

NATS 2.14



- [Features](#)
 - [Streams](#)
 - [Consumers](#)
 - [Operations](#)
 - [Improvements](#)
 - [Upgrade considerations](#)
 - [Reliable WorkQueue and Interest mirroring/sourcing](#)
 - [Domain-aware acknowledgement and flow control subjects](#)
 - [Filestore I/O error handling](#)
 - [Raft overrun protection](#)
 - [Downgrade considerations](#)
 - [Reliable WorkQueue and Interest mirroring/sourcing](#)
 - [Feature flags](#)
-

This guide is tailored for existing NATS users upgrading from NATS version v2.12.x. This will read as a summary with links to specific documentation pages to learn more about the feature or improvement.

Features

Streams

- **Fast batch publish:** The `AllowBatchPublish` stream configuration option allows for high throughput and flow controlled publishing into a stream. This includes support for replicated and non-replicated streams, as well as doing per-message consistency checks without intermediate staging of messages (like with atomic batch publish). More information is available in [ADR-50](#).
- **Recurring schedules:** The `AllowMsgSchedules` stream configuration option now also allows the usage of recurring schedules, either based on a simple interval, or with Cron.

More information is available in [ADR-51](#).

- **Scheduled subject sampling:** The `AllowMsgSchedules` stream configuration option now also allows to source the data of the last message matching the subject in the scheduled message. Useful for downsampling data on an interval. More information is available in [ADR-51](#).
- **Reliable WorkQueue and Interest mirroring/sourcing:** Sourcing or mirroring from a WorkQueue or Interest retention stream is now supported. A durable consumer, as opposed to an ephemeral one, will automatically be used to perform the async replication. A new ack policy of `AckFlowControl` is used to acknowledge messages after they were persisted, based on flow control. More information is available in [ADR-60](#), see also the upgrade considerations below.

Consumers

- **Consumer reset API:** Consumer delivery state can now be reset back to the acknowledgement floor, or to an arbitrary sequence (while still respecting start sequences etc). The consumer state after reset equals what would otherwise be a consumer delete and recreate at a specific starting sequence. More information is available in [ADR-60](#).

Operations

- **Leafnode remote config reload:** The leaf node remotes section can now be added and removed via a configuration reload, without requiring a server restart.
- **Filestore I/O error handling:** Previously, not all filestore I/O errors were properly handled, allowing the stream and server to continue to run. These errors will now be surfaced in the logs and show up in the health check, as well as freeze the stream to prevent further updates (see upgrade considerations below).
- **Raft overrun protection:** The server now recognizes if its Raft layer is being overrun by proposals, bounding the number of memory and disk resources the server is allowed to use during such an event (see upgrade considerations below).

Improvements

- **Deduplication changes when using stream sourcing:** Streams with sources now allow deduplication to be disabled. Additionally, sourcing streams can now perform deduplication when fanning in multiple sources.
- **Atomic batch publish, EOB commit support:** Atomic batches can now be committed through an EOB (End of Batch) message without persisting this final message. This is also supported when using the new fast batch publish.

- **Scheduled subject rollups:** The `Nats-Schedule-Rollup` header can now be used to place a rollup on the scheduled message, similar to the `Nats-Schedule-TTL` header.
- **Feature flags:** The server now supports a `feature_flags` field in the config which allows users to opt-in or opt-out to specific fixes or improvements prior to them becoming the default in a future release.
- **Domain-aware acknowledgement and flow control subjects:** The server now supports both v1 and v2 of the consumer acknowledgement and flow control subjects. More information is available in [ADR-15](#).
- **Traceparent header:** The `traceparent` header is no longer modified by the message tracing.
- **Asynchronous stream state snapshots for replicated streams:** Allows stream state snapshots to be taken and written without pausing stream processing, improving tail latencies. This is particularly impactful in cases where the stream has a large number of interior deletes.

Upgrade considerations

Reliable WorkQueue and Interest mirroring/sourcing

Version 2.14 adds support for sourcing and mirroring from WorkQueue or Interest streams. During an upgrade or downgrade, the server could temporarily log the following message:

```
[WRN] Invalid JetStream request '$G > $JS.API.CONSUMER.CREATE.0': json: unknown field
"sourcing"
```

This means that an upgraded server tried to create the ‘new-style’ sourcing consumer but the older target server didn’t recognize it and logged a warning. The upgraded server will automatically respond to this error and will send the request again using the ‘old-style’ sourcing consumer. It is expected that these logs could temporarily occur during upgrades/downgrades, but these should resolve once the whole system is upgraded.

Domain-aware acknowledgement and flow control subjects

The server now supports both v1 and v2 acknowledgement and flow control reply subjects documented in [ADR-15](#). The v2 format includes a domain and account hash to deconflict stream and consumer names across domains and accounts:

```
v1: $JS.ACK.<stream name>.<consumer name>.<num delivered>.<stream sequence>.<consumer
sequence>.<timestamp>.<num pending>
v2: $JS.ACK.<domain>.<account hash>.<stream name>.<consumer name>.<num delivered>.<stream
sequence>.<consumer sequence>.<timestamp>.<num pending>
```

While both v1 and v2 formats will be supported starting from 2.14, v1 remains the default. However, in version 2.15 the default will change to be the v2 format. Users that have defined account imports/exports or subject permissions containing the `$JS.ACK.<stream>.>` or `$JS.FC.<stream>.>` (or more granular) subjects **will be required to update their ACLs and/or account imports/exports before the 2.15 release** to allow the same stream and consumer names to be used in different domains or accounts without them conflicting with each other.

If you have not defined such account imports/exports or subject permissions, for example if you use JetStream only within a single account, or you defined them as the “catch-all wildcard” `$JS.ACK.>` or `$JS.FC.>` then you will not need to make any changes. The default will change in version 2.15 and there should be no impact.

To ease the migration path, the server now supports feature flags to test and enable this at your own convenience. Not specifying the feature flag means “use the server default” which for 2.14 will be the v1 format. Setting it to `true` will use the v2 format (also the v1 format will still be supported) and setting it to `false` will use the v1 format but still support v2.

```
feature_flags {  
  js_ack_fc_v2: true  
}
```

Filestore I/O error handling

Previously, not all filestore I/O errors were appropriately handled, allowing the stream and server to continue to run. In 2.14, these errors are surfaced: an affected stream freezes, logs the error, and reports an unhealthy state in health checks. Other streams on the same server remain unaffected, and for replicated streams another replica picks up the work transparently.

This change introduces a new operational condition to watch for: if I/O errors are encountered, each stream affected by it will stop making progress. NATS core traffic is not affected by I/O errors, so the server keeps functioning, but will require a restart to recover from these I/O issues. This condition can be observed by the health check failing and reporting an I/O related error, the error message will contain `write error` and which error was encountered.

Raft overrun protection

A new back-pressure mechanism in the Raft consensus layer prevents unbounded memory growth in overloaded clusters. In prior versions, there could be situations where entries would be written to the Raft write-ahead log faster than they could be committed and applied, resulting in increased memory and disk usage.

In 2.14, this condition is recognized and gives the relevant servers room to catch up. Leaders that detect they are falling behind step down so that a healthier peer can take over. If a majority of peers are equally overloaded, the system remains in this degraded state. The protection is a safety net for transient overload, not a substitute for adequate capacity, but it will allow the system to

continue in a degraded but functional mode instead of allowing the system to become overloaded further.

Downgrade considerations

Reliable WorkQueue and Interest mirroring/sourcing

The same consideration as mentioned above under the upgrade considerations holds here as well. The `Invalid JetStream request` log line may be observed during a downgrade. Additionally, stream sourcing or mirroring might temporarily be unable to function until all servers are downgraded to the 2.12 version.

If you've started using stream sourcing or mirroring on WorkQueue or Interest streams after upgrading, this will still seem to function but will operate under the less reliable ephemeral consumer mode. Additionally, any durable consumers created using the new configuration of `AckFlowControl` will be marked as "offline" and will not be usable until upgraded back to 2.14.

Feature flags

If you have defined the `feature_flags` field in your server config, you'll need to remove this prior to downgrading, since prior server versions will not recognize this field.

```
### #1
[] cocobook-sync [] 2026-10-28 11:08:48 UTC
[] cocobook-sync [] 2026-10-28 11:08:48 UTC
```