Enterprise Edition Release Notes

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

For help downloading Hazelcast Enterprise Edition, see Install Hazelcast Enterprise Edition or request a trial license key.

5.7.2

Release date: 2026-10-09

This is a maintenance release for Enterprise Edition.

Security

- **Masked SSL passwords in external configuration logs**: Prevented some passwords supplied through environment variables from being written in startup logs.
- **Resolved high and critical CVEs**: CVE-2026-56740, CVE-2026-56741, CVE-2026-59902, CVE-2026-59903, CVE-2026-62380, CVE-2026-65182, CVE-2026-65183, CVE-2026-65637, CVE-2026-65905, CVE-2026-65927, CVE-2026-66422, CVE-2026-68525, CVE-2026-68569, CVE-2026-68763, CVE-2026-71290, CVE-2026-73334, CVE-2026-75595, and CVE-2026-75596.

Fixes

- **Clarified CP member restart errors**: Removed misleading reset warnings and added cleanup instructions when restarting a removed CP member.
- **Fixed client cleanup timeout handling**: Corrected `ClientLifecycleMonitor` so timeouts configured in milliseconds are compared against elapsed time in the same unit.

- **Clarified locking behavior**: Updated `IMap` and `MultiMap` class documentation.
- **Fixed networking CPU spikes**: Prevented NIO threads from spinning when a member connection remains established but is dead.
- **Fixed `FencedLock` memory leak**: Prevented a memory leak, and the resulting slowdown of CP and other operations, when a CP session closed while holding a `FencedLock`.
- **Fixed CP group stall after leader change**: Prevented a CP group from being unable to commit operations when a leader change happened while a member was partway through receiving a snapshot.
- **Fixed delays after CP Subsystem reset**: Prevented clients that kept using CP objects across a reset from causing repeated retries.

Contributors

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

- guss77

5.7.1

Release date: 2026-08-13

This is a maintenance release for Enterprise Edition.

Security

- **Improved validation for serializable Jet functions**: Added additional validation to prevent serializable functions from connecting to attacker-controlled clusters or arbitrary addresses without a valid execution context.
- **Prevented arbitrary file read through Jet**: Hardened Jet usage in serializable contexts to prevent crafted operations from reading files from members, including configuration, secrets, or data stored on disk.
- **Removed exposure from experimental declarative pipeline functionality**: Addressed a permissions issue in the Enterprise declarative pipeline component that could allow

unauthorized Jet job submission and access to sources or sinks such as files, maps, Kafka, and JDBC when Jet/SQL was enabled.

- **Ensured SQL is enabled before deserializing messages**: Applied additional hardening to prevent SQL parameters from being deserialized when SQL is not enabled.
- **Enforced checks on classes in `JsonUtil` deserialization**: Some classes, which are unlikely to be used, are no longer allowed to be deserialized using `JsonUtil`.
- **Blocked Jackson serializable classes**: Prevented serializable Jackson classes from exposing information.
- **Hardened Jet path and class-loading validation**: Fixed path traversal issues in Jet processor class loading and in file sinks.
- **Hardened client deserialization**: Added a default Java serialization blocklist, and extended the zero-config Compact serialization blocklist.
- **Blocked reflective access via Spring classes**: Restricted classes in Predicate API and SQL paths to prevent reflective access to Hazelcast instances and sensitive objects.
- **Hardened SQL object creation and method invocation**: Prevented unprivileged actions in `IMap`.
- **Rejected malformed serialized arrays**: Added validation to detect truncated array data before allocating or deserializing unexpectedly large arrays.
- **Resolved vulnerabilities in Jackson**: Addressed reported vulnerabilities, including Jackson Core async parser DoS GHSA-72hv-8253-57qq and CVEs CVE-2026-54512 and CVE-2026-54513.
- **Resolved high and critical CVEs identified by security scans**: Updated affected components to resolve high and critical CVEs CVE-2026-56816, CVE-2026-2332, CVE-2026-33117, CVE-2026-41293, CVE-2026-41855, CVE-2026-42579, CVE-2026-42581, CVE-2026-42584, CVE-2026-43512, CVE-2026-43515, CVE-2026-45674, CVE-2026-47691, CVE-2026-53434, CVE-2026-55276, CVE-2026-59083, CVE-2026-59084, CVE-2025-37731, CVE-2026-24051, CVE-2026-33870, CVE-2026-33871, CVE-2026-39883, CVE-2026-41284, CVE-2026-41838, CVE-2026-41842, CVE-2026-41848, CVE-2026-41850, CVE-2026-41851, CVE-2026-42498, CVE-2026-42583, CVE-2026-42585, CVE-2026-42586, CVE-2026-42587, CVE-2026-43513, CVE-2026-44248, CVE-2026-44249, CVE-2026-44250, CVE-2026-44890, CVE-2026-44891, CVE-2026-44893, CVE-2026-45416, CVE-2026-46340, CVE-2026-48006, CVE-2026-48043, CVE-2026-48059, CVE-2026-50010, CVE-2026-50011, CVE-2026-53404, CVE-2026-54291, CVE-2026-54399, CVE-2026-54428, CVE-2026-54512, CVE-2026-54513, CVE-2026-55831, CVE-2026-55833, CVE-2026-5795.

Fixes

- **Fixed `IndexOutOfBoundsException` in HD `IMap` SetOperation with `MapStore` and TTL**: Resolved an issue where a slow `MapStore` store operation could outlive an entry TTL and cause an `IndexOutOfBoundsException` while returning the old value for an HD `IMap` update. The fix prevents unexpected operation failures in this scenario.
- **Fixed Kafka sink failures caused by duplicate transactional IDs**: Resolved an issue where Jet jobs writing to Kafka could suspend with `ProducerFencedException` when a newer producer with the same transactional ID fenced the current producer.
- **Fixed potential `MapStore` data loss when a member fails during a flush**: Resolved an issue where manual flush operations could clear queues before the entries were persisted.
- **Fixed Jet jobs becoming stuck after an ungraceful restart**: Resolved two issues where a running snapshot from a previous execution left the job stuck in the `COMPLETING` state, or a failed cleanup blocked subsequent snapshots.
- **Added Kubernetes state-management logging**: Added FINE and FINEST logging around Kubernetes topology state management to help diagnose clusters stuck in `NO_MIGRATION`.
- **Released retained WAN events**: Prevented the last drained event from remaining in the WAN event queue and consuming unnecessary resources.
- **Fixed persistence with `MapStore` after restart**: Prevented map operations from remaining blocked in the passive state when persistence and `MapStore` are both enabled after a cluster restart.
- **Removed redundant field deserialization**: Resolved an issue where `ReflectiveCompactSerializer` deserialized each non-primitive field twice when reading Compact-serialized objects.

5.7.0

Release date: 2026-05-13

- New features
- Breaking changes
- Enhancements
- Fixes

- Deprecations
- Security
- Known issues
- Contributors

New features

- **CP Leader Auto Step-Down**: CP members can now be configured to automatically step down (abdicate) from leadership if elected, ensuring CP group leaders remain on preferred, low-latency "leader-capable" members. This reduces the risk of performance degradation and instability caused by high-latency leaders while preserving Raft fault-tolerance requirements during normal operations. The change is observable via logs and metrics.

For more details on new features, see What's new.

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.

> **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.

> **NOTE**
>
> The Hazelcast Platform Operator defaults to Java 25 unless the deployment is pinned to a specific Java version.

- **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.

- **Improved visibility for WAN Replication queue overflow and dropped events**: Enhanced observability for WAN Replication scenarios where queues become full and events are dropped. Member log messages for "queue full" conditions now include the affected Map/Cache name and the corresponding WAN Publisher, making it easier to identify the source of dropped events. Management Center surfaces alerts on the WAN Replication page to indicate when queues are full and entries are dropped, including the relevant queue and Map/Cache details.

- **Improved Native `OutOfMemoryError` (OOME) diagnostics**: Native OOME messages from `PoolingMemoryManager` now include detailed fragmentation and memory distribution statistics, such as maximum page fragmentation, thread memory imbalance, unusable free memory, and total allocated (including external) memory. This makes it easier to identify whether allocation failures are caused by fragmentation, uneven thread-local memory usage, or actual memory exhaustion.

- **Improved Native `OutOfMemoryError` handling for HD `IMap` with JSON keys and values**: Resolved an issue where a Native OOME during insertion into an HD `IMap` using `HazelcastJsonValue` keys and values could lead to inconsistent behavior, such as

incorrect exceptions being reported or native memory not being fully released. The fix ensures that NOOME is handled and reported correctly, prevents unintended double-free scenarios, and keeps native memory accounting and system behavior consistent.

- **Added Jackson 3 support alongside Jackson 2**: Introduced Jackson 3.x to enable support for records with JSON annotations, while retaining Jackson 2.x on the classpath for compatibility with existing dependencies.

- **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.

Fixes

- **Fixed failures and inconsistent results in `IMap.containsValue()` for maps in HD Memory 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 HD and TTL**: Resolved a performance issue affecting Near Caches using on-heap in-memory format 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.

- **Fixed log format inconsistency when Enterprise REST is enabled**: Resolved an issue where enabling the Enterprise REST endpoint caused Hazelcast to emit log entries in two conflicting formats — the default Hazelcast format and the Spring-based format introduced by the REST layer. The `hazelcast.logging.type` setting now takes precedence over Spring Boot's logging system. When Hazelcast runs in embedded mode in the same JVM as Spring Boot with the REST API enabled, Spring Boot's logging system is automatically aligned with `hazelcast.logging.type`. To keep a different Spring Boot logging system, set `org.springframework.boot.logging.LoggingSystem` explicitly.
- **Fixed missing OSGi package exports in Enterprise bundle**: Resolved an issue where several core Hazelcast packages were not exported correctly in the `MANIFEST.MF` starting from 5.5.0.
- **Fixed CP Raft log loading failure caused by incomplete chunked snapshots**: Resolved an issue where partially written snapshot chunks could be treated as valid Raft log files, leading to errors during state reload. The fix ensures incomplete files are ignored and logged, and previous log files are only deleted once a complete snapshot is successfully persisted, improving recovery reliability with no user action required.
- **Fixed failure during CP snapshot restore when upgrading from 5.5 to 5.6**: Resolved an issue where restoring Raft snapshots created in 5.5 could fail in 5.6 due to a missing conversion step in the snapshot processing path, causing repeated errors and blocking upgrades.
- **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.

Deprecations

- **Tiered Storage**: Tiered Storage is deprecated and will be removed in Hazelcast Platform 6.0.
- **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 Hazelcast Platform 6.0.

Security

- **Improved permission checks for `IMap` `ServiceFactory` used in Jet pipelines**: Resolved an issue where some mutating `IMap` operations executed through the `ServiceFactory` did not enforce the required permissions. As a result, Jet jobs submitted by users without sufficient permissions may now fail when performing `IMap` operations that require permissions beyond read access.
- **Fixed missing permission checks for `EntryProcessor` execution through `ExecutorService`**: Resolved an issue where `EntryProcessor` operations submitted via `submitToKey`, `submitToKeys`, or `executeOnKeys` from an `ExecutorService` callable only validated the `remove` permission, while the required `put` permission was not enforced.
- **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.

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.

- **CVE-2024-6763 may be flagged as unresolved in Jetty despite the fix**: Although CVE-2024-6763 was resolved in Hazelcast Platform 5.6.1 by upgrading the Jetty dependency, the vulnerability may continue to be reported as present in security scans. This occurs because Jetty 9 has reached End of Life and scanners may flag it regardless of the fix. Hazelcast is aware of this and is working to upgrade Jetty in a future release.

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