sqlcmd. The issue has been resolved.

- In the Volt Management Center (VMC) sizing worksheet, changing the number of nodes or the K-factor did not affect the per-node memory estimates. This issue has been resolved and the estimates now correctly account for these settings.

- It was possible, during an in-service upgrade, for the node being upgraded to fail with a bogus error stating that migrating partition leadership was blocked by an elastic operation. This issue has been resolved.

- There is a rare condition where, under an extremely heavy XDCR workload, a DR consumer could stall while processing a multi-partition transaction, resulting in the node no longer consuming binary logs. This is another instance, although even less common, of a similar bug addressed in 13.3.8. This issue has been resolved.

- In certain situations during an in-service upgrade on Kubernetes, if the Volt operator starts to upgrade a node, but then detects an error, and aborts the upgrade before the node reboots, the cluster could end up in an unbalanced state. The issue is that the operator does not cancel the "stop node" operation on the node currently being upgraded, so it never returns partition leadership to that node, or redirects client transactions back to the node. This issue has been resolved. However, the fix requires updates to VoltDB (this release), the Volt Java client, and the Volt Operator (version 3.16.0). It also means that the fix does not take effect until performing an in-service upgrade from the new version to a subsequent version.

- If a Java program created multiple instances of the VoltBulkLoader class, the success callback was not invoked for all instances. This issue has been resolved.

2. Release V15.2.0 (March 31, 2026)

2.1. Recent Changes

- Volt Active Data now offers a Model Content Protocol (MCP) Server for VoltDB. See the Using VoltDB guide for more information and contact your Volt sales representative if you are interested.

- VoltDB now supports Kubernetes up through version 1.35.

- The @Statistics INITIATOR output includes a new column, FORWARDED, that reports the number of transactions that are forwarded from the node that receives the request to a different node for coordination.

- Volt Active Data now provides AI skills that help design, implement, and test VoltDB and VoltSP applications. See the skills repository at https://github.com/VoltDB/volt-skills for code and the chapter on integrating Volt with AI Agents in the Using VoltDB guide for more information.

- Previously, client connections were periodically notified of cluster topology changes. Now client connections are notified immediately about partition leader migrations.

- A rejoining node did not accept client transactions until the rejoin finished and a truncation snapshot was completed. As a result, client applications could see elevated latencies for this period. This has changed. A rejoining node now accepts transactions as soon as the rejoin operation is complete. It no longer waits for the truncation snapshot to finish.

2.2. Security Updates

The following high and critical CVEs were resolved:

CVE-2025-50181
CVE-2025-50182
CVE-2025-66418
CVE-2025-66471

CVE-2025-67735
CVE-2025-68161

2.3. Additional Improvements

The following limitations in previous versions have been resolved:

- In rare circumstances, a Volt node shutting down with a FATAL error could exit before flushing command log buffers to disk. While the number of transactions held in memory was generally small, all exit conditions now write to disk before the process stops.
- Extremely heavy ad hoc workloads could expose a memory leak, causing excessive heap usage. This issue has been resolved.
- VMC session handling has been changed to avoid creating unneeded session objects when VMC cannot connect to VoltDB.
- Under certain rare circumstances the MpInitiator stack could overflow when loading stored procedure definitions. This issue has been resolved. In addition, the error messages associated with procedure loading have been improved.

3. Release V15.1.0 (December 29, 2025, updated March 30, 2026)

3.1. Recent Changes

- VoltDB V15.1.0 adds support for a new datatype, BOOLEAN, with allowable values of TRUE, FALSE, and NULL.
- The HTTP/JSON programming interface returns a session cookie that lets the calling program reuse the client connection. A new section in the chapter on Using Other Programming Languages in the Using VoltDB guide explains how to use this cookie to manage client connections for applications that make frequent calls to the HTTP/JSON API.
- The file descriptor count in the result of the LIMITS and LIVECLIENTS statistics now reflects the current in-use count, rather than the count seen in the most recent check, which could be up to 10 minutes before the stats call.
- The output from @SystemInformation DEPLOYMENT now contains a row for the TLS/SSL settings from the deployment.
- VoltDB has been tested and verified to run on OpenShift 4.20.
- VoltDB no longer utilizes custom Guava code. Instead, it relies on the standard Guava library and corresponding JAR file. This means that when building your VoltDB applications, you must make sure the Guava JAR file — which is provided in the lib/ directory as part of the VoltDB kit — is in your classpath.

3.2. Security Updates

The following high and critical CVEs were resolved:

CVE-2025-59375

3.3. Additional Improvements

The following limitations in previous versions have been resolved:

- There was a slow heap memory leak in the metrics/statistics tracking of snapshots. This issue has been resolved.

- If, during a rejoin, the rejoin snapshot fails for any reason, all subsequent attempts to rejoin the cluster, or to initialize a snapshot on the cluster, will fail. When this happens, the subsequent failures were reported in the log with the message "Unable to remove outdated rejoin snapshot request." This issue has been resolved.

- In rare cases, if multiple read-only multi-partition transactions are executing simultaneously during a cluster topology change (that is, a node failing or rejoining), they could trigger a race condition, causing a deadlock. This issue has been resolved.

- Under certain rare conditions during cluster restart, it was possible for a multi-partition transaction being replayed from the command log to generate a null pointer exception. This issue has been resolved.

- In recent releases, the Volt Management Center (VMC) charts for CPU and memory usage did not display any data. This issue has been resolved.

- In rare cases, if a node fails, then fails again while rejoining the cluster but before receiving the synchronization snapshot, any further attempts to rejoin would also fail. This issue has been resolved.

- Under certain rare circumstances the MpInitiator stack could overflow when loading stored procedure definitions. This issue has been resolved.

- In certain situations, if more than one Volt cluster was started within a single Kubernetes namespace, the second or subsequent Volt clusters could be started with incorrect configuration values for properties such as XDCR, placement groups, and topics. This issue has been resolved.

4. Release V15.0.0 (September 29, 2025)

4.1. Recent Changes

- Internal-only methods of GeographyValue and GeographyPointValue, deprecated in Volt 7.8, have been deleted.

- VoltDB supports Kafka 0.10.2 or later for Kafka export, import, and topics and has been tested up through Kafka 3.9.1.

- Due to the change in YAML syntax between VoltDB V14 and V15, there are cases where the converted database configuration will attempt to trigger a cluster restart, although it is not immediately necessary. To avoid an unexpected restart during the upgrade to V15 on Kubernetes, be sure to set the property "cluster.clusterSpec.allowRestartDuringUpdate" to false when initiating the upgrade.

- The @UpdateApplicationCatalog system procedure was deprecated in V14 and is now no longer accessible from the sqlcmd command line. Please use the sqlcmd directives "load classes" and "remove classes" to manage stored procedures and the voltadmin "get", "set", and "update" commands to update the database configuration.

- The @QueryStats system procedure and associated querystats directive in sqlcmd were deprecated in V14 and have now been removed.

- There was an issue where compaction was inefficient for tables with indexes containing string columns. The compaction algorithm has been changed to correct this situation. Specifically, the maximum relocation count for these tables is now limited to 1 block, regardless of the overall configuration setting. In addition, compaction will only occur if the fragmentation threshold reaches 60% instead of 95%.

- The Volt server no longer supports the less secure SHA-1 password hashing. Only SHA-256 is supported. SHA-256 has been the default for clients for several years now.

- VoltDB V15 adds a new SQL datatype, DATE, that represents a date between January 1, 1583 and December 31, 9999, inclusive. In addition to support in the existing date/time functions such as DATEADD and DATEDIFF, a new function CURRENT_DATE has been added as a compliment to the existing CURRENT_TIMESTAMP function for timestamps.
- The YAML syntax for configuring the database on bare metal has been updated for consistency with the YAML syntax for Kubernetes. See the Volt Upgrade Guide for details.
- The Java, Python, Go , and C++ clients no longer support the less secure SHA-1 password hashing. Only SHA-256 is supported, which has been the default for several years. Configuration methods that allow specification of a hash scheme are deprecated.
- The original logging framework used by VoltDB, Log4J, has been replaced with Log4J2. The new framework uses properties for configuring logging rather than the previous XML syntax. If you customize the logging for your database, you will need to rewrite your Log4J configuration file. Please see the chapter on "Logging and Analyzing Activity in a VoltDB Database" in the Volt Administrator's Guide for details.

4.2. Security Updates

The following high and critical CVEs were resolved:

CVE-2025-58056

4.3. Additional Improvements

The following limitations in previous versions have been resolved:

- Previously, if a stored procedure queued more than 32,767 statements before calling voltExecuteSQL(), it exceeded the allowable limit of procedure results. However, this error condition was not caught; instead, the database reported a hash mismatch. The incorrect error message has been resolved. However, the limit on procedure results remains and the database now reports a more meaningful error.
- An attempt to set the value of a FLOAT column to '-Infinity' incorrectly converted the value to null. This has been corrected. Note that arithmetic operations that produce a result less than about -1.7E+308 may now evaluate to negative infinity instead of the previous incorrect null result. Volt does not recommend computation with infinities.
- When metrics are enabled, there was a slow memory leak in the Java heap associated with frequent ad hoc read-only multi-partition queries or frequent schema changes. This issue has been resolved.
- There was a race condition when initializing XDCR clusters. In certain rare cases, the initialization could fail when the cluster attempts to process incoming data after receiving the synchronization snapshot but before initialization is complete. This issue has been resolved.
- Under certain rare conditions, XDCR and Export tasks could interfere with each other for extended periods, often resulting in the message "The multipartition procedure {procedure-name} is taking a long time...". This issue has been resolved.
- Previously, if a DELETE statement did not contain a WHERE clause, it was treated like a TRUNCATE statement, deleting all data in the table on the current cluster but not creating the binary logs needed to synchronize with other XDCR clusters. This behavior has been corrected so that DELETE always creates the necessary logs to maintain consistency across XDCR clusters.
- In certain rare instances, particularly involving frequent multi-partition procedures, XDCR flow control could stall, resulting in the affected cluster no longer processing incoming XDCR data. This issue has been resolved.

- The Volt Helm chart did not properly handle cert-manager certificates issued by Certificate Authorities; that is, only self-signed certificates worked as expected. This issue has been resolved.
- The "voltadmin status --dr" command is meant to show the status of replication for all cooperating XDCR clusters. However, there are cases where the command cannot retrieve the information from the remote clusters -- most notably, when the remote nodes do not all use the same client and admin port numbers. Previously, when this happened, the command incorrectly displayed the current cluster's information multiple times. This issue has been resolved and the command now detects the problem and only displays information for the clusters it can reach.
- There was an issue with handling multiple certificates in a single PEM trust store. This issue has been resolved.
- Under certain rare conditions, the metrics subsystem could receive too much data internally, resulting in the illegal state exception "would overflow integer count", causing the server process to fail with the additional warning "encountered an unexpected error and will die." This issue has been resolved.
- Previously, after a node stopped and then rejoined the cluster, the "voltadmin resize --test" command would report the original host IDs of the cluster, not the new assignment for the rejoined node. This issue has been 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. Command Logging

1.1. Do not use the subfolder name "segments" for the command log snapshot directory.

VoltDB reserves the subfolder "segments" under the command log directory for storing the actual command log files. Do not add, remove, or modify any files in this directory. In particular, do not set the command log snapshot directory to a subfolder "segments" of the command log directory, or else the server will hang on startup.

2. Database Replication

2.1. Avoid bulk data operations within a single transaction when using database replication

Bulk operations, such as large deletes, inserts, or updates are possible within a single stored procedure. However, if the binary logs generated for DR are larger than 45MB, the operation will fail. To avoid this situation, it is best to break up large bulk operations into multiple, smaller transactions. A general rule of thumb is to multiply the size of the table schema by the number of affected rows. For deletes and inserts, this value should be under 45MB to avoid exceeding the DR binary log size limit. For updates, this number should be under 22.5MB (because the binary log contains both the starting and ending row values for updates).

2.2. Database replication ignores resource limits

There are a number of VoltDB features that help manage the database by constraining memory size and resource utilization. These features are extremely useful in avoiding crashes as a result of unexpected or unconstrained growth. However, these features could interfere with the normal operation of DR when passing data from one

cluster to another, especially if the two clusters are different sizes. Therefore, as a general rule of thumb, DR overrides these features in favor of maintaining synchronization between the two clusters.

Specifically, DR ignores any resource monitor limits defined in the deployment file when applying binary logs on the consumer cluster. This means, for example, if the replica database in passive DR has less memory or fewer unique partitions than the master, it is possible that applying binary logs of transactions that succeeded on the master could cause the replica to run out of memory. Note that these resource monitor limits *are* applied on any original transactions local to the cluster (for example, transactions on the master database in passive DR).

2.3. Different cluster sizes can require additional Java heap

Database Replication (DR) now supports replication across clusters of different sizes. However, if the replica cluster is smaller than the master cluster, it may require a significantly larger Java heap setting. Specifically, if the replica has fewer unique partitions than the master, each partition on the replica must manage the incoming binary logs from more partitions on the master, which places additional pressure on the Java heap.

A simple rule of thumb is that the worst case scenario could require an additional P * R * 20MB space in the Java heap , where P is the number of sites per host on the replica server and R is the ratio of unique partitions on the master to partitions on the replica. For example, if the master cluster is 5 nodes with 10 sites per host and a K factor of 1 (i.e. 25 unique partitions) and the replica cluster is 3 nodes with 8 sites per host and a K factor of 1 (12 unique partitions), the Java heap on the replica cluster may require approximately 320MB of additional space in the heap:

Sites-per-host * master/replace ratio * 20MB
8 * 25/12 * 20 = ~ 320MB

An alternative is to reduce the size of the DR buffers on the master cluster by setting the DR_MEM_LIMIT Java property. For example, you can reduce the DR buffer size from the default 10MB to 5MB using the VOLTDB_OPTS environment variable before starting the master cluster.

```
$ export VOLTDB_OPTS="-DDR_MEM_LIMIT=5"

$ voltdb start
```

Changing the DR buffer limit on the master from 10MB to 5MB proportionally reduces the additional heap size needed. So in the previous example, the additional heap on the replica is reduced from 320MB to 160MB.

2.4. The **voltadmin status --dr** command does not work if clusters use different client ports

The **voltadmin status --dr** command provides real-time status on the state of database replication (DR). Normally, this includes the status of the current cluster as well as other clusters in the DR environment. (For example, both the master and replica in passive DR or all clusters in XDCR.) However, if the clusters are configured to use different port numbers for the client port, VoltDB cannot reach the other clusters and the command hangs until it times out waiting for a response from the other clusters.

3. Cross Datacenter Replication (XDCR)

3.1. Avoid replicating tables without a unique index.

Part of the replication process for XDCR is to verify that the record's starting and ending states match on both clusters, otherwise known as *conflict resolution*. To do that, XDCR must find the record first. Finding uniquely indexed records is efficient; finding non-unique records is not and can impact overall database performance.

To make you aware of possible performance impact, VoltDB issues a warning if you declare a table as a DR table and it does not have a unique index.

3.2. When starting XDCR for the first time, only one database can contain data.

You cannot start XDCR if both databases already have data in the DR tables. Only one of the two participating databases can have preexisting data when DR starts for the first time.

3.3. During the initial synchronization of existing data, the receiving database is paused.

When starting XDCR for the first time, where one database already contains data, a snapshot of that data is sent to the other database. While receiving and processing that snapshot, the receiving database is paused. That is, it is in read-only mode. Once the snapshot is completed and the two database are synchronized, the receiving database is automatically unpaused, resuming normal read/write operations.

3.4. A large number of multi-partition write transactions may interfere with the ability to restart XDCR after a cluster stops and recovers.

Normally, XDCR will automatically restart where it left off after one of the clusters stops and recovers from its command logs (using the **voltdb recover** command). However, if the workload is predominantly multi-partition write transactions, a failed cluster may not be able to restart XDCR after it recovers. In this case, XDCR must be restarted from scratch, using the content from one of the clusters as the source for synchronizing and recreating the other cluster (using the **voltdb create --force** command) without any content in the DR tables.

3.5. Avoid using TRUNCATE TABLE in XDCR environments.

TRUNCATE TABLE is optimized to delete all data from a table rather than deleting tuples row by row. This means that the binary log does not identify which rows are deleted. As a consequence, a TRUNCATE TABLE statement and a simultaneous write operation to the same table can produce a conflict that the XDCR clusters cannot detect or report in the conflict log.

Therefore, do not use TRUNCATE TABLE with XDCR. Instead, explicitly delete all rows with a DELETE statement and a filter. For example, `DELETE * FROM table WHERE column=column` ensures all deleted rows are identified in the binary log and any conflicts are accurately reported. Note that `DELETE FROM table` without a WHERE clause is *not* sufficient, since its execution plan is optimized to equate to TRUNCATE TABLE.

3.6. Use of the VoltProcedure.getUniqueId method is unique to a cluster, not across clusters.

VoltDB provides a way to generate a deterministically unique ID within a stored procedure using the getUniqueId method. This method guarantees uniqueness *within the current cluster*. However, the method could generate the same ID on two distinct database clusters. Consequently, when using XDCR, you should combine the return values of VoltProcedure.getUniqueId with VoltProcedure.getClusterId, which returns the current cluster's unique DR ID, to generate IDs that are unique across all clusters in your environment.

3.7. Multi-cluster XDCR environments require command logging.

In an XDCR environment involving three or more clusters, command logging is used to ensure the durability of the XDCR "conversations" between clusters. If not, when a cluster stops, the remaining clusters can be at different stages of their conversation with the downed cluster, resulting in divergence.

For example, assume there are three clusters (A, B, and C) and cluster B is processing binary logs faster than cluster C. If cluster A stops, cluster B will have more binary logs from A than C has. You can think of B being "ahead" of C. With command logging enabled, when cluster A restarts, it will continue its XDCR conversations and cluster C will catch up with the missing binary logs. However, without command logging, when A stops, it must restart from scratch. There is no mechanism for resolving the difference in binary logs processed by clusters B and C before the failure.

This is why command logging is required to ensure the durability of XDCR conversations in a multi-cluster (that is , three or more) XDCR environment. The alternative, if not using command logging, is to restart all but one of the remaining clusters to ensure they are starting from the same base point.

3.8. Do not use **voltadmin dr reset** to remove an XDCR cluster that is still running

There are two ways to remove an cluster from an XDCR relationship: you can use **voltadmin drop** on a running cluster to remove it from the XDCR network, or you can use **voltadmin dr reset** from the remaining clusters to remove a cluster that is no longer available. But you *should not* use **voltadmin dr reset** to remove a cluster that is still running and connected to the network. Resetting a running cluster will break DR for that cluster, but will result in errors on the remaining clusters and leave their DR queues for the reset cluster in an ambiguous state. Ultimately, this can result in the removed cluster not being able to rejoin the XDCR network at a later time.

4. TTL

4.1. Use of TTL (time to live) with replicated tables and Database Replication (DR) can result in increased DR activity.

TTL, or time to live, is a feature that automatically deletes old records based on a timestamp or integer column. For replicated tables, the process of checking whether records need to be deleted is performed as a write transaction — even if no rows are deleted. As a consequence, any replicated DR table with TTL defined will generate frequent DR log entries, whether there are any changes or not, significantly increasing DR traffic.

Because of the possible performance impact this behavior can have on the database, use of TTL with replicated tables and DR is not recommended at this time.

5. Export

5.1. Synchronous export in Kafka can use up all available file descriptors and crash the database.

A bug in the Apache Kafka client can result in file descriptors being allocated but not released if the producer.type attribute is set to "sync" (which is the default). The consequence is that the system eventually runs out of file descriptors and the VoltDB server process will crash.

Until this bug is fixed, use of synchronous Kafka export is not recommended. The workaround is to set the Kafka producer.type attribute to "async" using the VoltDB export properties.

6. Import

6.1. Data may be lost if a Kafka broker stops during import.

If, while Kafka import is enabled, the Kafka broker that VoltDB is connected to stops (for example, if the server crashes or is taken down for maintenance), some messages may be lost between Kafka and VoltDB. To ensure no data is lost, we recommend you disable VoltDB import before taking down the associated Kafka broker. You can then re-enable import after the Kafka broker comes back online.

6.2. Kafka import can lose data if multiple nodes stop in succession.

There is an issue with the Kafka importer where, if multiple nodes in the cluster fail and restart, the importer can lose track of some of the data that was being processed when the nodes failed. Normally, these pending imports are replayed properly on restart. But if multiple nodes fail, it is possible for some in-flight imports to get lost. This issue will be addressed in an upcoming release.

7. SQL and Stored Procedures

7.1. Comments containing unmatched single quotes in multi-line statements can produce unexpected results.

When entering a multi-line statement at the sqlcmd prompt, if a line ends in a comment (indicated by two hyphens) and the comment contains an unmatched single quote character, the following lines of input are not

interpreted correctly. Specifically, the comment is incorrectly interpreted as continuing until the next single quote character or a closing semi-colon is read. This is most likely to happen when reading in a schema file containing comments. This issue is specific to the sqlcmd utility.

A fix for this condition is planned for an upcoming point release

7.2. 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.

7.3. The UPPER() and LOWER() functions currently convert ASCII characters only.

The UPPER() and LOWER() functions return a string converted to all uppercase or all lowercase letters, respectively. However, for the initial release, these functions only operate on characters in the ASCII character set. Other case-sensitive UTF-8 characters in the string are returned unchanged. Support for all case-sensitive UTF-8 characters will be included in a future release.

8. Client Interfaces

8.1. 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.

9. SNMP

9.1. Enabling SNMP traps can slow down database startup.

Enabling SNMP can take up to 2 minutes to complete. This delay does not always occur and can vary in length. If SNMP is enabled when the database server starts, the delay occurs after the server logs the message "Initializing SNMP" and before it attempts to connect to the cluster. If you enable SNMP while the database is running, the delay can occur when you issue the **voltadmin update** command or modify the setting in the VoltDB Management Center Admin tab. This issue results from a Java constraint related to secure random numbers used by the SNMP library.

10. TLS/SSL

10.1. Using Commercial TLS/SSL certificates with the command line utilities.

The command line utilities, such as **sqlcmd**, **voltadmin**, and the data loaders, have an `--ssl` flag to specify the truststore for user-created certificates. However, if you use commercial certificates, there is no truststore. In that case, you need to specify an empty string to the `--ssl` command qualifier to access the database. For example:

```
$ sqlcmd --ssl=""
```

10.2. Bypassing TLS/SSL verification with **voltadmin**.

When TLS/SSL is enabled, the command line utility **voltadmin** assumes you want to verify the authenticity of the database server. If you are using user-created certificates, you must provide the associated truststore on the command line using the `--ssl` flag. However, if you trust the server and do not need to verify the certificate, you can use the syntax `--ssl=nocheck` to bypass the verification. For example:

```
$ voltadmin --ssl=nocheck
```

This is particularly important if you exec into the shell of a Kubernetes pod running Volt with TLS/SSL enabled, since the Volt Docker image includes a slimmed-down set of command line tools not designed for accessing databases outside of the pod.

11. Kubernetes

11.1. Shutting down a VoltDB cluster by setting `cluster.clusterSpec.replicas` to zero might not stop the associated pods.

Shutting down a VoltDB cluster by specifying a replica count of zero should shut down the cluster and remove the pods on which it ran. However, on very rare occasions Kubernetes does not delete the pods. As a result, the cluster cannot be restarted. This is an issue with Kubernetes. The workaround is to manually delete the pods before restarting the cluster.

11.2. Specifying invalid or misconfigured volumes in `cluster.clusterSpec.additionalVolumes` interferes with Kubernetes starting the VoltDB cluster.

The property `cluster.clusterSpec.additionalVolumes` lets you specify additional resources to include in the server classpath. However, if you specify an invalid or misconfigured volume, Helm will not be able to start the cluster and the process will stall.

11.3. Using binary data with the Helm --set-file argument can cause problems when later upgrading the cluster.

The Helm `--set-file` argument lets you set the value of a property as the contents of a file. However, if the contents of the file are binary, they can become corrupted if you try to resize the cluster with the **helm upgrade** command, using the `--reuse-values` argument. For example, this can happen if you use `--set-file` to assign a JAR file of stored procedure classes to the `cluster.config.classes` property.

This is a known issue for Kubernetes and Helm. The workaround is either to explicitly include the `--set-file` argument again on the **helm upgrade** command. Or you can include the content through a different mechanism. For example, you can include class files by mounting them on a separate volume that you then include with the `cluster.clusterSpec.additionalVolumes` property.

12. User-Defined Functions (UDF)

12.1. Certain UDF aggregate functions can result in SQL run-time error

Under certain circumstances, if a user-defined aggregate function (UDF) uses both numeric and character data, attempts to execute the UDF can result in a run-time SQL error. The workaround is to rewrite the UDF as a stored procedure.

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. Networking

1.1. Support for IPv6 addresses

VoltDB works in IPv4, IPv6, and mixed network environments. Although the examples in the documentation use IPv4 addresses, you can use IPv6 when configuring your database, making connections through applications, or using the VoltDB command line utilities, such as **voltdb** and **voltadmin**. When specifying IPv6 addresses on the command line or in the configuration file, be sure to enclose the address in square brackets. If you are specifying both an IPv6 address and port number, put the colon and port number *after* the square brackets. For example:

```
voltadmin status --host=[2001:db8:85a3::8a2e:370:7334]:21211
```

1.2. Issues with Jumbo Frames

There have been reports that VoltDB does not operate correctly when the networking environment is configured to use *jumbo frames*. The symptoms are that TCP packets with MTU>9000 are dropped. However, this is not a Volt-specific issue. When configuring the network to use jumbo frames, it is crucial to thoroughly test the network to guarantee that TCP packets are not fragmented or have their segments reordered during reassembly. If not, there is a danger of lost packets in the network layer, unrelated to the specific application involved.

2. XDCR

2.1. Upgrading existing XDCR clusters for Dynamic Schema Change

Volt Active Data V12.3 introduces a new feature, dynamic schema change for XDCR clusters. To use this feature, clusters must be configured to enable the feature and must be using the latest DR protocol. However, existing clusters that upgrade from earlier versions will not automatically use the new protocol. Instead, you must explicitly upgrade the DR protocol using the **voltadmin dr protocol --update** command.

First, to use dynamic schema change you must enable the feature in the configuration file. This must be done when you initialize the cluster which, for existing XDCR clusters, is easiest to when performing the version upgrade. To upgrade XDCR clusters, you must drop the cluster from the XDCR environment, upgrade the software, then reinitialize and restart the cluster. While reinitializing the cluster, add the `<schemachange enabled="true"/>` element to the configuration file. For example:

```
<deployment>
  <dr id="1" role="xcdr">
    <schemachange enabled="true"/>
    <connection source="paris.mycompany.com,rome.mycompany.com"/>
  </dr>
</deployment>
```

Similarly, for Kubernetes, the YAML would be:

```
cluster:
  config:
    deployment:
      dr:
        id: 1
        role: xdcr
        schemachange:
          enabled: true
        connection:
            source: paris.mycompany.com,rome.mycompany.com
```

Once all of the clusters are upgraded to the appropriate software version, you can upgrade the DR protocol. When used by itself, the **voltadmin dr protocol** command displays information about what versions of the DR protocol the XDCR clusters support and which version they are using. When the XDCR relationship starts, the

clusters use the highest supported protocol. However, when an existing XDCR group is upgraded, they do not automatically upgrade the originally agreed-upon protocol. In this case, you must go to each cluster and issue the **voltadmin dr protocol --update** command to use the highest protocol that all the clusters can support. Once you issue the **voltadmin dr protocol --update** command on all of the clusters, you are then ready to perform dynamic schema changes on your XDCR environment.

3. Volt Management Center

3.1. Schema updates clear the stored procedure data table in the Management Center Monitor section

Any time the database schema or stored procedures are changed, the data table showing stored procedure statistics at the bottom of the Monitor section of the VoltDB Management Center get reset. As soon as new invocations of the stored procedures occur, the statistics table will show new values based on performance after the schema update. Until invocations occur, the procedure table is blank.

3.2. TLS/SSL for the HTTPD port and Volt Management Console (VMC)

Starting with version 14.2, VoltDB uses PEM files to certify and authenticate the database servers, including interactions between the VMC service and the servers. However, the Jetty service that provides the HTTP interface to VMC for web browsers and the JSON API does *not* support PEM format. Therefore, to enable encryption to VMC from client browsers and apps, you should use either JKS or PKCS12 formatted key and trust stores. You can either convert an existing PEM file to JKS or PKCS12 format using the **openssl** utility or see the VoltDB V13 documentation for instructions on creating user-defined certificates using the **keytool** utility. (To create PKCS12 rather than JKS files, simply replace the `-storetype jks` qualifier with `-storetype pkcs12` in the examples.)

On bare metal, use the `--publicssl` qualifier when starting VMC to specify a properties file identifying the PKS-formatted truststore and password. For example:

```
$ cat vmcpublic.properties
trustStore=/etc/ssl/local/mydb.truststore
trustStorePassword=mypasswd
$ vmc --publicssl=vmcpublic.properties
```

On Kubernetes, see the instructions for enabling TLS/SSL for VMC in the *Volt Kubernetes Administrator's Guide* .

4. SQL

4.1. You cannot partition a table on a column defined as ASSUMEUNIQUE.

The ASSUMEUNIQUE attribute is designed for identifying columns in partitioned tables where the column values are known to be unique but the table is not partitioned on that column, so VoltDB cannot verify complete uniqueness across the database. Using interactive DDL, you can create a table with a column marked as ASSUMEUNIQUE, but if you try to partition the table on the ASSUMEUNIQUE column, you receive an error. The solution is to drop and add the column using the UNIQUE attribute instead of ASSUMEUNIQUE.

4.2. Adding or dropping column constraints (UNIQUE or ASSUMEUNIQUE) is not supported by the ALTER TABLE ALTER COLUMN statement.

You cannot add or remove a column constraint such as UNIQUE or ASSUMEUNIQUE using the ALTER TABLE ALTER COLUMN statement. Instead to add or remove such constraints, you must first drop then add the modified column. For example:

```
ALTER TABLE employee DROP COLUMN empID;
```

```
ALTER TABLE employee ADD COLUMN empID INTEGER UNIQUE;
```

4.3. 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.

4.4. Ambiguous column references no longer allowed.

Starting with VoltDB 6.0, ambiguous column references are no longer allowed. For example, if both the *Customer* and *Placedorder* tables have a column named *Address*, the reference to *Address* in the following SELECT statement is ambiguous:

```
SELECT OrderNumber, Address FROM Customer, Placedorder
   . . .
```

Previously, VoltDB would select the column from the leftmost table (Customer, in this case). Ambiguous column references are no longer allowed and you must use table prefixes to disambiguate identical column names. For example, specifying the column in the preceding statement as *Customer.Address*.

A corollary to this change is that a column declared in a USING clause can now be referenced using a prefix. For example, the following statement uses the prefix *Customer.Address* to disambiguate the column selection from a possibly similarly named column belonging to the *Supplier* table:

```
SELECT OrderNumber, Vendor, Customer.Address
   FROM Customer, Placedorder Using (Address), Supplier
    . . .
```

5. Runtime

5.1. 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.

5.2. Use of Resources in JAR Files

There are two ways to access additional resources in a VoltDB database. You can place the resources in the `/lib` folder where VoltDB is installed on each server in the cluster or you can include the resource in a subfolder of a JAR file you add using the sqlcmd **LOAD CLASSES** directive. Adding resources via the `/lib` directory is useful for stable resources (such as third-party software libraries) that do not require updating. Including resources (such as XML files) in the JAR file is useful for resources that may need to be updated, as a single transaction, while the database is running.

LOAD CLASSES is used primarily to load classes associated with stored procedures and user-defined functions. However, it will also load any additional resource files included in subfolders of the JAR file. You can remove classes that are no longer needed using the **REMOVE CLASSES** directive. However, there is no explicit command for removing other resources.

Consequently, if you rename resources or move them to a different location and reload the JAR file, the database will end up having multiple copies. Over time, this could result in more and more unnecessary memory being used by the database. To remove obsolete resources, you must first reinitialize the database root directory, start a fresh database, reload the schema (including the new JAR files with only the needed resources) and then restore the data from a snapshot.

5.3. Servers with Multiple Network Interfaces

If a server has multiple network interfaces (and therefore multiple IP addresses) VoltDB will, by default, open ports on all available interfaces. You can limit the ports to an single interface in two ways:

- Specify which interface to use for internal and external ports, respectively, using the **--internalinterface** and **--externalinterface** arguments when starting the database process with the **voltdb start** command.
- For an individual port, specify the interface and port on the command line. For example **voltdb start --client=32.31.30.29:21212**.

Also, when using an IP address to reference a server with multiple interfaces in command line utilities (such as **voltadmin stop node)**, use the @SystemInformation system procedure to determine which IP address VoltDB has selected to identify the server. Otherwise, if you choose the wrong IP address, the command might fail.

5.4. Using VoltDB where the /tmp directory is noexec

On startup, VoltDB extracts certain native libraries into the /tmp directory before loading them. This works in all standard operating environments. However, in custom installations where the /tmp storage is mounted with the "noexec" option, VoltDB cannot extract the libraries and, in the past, refused to start.

For those cases where the /tmp directory is assigned on storage mounted with the ‘noexec’ option, you can assign an alternative storage for VoltDB to use for extracting and executing temporary files. This is now done automatically on Kubernetes and does not require any user intervention.

On non-Kubernetes environments, you can identify an alternative location by assigning it to `volt.tmpdir`, `org.xerial.snappy.tmpdir`, and `jna.tmpdir` in the VOLTDB_OPTS environment variable before starting the server process. The specified location must exist, must be an absolute path, and cannot be on storage mounted with the "noexec" option. For example, the following command assigns an alternate temporary directory called `/volttemp`:

```
export VOLTDB_OPTS="-Dvolt.tmpdir=/volttemp \
                    -Dorg.xerial.snappy.tempdir=/volttemp \
                    -Djna.tmpdir=/volttemp"
```

When using an alternate temporary directory, files can accumulate over time since the directory is not automatically purged on startup. So you should make sure you periodically delete old files from the directory.

5.5. Text Data and Character Sets

Volt Active Data processes and stores text data as UTF-8 encoded strings. When using the Volt Java API, the Java String datatype always stores text in Unicode. So there is no conversion necessary when passing string data to Volt stored procedures. If the application must handle data in alternate character encodings you can use existing Java functionality to convert the data to Unicode on input. For example, using the second argument to specify the character encoding when instantiating a buffered reader:

```
BufferedReader myfile = Files.newBufferedReader(filename,charset);
```

On the other hand, when users enter data interactively using an alternate character encoding, Volt automates the conversion of the character set of the current session to UTF-8 on input and output. In other words, if the user's session is localized to use a specific character set, when entering data at the **sqlcmd** prompt (for example, when executing an INSERT statement) the data is automatically converted to UTF-8 before processing. Similarly, on output (such as displaying the results of a SELECT statement) the UTF-8 data is converted to the user's localized character set.

When processing text files, such as CSV files or **sqlcmd** scripts, the command line utilities provide a `--charset` qualifier that lets you specify the character set of the input file. The `--charset` qualifier affects the FILE directive and `--file` and `--output` qualifiers for **sqlcmd** as well as the input files for the **csvloader**. Note that the user's localized session character set does *not* affect file input. If the `--charset` qualifier is not used the data is assumed to be UTF-8 by default.

Finally, which character sets are supported depends on which Java virtual machine (JVM) release you are using on your servers (for export) or client machines (for **sqlcmd** and **csvloader**). For established character sets, such as Shift_JIS or ISO-8859-1, all supported JVM releases provide support. For newer character sets, you may need a more recent release of the JVM. For example, the recent Simplified Chinese character set GB18030-2022 requires a JVM released in 2023 or later. For OpenJDK this includes the following releases:

- Java 8 — release 8u32-b05
- Java 11 — release 11.0.20+8
- Java 17 — release 17.0.8+7

5.6. Using Java 11 in Client Applications

When running a client application linked against the VoltDB Java API and using the Java 11 runtime, the application may issue warnings about an " illegal reflective access operation." The message does not impact the actual operation of the client or the database itself. You can suppress this warning using the --add-opens qualifier on the **java** command. For example:

```
$ java --add-opens=java.base/java.lang=ALL-UNNAMED  \
       --add-opens=java.base/sun.nio.ch=ALL-UNNAMED \
       --add-opens=java.base/java.net=ALL-UNNAMED   \
       --add-opens=java.base/java.nio=ALL-UNNAMED   \
       myclientapp
```

6. Kubernetes

6.1. Do not change the UID on the Kubernetes account used to run VoltDB

In the security context section of the Helm chart for VoltDB, the user account UID is set to 1001. This value is required. Do not change or override any of the following properties when configuring your database:

cluster.clusterSpec.podSecurityContext.runAsUser
cluster.clusterSpec.podSecurityContext.fsGroup
cluster.clusterSpec.securityContext.runAsUser
cluster.clusterSpec.securityContext.runAsGroup

6.2. OpenShift and Transparent Huge Pages (THP)

For production, VoltDB requires that Transparent Huge Pages (THP) are disabled because they interfere with memory-intensive applications. However, THP may be enabled on OpenShift containers and the containers

themselves not have permission to disable them. To overcome this situation, you must run the Helm chart for disabling THP from a privileged container:

```
$ helm -n kube-system install thp voltdb/transparent-hugepages \
       --set thp.securityContext.privileged=true
```

6.3. Kubernetes Compatibility

See the *Volt Kubernetes Compatibility Chart* for information on which versions of the Volt Operator and Helm charts support which version of VoltDB. See the *VoltDB Operator Release Notes* for additional information about individual releases of the VoltDB Operator.

7. Java Client API

7.1. The BulkLoader and Success and Failure Responses

The Volt data loader command line utilities (csvloader, jdbcloader, and kafkaloader) all use the generic bulkloader interface available from the Java Client API. The bulkloader lets you create your own custom bulkloader, batching and optimizing multiple insert operations by grouping single partition statements according to partition and applying them as a single transaction.

When writing a custom application using the bulkloader interface it is important to understand how the callbacks work. When the batch transaction succeeds, the success callback is called for every insert in the batch. If the batch transaction fails, the bulkloader API switches to trying each insert as a separate transaction. In this second mode of operation, the failure callback is called for any inserts that fail. However, the success callback is *not* invoked for any of the inserts that succeed individually.