Version: 5.7

Community Edition Release Notes

These release notes list any new features, enhancements, fixes, security issues and breaking changes that were made for Hazelcast Platform Community Edition.

> **NOTE**
>
> Hazelcast Platform Community Edition is available in major and minor releases only (e.g. x.0, x.1, x.2, etc.). From release 5.4, patch releases are only available for Enterprise Edition i.e. no patch releases (e.g. 5.4.1, 5.5.2) will be made available for Community Edition.

For help downloading Hazelcast Community Edition, see Install Hazelcast Community Edition.

5.7.0

Release date: 2026-05-13

This release also includes the Community Edition fixes and security updates that were delivered in the Enterprise Edition 5.6.1 maintenance release.

- Breaking changes
- Enhancements
- Fixes
- Deprecations
- Security
- Known issues
- Contributors

Breaking changes

- **Decoupled `hazelcast-spring` from Spring Boot autoconfiguration**: Removed the direct dependency on Spring Boot from `hazelcast-spring` to resolve a dependency cycle that prevented Spring Boot from upgrading to newer Hazelcast versions. As part of this change, Spring Boot autoconfiguration for exposing Hazelcast components has been moved to dedicated modules: `hazelcast-spring-boot3` and `hazelcast-spring-boot4`. Users relying on Spring Boot autoconfiguration must explicitly add the appropriate module to their dependencies to retain existing behavior.

Enhancements

- **Java 25 support**: Hazelcast Platform 5.7 supports Java 17, 21 and 25. Adding Java 25 enables customers to adopt the latest JDK with full compatibility, ensuring alignment with evolving JVM features and allowing teams to standardize on newer runtimes without impacting cluster stability.

  > **IMPORTANT**
  >
  > When running Hazelcast on Java 25, the `SecurityManager` is not functional due to ecosystem changes in the JDK. Deployments that rely on `SecurityManager`-based security policies must run on Java 17 or 21.

  > **NOTE**
  >
  > Docker users who want to run Hazelcast on an earlier Java version instead of the default Java 25 must pull a JDK-tagged image. For example, `hazelcast/hazelcast:latest-jdk21` will resolve to 5.7 built with JDK 21.

- **Dynamic Diagnostic Logs are now Generally Available**: First introduced as BETA in 5.6, Dynamic Diagnostic Logs are now GA. Operators can enable and adjust diagnostic logging at runtime without restarts, significantly improving on-demand troubleshooting and reducing the need for disruptive configuration changes in production environments.

- **Migrated Kinesis connector to AWS SDK v2**: Replaced usage of the deprecated AWS SDK v1, which reached end of support, with the v2 SDK to eliminate deprecation warnings and

ensure long-term compatibility and support; no user action is required unless relying on SDK v1-specific behavior.

- **Improved User Code Namespace support for Jet IMap sinks using EntryProcessors**: Enhanced the behavior of `Sinks.mapWithMerging`, `mapWithUpdate`, and `mapWithEntryProcessor` to correctly resolve classes from the job's User Code Namespace (UCN) during deserialization. The job namespace is now checked first, with a fallback to the IMap namespace if needed, improving compatibility with custom job resources.

- **Improved Jet backpressure metrics**: Resolved an issue where input queue sizes were reported incorrectly under backpressure, even when queues were full and jobs were stalled. Queue sizes are now updated during metric collection, providing better visibility into where backpressure occurs.

- **Improved handling of concurrent Event Journal readers**: Resolved an issue where multiple readers of the same Event Journal, especially with different filters, could block each other, causing increased latency, misleading warnings, and occasional event loss. Readers are now unparked in a scheduled manner, ensuring independent progress, more accurate diagnostics, and more reliable event delivery.

- **Introduced extensible transformation mechanism for Pipeline API**: Added a new `using(…)` extension mechanism that enables type-safe, fluent extension of Pipeline transformations. This allows both built-in and user-defined extensions to add custom transformation methods across stream, batch, and keyed stages.

- **Added persistence support for Jet job namespaces**: Namespaces used by Jet jobs are now persisted to Hot Restart storage during job startup and when updated by a running job. This ensures that namespaces survive lossless restarts, eliminating the need to re-add them.

- **Added `INITIAL_SNAPSHOT_REQUIRED` to prevent data loss on job restarts**: Introduced a new job configuration option that forces Jet jobs with stateful sources (such as map journal or Kafka) to complete an initial snapshot before processing any data. This ensures that sources depending on initialization-time state do not reinitialize inconsistently after a restart, which could otherwise lead to data loss despite `exactly_once` or `at_least_once` guarantees. When enabled, the job only transitions to `RUNNING` after the initial snapshot completes successfully.

- **Added immediate state eviction option for keyed stateful stream stages**: Introduced a new `deleteStatePredicate` parameter to the `mapStateful`, `flatMapStateful`, and `filterStateful` methods for keyed stream stages. This predicate allows the state associated with a key to be removed immediately after processing an event when the condition evaluates to true. When triggered, the state is deleted without invoking the `onEvictFn` callback, giving developers finer control over state lifecycle and memory usage in stateful stream processing.

- **Added state size metric for `mapStateful` processors**: Introduced a new `totalStates` metric that reports the number of states maintained by each `mapStateful` processor. This helps users identify jobs where state growth may lead to excessive memory usage or potential out-of-memory conditions.

- **Added MapFlush sink API for snapshot-driven persistence**: Introduced `EnterpriseSinks.mapFlushSink` to enable Jet pipelines to flush an IMap to its configured MapStore during snapshot phases. This is particularly useful for deduplication and other scenarios requiring consistent commit semantics to the backing store.

- **Improved Map and Cache Event Journal cleanup to prevent memory leaks**: Enhanced the cleanup process for event journals to prevent unbounded heap growth on backup replicas. Cleanup now runs on both primary and backup replicas before add operations. A new cluster property, `hazelcast.journal.cleanup.threshold`, was also introduced to trigger cleanup when the journal's remaining capacity drops below a configured threshold.

Fixes

- **Fixed failures and inconsistent results in `IMap.containsValue()` with TTL**: Resolved an issue where entry expiration during iteration could interfere with map traversal, causing operation failures and missed values. The fix ensures stable iteration by checking expiration instead of modifying entries during traversal, resulting in reliable results.
- **Reduced CPU spike on Near Cache with TTL**: Resolved a performance issue affecting Near Caches when TTL or max-idle was configured. Previously, the expiration task scanned all cache entries every 5 seconds, leading to excessive CPU usage for large caches with long TTLs. The fix introduces a time-bound expiration cycle that limits CPU consumption per run and resumes from the last examined entry in the next cycle if needed. This preserves the existing guarantee that expired entries are never returned, while significantly reducing unnecessary CPU load.

- **Fixed missing Near Cache invalidation on TTL expiration for Java client**: Resolved an issue where entries expired via TTL in map/cache did not trigger invalidation events for the Java client Near Cache, leading to stale data.
- **Fixed incorrect handling of `serializeKeys` in dynamically added Near Cache configurations**: Resolved an issue where `serializeKeys=true` was ignored and stored as `false`. During rolling upgrades, re-adding such configurations may fail due to this mismatch; in these cases, use `serializeKeys=false` to match the existing state.
- **Fixed validation for duplicate advanced network endpoint configurations (XML)**: Resolved an issue where XML configuration allowed multiple declarations of advanced network endpoint sections (such as `member-server`, `client-server`, REST, and memcache socket endpoint configs), even though only a single declaration is supported.
- **Fixed non-functional operation timeout metrics**: Resolved an issue where the `operation.callTimeoutCount` and `operation.operationTimeoutCount` metrics were listed in the documentation but never incremented, making them ineffective. The fix restores the increment logic for `operation.callTimeoutCount`, ensuring it correctly tracks invocation call timeouts. The `operation.operationTimeoutCount` metric has been deprecated since its functionality is already covered by `InvocationMonitor` timeout metrics and will be removed in a future major release.
- **Fixed initialization failure handling and dependency issues in `GenericMapStore`**: Improved error handling during initialization to ensure failures are correctly propagated, and removed an unnecessary `slf4j` classpath requirement. Additionally, the `columns` property now automatically trims leading and trailing whitespace.
- **Fixed endpoint resolution for nonstandard AWS regions**: Resolved an issue where incorrect AWS domains were used for regions such as China and ISO partitions. The fix ensures endpoints are resolved using the correct domain based on the region, improving compatibility across AWS environments.
- **Added shared `HttpClient` cache for `RestClient`**: Introduced a static cache to enable reuse of `HttpClient` instances across multiple `RestClient` instances, reducing overhead from repeated client creation.
- **Improved resilience of Jet jobs during member shutdown or topology changes**: Resolved an issue where Jet jobs could fail permanently if certain topology-related exceptions occurred during job startup, such as `HazelcastInstanceNotActiveException`, when a member stopped or restarted. Previously, these exceptions caused the job to enter a failed state without recovery. The fix ensures that such conditions trigger a job restart instead, improving reliability during cluster changes.

- **Fixed race condition during Jet job initialization and termination with restart**: Resolved a race condition between job initialization and a termination request with restart that could cause an `AssertionError` (mode is null). This could occur when a termination request was received while the job was still starting, leading to inconsistent internal state and job failure. The fix ensures that termination requests received during initialization are properly handled by aborting initialization and scheduling the restart through the standard job restart flow, preventing unexpected job failures and improving reliability during graceful member shutdown or scaling scenarios.
- **Updated Jet job cancellation handling for JDK 23+ compatibility**: Resolved an issue where changes in JDK 23 caused `Job.join()` and `isUserCancelled()` to lose the root cause `CancellationByUserException`.
- **Improved detection of event loss in Event Journal sources**: Event Journal sources for maps and caches now mark entries when events have been lost due to journal overlap (such as slow consumers). A new `isAfterLostEvents()` indicator allows Jet pipelines to detect and handle such cases, improving observability without adding overhead or additional events.
- **Fixed database errors and potential data loss after cluster network recovery**: Resolved an issue where reconnecting nodes could cause duplicate database write operations when using a write-behind MapStore. During a split-brain recovery, the system could keep old background tasks running alongside new ones, leading to database conflicts and lost updates. The fix ensures that old tasks are properly stopped during the merge process. This guarantees that only a single process writes to the database, ensuring reliable data persistence and preventing database errors.
- **Improved validation for null client cluster name configuration**: Resolved an issue where setting a `null` cluster name via `ClientConfig#setClusterName(null)` resulted in a `NullPointerException` during client connection with an unclear error message. The fix introduces fail-fast validation that throws an `IllegalArgumentException` when a `null` cluster name is configured, helping users identify the configuration problem earlier and with a clearer error message.
- **Fixed premature database writes for the first entry in write-behind maps**: Resolved an issue where the first item added to a write-behind map could be flushed to the external data store before the configured write delay had expired. This rare behavior occurred when the internal queue initialization and the initial data insertion happened within the exact same millisecond, causing a timing collision. The fix ensures that the configured write delay is strictly respected for all entries, preventing unexpected early database writes and maintaining consistent write-behind behavior.

- **Fixed cluster data migration failures if TRACE logging was enabled**: Resolved an issue where the cluster could get stuck while moving data between nodes. This occurred because an internal error could be triggered when the system attempted to write diagnostic logs at the exact same time as other background tasks were completing. The fix ensures that the internal logging mechanism safely handles simultaneous operations, allowing data migrations to complete smoothly and reliably.

Deprecations

- **Spring 5 Support**: Spring 5 support is deprecated and will be removed in Hazelcast Platform 6.0. Users are encouraged to upgrade to Spring 6.0 or later.
- **`operation.operationTimeoutCount` metric**: Deprecated because its functionality is already covered by `InvocationMonitor` timeout metrics. It will be removed in a future major release.

Security

- **Enforced permission checks for IMap projection and aggregation operations**: New `IMap.project()` and `IMap.aggregate()` permissions have been introduced and these operations are rejected if the corresponding permissions are not granted. **Action required**: users with fine-grained security must update their configurations to explicitly include these permissions where needed, otherwise these operations will fail.
- **Added configurable class restrictions for Zero Config Compact Serialization (ZCCS)**: Introduced optional allowlist/blocklist controls using `JavaSerializationFilterConfig` to restrict which classes can be used with ZCCS, mitigating risks from unsafe deserialization. Users are encouraged to configure restrictions to improve security ahead of stricter defaults in a future major release.
- **Resolved CVE-2026-33870 and CVE-2026-33871 in Netty**: Fixed vulnerabilities by upgrading the Netty dependency.
- **Resolved CVE-2026-34478, CVE-2026-34480, and CVE-2026-34481 in Apache Log4j**: Addressed potential security risks related to improper request handling and input processing.
- **Resolved CVE-2026-42198 in PostgreSQL JDBC driver**: Fixed a vulnerability by upgrading the `pgjdbc` dependency.

- **Enforced stricter checks on classes in SQL**: Resolved an issue where restrictions on classnames used in SQL mappings and types were not checked in some cases.
- **Enforced checks on classes in `JsonUtil` deserialization**: Some classes (unlikely to be used) are no longer allowed to be deserialized using `JsonUtil`. Customers using the `com.hazelcast.jet.json.JsonUtil` class are recommended to review the Javadoc for more secure alternative methods.
- **Enforced checks on classes used in client protocol**: In some situations it was possible to instantiate arbitrary classes in client protocol error conditions. Strict filtering has been implemented to prevent this issue.
- **Resolved CVE-2025-33042 in Avro**: Fixed a vulnerability by upgrading the Avro and Parquet dependencies.
- **Security Advisory regarding Elasticsearch 7**: CVE-2025-66566 has been identified in Elasticsearch 7. As Elasticsearch 7 is currently End of Life (EOL), no upstream fix is available from the vendor. We strongly recommend that users evaluate the security risks associated with this CVE. Support for Elasticsearch 7 will be removed in a future version of Hazelcast.
- **Fixed unrestricted attribute access during query lookups**: Resolved an issue where query lookups could access attributes that should not be exposed during query evaluation. The fix adds configurable restrictions on which attributes can be accessed, improving security and giving users more control over query behavior.
- **Resolved CVE-2025-12183 in LZ4 Java**: Fixed a vulnerability by upgrading the LZ4 Java dependency.
- **Resolved CVE-2025-59419 in Netty**: Fixed a vulnerability by upgrading the Netty dependency.
- **Resolved CVE-2026-22731, CVE-2026-22733, CVE-2026-22737, and CVE-2026-22735 in Spring Boot**: Fixed vulnerabilities by upgrading the Spring Boot dependency.
- **Resolved CVE-2024-6763 in Jetty**: Fixed vulnerability by upgrading the Jetty dependency. Note that although this CVE is fixed, it may be flagged as still present in security scans due to the EOL status of Jetty 9. Hazelcast are aware of the issue and are working to upgrade Jetty in a future release.

Known issues

- **Deeply nested JSON objects on JDK 25**: When running on JDK 25, extremely deeply nested JSON objects (approaching the ~1000 nesting depth limit) may trigger a `StackOverflowError`, causing the operation to fail. This is due to changes in the JDK and

not caused by Hazelcast code changes in this release. As a workaround, increase the JVM stack size using the `-Xss` startup flag.

- **`@ExposeHazelcastObjects` with dependent Hazelcast configuration or instance beans**: When running an application with Spring integration and `@ExposeHazelcastObjects` (explicit or implicit via autoconfiguration), beans with Hazelcast `Config` and/or `HazelcastInstance` cannot rely on dependencies such as classes marked as `@ConfigurationProperties`. The system incorrectly triggers early initialization of those beans before `@ConfigurationProperties` are injected.

Contributors

We would like to thank the contributors from our open source community who worked on this release:

- Connor-Roche-Fidelis
- Haricshore
- snicoll

Feedback

Edit this page