

NATS 2.11



- [Features](#)
 - [Observability](#)
 - [Streams](#)
 - [Consumers](#)
 - [Operations](#)
 - [MQTT](#)
 - [Improvements](#)
 - [Upgrade Considerations](#)
 - [Stream ingest rate limiting](#)
 - [Replicated delete proposals](#)
 - [JetStream healthcheck](#)
 - [Exit code](#)
 - [Server, cluster and gateway names](#)
 - [Downgrade Considerations](#)
 - [Stream state](#)
-

This guide is tailored for existing NATS users upgrading from NATS version v2.10.x. This will read as a summary with links to specific documentation pages to learn more about the feature or improvement.

Features

Observability

- **Distributed message tracing:** Users can now trace messages as they move through the system by setting a `Nats-Trace-Dest` header to an inbox subject. Servers on the message path will return events to the provided subject that report each time a message enters or leaves a server, by which connection type, when subject mappings occur, or

when messages traverse an account import/export boundary. Additionally, the `Nats-Trace-Only` header (if set to true) will allow tracing events to propagate on a specific subject without delivering them to subscribers of that subject.

Streams

- **JetStream per-message TTLs:** It is now possible to age out individual messages using a per-message TTL. The `Nats-TTL` header, in either string or integer format (in seconds) allows for individual message expiration independent of stream limits. This can be combined with other limits in place on the stream. More information is available in [ADR-43](#).
- **Subject delete markers on MaxAge:** The `SubjectDeleteMarkerTTL` stream configuration option now allows for the placement of delete marker messages in the stream when the configured `MaxAge` limit causes the last message for a given subject to be deleted. The delete markers include a `Nats-Marker-Reason` header explaining which limit was responsible for the deletion.
- **Stream ingest rate limiting:** New options `max_buffered_size` and `max_buffered_msgs` in the `jetstream` configuration block enable rate limiting on Core NATS publishing into JetStream streams, protecting the system from overload.

Consumers

- **Pull consumer priority groups:** Pull consumers now support priority groups with pinning and overflow, enabling flexible failover and priority management when multiple clients are pulling from the same consumer. Configurable policies based on the number of pending messages on the consumer, or the number of pending acks, can control when messages overflow from one client to another, enabling new design patterns or regional awareness.
- **Consumer pausing:** Message delivery to consumers can be temporarily suspended using the new pause API endpoint (or the `PauseUntil` configuration option when creating), ideal for maintenance or migrations. Message delivery automatically resumes once the configured deadline has passed. Consumer clients continue to receive heartbeat messages as usual to ensure that they do not surface errors during the pause.

Operations

- **Replication traffic in asset accounts:** Raft replication traffic can optionally be moved into the same account in which replicated assets live on a per-account basis, rather than being sent and received in the system account using the new `cluster_traffic` property in the JetStream account settings of an account. When combined with multiple route connections, this can help to reduce latencies and avoid head-of-line blocking issues that

may occur in heavily-loaded multi-tenant or multi-account deployments.

- **TLS first on leafnode connections:** A new `handshake_first` in the leafnode `tls` block allows setting up leafnode connections that perform TLS negotiation first, before any other protocol handshakes take place.
- **Configuration state digest:** A new `-t` command line flag on the server binary can generate a hash of the configuration file. The `config_digest` item in `varz` displays the hash of the currently running configuration file, making it possible to check whether a configuration file has changed on disk compared to the currently running configuration.
- **TPM encryption on Windows:** When running on Windows, the filestore can now store encryption keys in the TPM, useful in environments where physical access may be a concern.

MQTT

- **SparkplugB:** The built-in MQTT support is now compliant with SparkplugB Aware, with support for `NBIRTH` and `NDEATH` messages.

Improvements

- **Replicated delete proposals:** Message removals in clustered interest-based or workqueue streams are now propagated via Raft to guarantee consistent removal order across replicas, reducing a number of possible ways that a cluster failure can result in de-synced streams.
- **Metalayer, stream and consumer consistency:** A new leader now only responds to read/write requests after synchronizing with its Raft log, preventing desynchronization between KV key updates and the stream during leader changes.
- **Replicated consumer reliability:** Replicated consumers now consistently redeliver unacknowledged messages after a leader change.
- **Consumer starting sequence:** The consumer starting sequence is now always respected, except for internal hidden consumers for sources/mirrors.

Upgrade Considerations

Stream ingest rate limiting

The NATS Server can now return a 429 error with type `JStreamTooManyRequests` when too many messages have been queued up for a stream. It should not generally be possible to hit this limit while using JetStream publishes and waiting for PubAcks, but may trigger if trying to publish into JetStream using Core NATS publishes without waiting for PubAcks, which is not advised.

The new `max_buffered_size` and `max_buffered_msgs` options control how many messages can be queued for each stream before the rate limit is hit, therefore if needed, you can increase these limits on your deployments. The default values for `max_buffered_size` and `max_buffered_msgs` are 128MB and 10,000 respectively, whereas in v2.10 these were unlimited.

You can detect in the server logs whether running into a queue limit with the following warning:

```
[WRN] Dropping messages due to excessive stream ingest rate on 'account' > 'my-stream': IPQ
len limit reached
```

If your application starts to log the above warnings then you can first try to increase the limits to higher values while investigating the fast publishers, for example:

```
jetstream {
  max_buffered_msgs: 50000
  max_buffered_size: 256mib
}
```

Replicated delete proposals

Since stream deletes are now replicated through group proposals in a replicated stream, there may be a slight increase in replication traffic on this version.

JetStream healthcheck

The `js-server-only` healthcheck no longer checks for the health of the metaleader on v2.11.0. Since this healthcheck was designed to detect the server readiness (or in k8s for the readiness probe) checking the metaleader would sometimes cause a NATS server to be considered unhealthy when restarting the servers. In v2.11, this should no longer be an issue. If the previous behavior from v2.10 is preferred, there is a new healthcheck option `js-meta-only` which can be used to check whether the meta group is healthy.

Exit code

Earlier versions of the NATS Server would return an exit code 1 when gracefully shut down, i.e. after SIGTERM. From v2.11, an exit code of 0 (zero) will now be returned instead.

Server, cluster and gateway names

Configurations that have server, cluster, and gateway names with spaces are now considered invalid, as this can cause problems at the protocol level. A server running NATS v2.11 will fail to start with spaces configured in these names. Please ensure that spaces are not used in server, cluster or gateway names.

Downgrade Considerations

Stream state

When downgrading from v2.11 to v2.10, the stream state files on disk will be rebuilt due to a change in the format of these files in v2.11. 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.

[] [] #2	
[] cocobook-sync [] []	2026-10-28 11:09:55 UTC
[] cocobook-sync [] []	2026-10-28 11:15:12 UTC