

NATS 2.12



- [Features](#)
 - [Streams](#)
 - [Consumers](#)
 - [Operations](#)
 - [Improvements](#)
 - [Upgrade Considerations](#)
 - [Memory usage](#)
 - [Strict JetStream API](#)
 - [Downgrade Considerations](#)
 - [Stream state](#)
-

This guide is tailored for existing NATS users upgrading from NATS version v2.11.x. This will read as a summary with links to specific documentation pages to learn more about the feature or improvement.

Features

Streams

- **Atomic batch publish:** The `AllowAtomicPublish` stream configuration option allows to atomically publish N messages into a stream. This includes support for replicated and non-replicated streams, as well as doing per-message consistency checks prior to committing the batch. More information is available in [ADR-50](#).
- **Distributed Counter CRDT:** The `AllowMsgCounter` stream configuration option allows increment/decrement counter semantics on a stream. These counter streams can also be mirrored or aggregated through stream mirroring and sourcing. More information is available in [ADR-49](#)
- **Delayed Message Scheduling:** The `AllowMsgSchedules` stream configuration option allows the scheduling of messages. Users can use this feature for delayed

publishing/scheduling of messages. More information is available in [ADR-51](#)

Consumers

- **Prioritized pull consumer policy:** In addition to the consumer policies like overflow or client pinning, a new `prioritized` policy has been added. In contrast with the overflow policy, this allows a consumer to receive messages sooner instead of delaying failover, but at the cost of potentially flip-flopping work between clients. More information is available in [ADR-42](#)

Operations

- **Server metadata:** Similar to `server_tags` which contains a set of tags describing the server, `server_metadata` is a map containing string keys and values describing metadata of the server.
- **Promoting mirrors:** A stream that's mirroring can now be promoted to be the primary, enabling new disaster recovery methodology. The current primary stream should be deleted or have its configured subjects removed prior to promoting mirrors, before configuring the promoted mirrors to start listening on those subjects.
- **Exponential backoff on route and gateway connections:** Cluster routes and gateways can now use exponential backoff on reconnection attempts by setting `connect_backoff`. If `true`, will start exponential backoff at 1 second up to 30 seconds. This can slow down the speed of reconnection but significantly reduces the amount of DNS queries and general connection attempts during server restarts or outages.
- **Offline assets:** When downgrading to an older version, the server can now recognize new features were used and puts the stream and/or consumer into an unsupported/offline mode. For more information, read also the downgrade considerations. More information is available in [ADR-44](#)
- **Stream/consumer scaleup and reset disk/state protection:** The server now has better protections against leader elections based on empty state. This also improves reliability of replicated in-memory streams. Usually a quorum of servers needs to be online and contain data. Now all but one server can be restarted and the in-memory stream's data can reliably be caught back up. However, during such a scenario all servers involved with replication of that stream will need to be available, not just what's needed for quorum. This lets the servers decide the best course of action to preserve all data.

Improvements

- **Async stream flushing:** Replicated streams will now asynchronously flush data to the underlying store on disk, resulting in a significant improvement in performance. Writes to a replicated stream are still persisted synchronously in the Raft log prior to committing them, so the improved performance has no downsides with respect to consistency.
- **Elastic pointers in the filestore:** File-based streams now use elastic pointers for its write-through caches. This allows the server to better respond during garbage collection, these caches can be evicted early to avoid out-of-memory conditions (see upgrade considerations below).
- **Use cipher suites from `crypto/tls`:** New cipher suites are now automatically added. Additionally, insecure cipher suites are disabled by default, but can be allowed when enabling `allow_insecure_cipher_suites`.
- **System events for the `$G` account:** The global account (`$G`) will now also produce system events, such as connect and disconnect events.
- **`GOMAXPROCS` and `GOMEMLIMIT` in server stats:** The server stats already contained the CPU and memory usage of the server but now also contains the effective Go limits.
- **New subject transforms: `partition(n)` and `random(n)`:** In addition to `partition(n, ...)` which allows to determine a partition number based on tokens at specified indices, `partition(n)` and `random(n)` are convenience functions to create a partition or random number up to `n` based on the whole subject.
- **Account name and user logging:** Any logging related to a client connection, for example when reaching maximum connections or for authentication errors, will now include the account name and user of that client connection.
- **Logging improvements:** Any logging related to a client connection now includes the account and user name. Connection closed logging now includes the remote server name.
- **Isolated leaf node property:** In a large deployment with lots of leaf nodes, propagating east-west interest can result in a lot of traffic, which is wasted if leaf nodes don't need to be able to publish/subscribe to each other directly. Instead of the workaround of setting the cluster name of those leaf nodes to be the same, the `isolate_leafnode_interest` property can now be used.
- **Disable leaf node connection through config reload:** This allows disabling a remote leaf node using configuration reload, when using `disabled: true`. If changed from false to true, a solicited leaf node will be disconnected and will not reconnect. If changed from true to false, the leafnode will be solicited again.

Upgrade Considerations

Memory usage

With the new elastic pointers in the filestore, it is expected that a NATS Server running 2.12 may show a different memory usage pattern to before. In some systems this may result in lower resident set size (RSS) reported, in others it may result in higher, depending on the number of assets and publish/access patterns.

For the first time, the server will be able to respond to memory pressure by freeing filestore caches on demand and returning the memory to the operating system. This reduces the chance that sudden spikes in utilisation will result in an out-of-memory (OOM) kill. However, this means that the server can more optimistically retain caches in memory when available resources allow in order to facilitate improved read access times.

This behaviour is largely controlled by the GC thresholds as set by the `GOMEMLIMIT` [environment variable](#). You may wish to tune this value in your environment based on available system memory, or in the case of Kubernetes environments, memory reservations.

Strict JetStream API

Starting from version v2.11, the server would start logging the following statement if an invalid JetStream request was received:

```
[WRN] Invalid JetStream request '$G > $JS.API.STREAM.CREATE.test-stream': json: unknown field
"unknown"
```

Starting from version v2.12, the server will not only log, but also return an error to the client as “strict mode” is now enabled by default. This means invalid JetStream requests will be rejected by default.

If the above log message is observed, please make sure that the application or client is sending correct requests to the server and that NATS client libraries are up-to-date. Strict mode can be temporarily disabled in the server configuration, allowing you more time to fix the issue:

```
jetstream {
  strict: false
}
```

Downgrade Considerations

Stream state

When downgrading from v2.12 to v2.11, the stream state files on disk will be rebuilt due to a change in the format of these files in v2.12. This requires re-scanning all stream message blocks, which may use higher CPU than usual and will likely take longer for the restarted node to report healthy. This will only happen on the first restart after downgrading and will not result in data loss.

When downgrading, only downgrade to v2.11.9 or higher. Starting from this version, the server will recognize the use of new v2.12 features and will safely put the stream and/or consumer that uses these new features into an unsupported/offline mode. Importantly, this will both protect the data as well as the server itself from accessing unsupported features or data.

📁 #1

📁 cocobook-sync 📁 2026-10-28 11:08:49 UTC

📁 cocobook-sync 📁 2026-10-28 11:08:49 UTC