JetStream RePublish

Metadata	Value
Date	2022-07-08
Author	@derekcollison, @tbeets
Status	Implemented
Tags	jetstream, server

Update History

Date	Author	Description
2023-06-27	@tbeets	Fix typo on JSON boolean in `headers_only` example

Context and Problem Statement

In some use cases it is useful for a subscriber to monitor messages that have been ingested by a stream (captured to store) without incurring the overhead of defining and using a JS Consumer on the stream.

Such use cases include (but are not limited to):

- Lightweight stream publish monitors (such as a dashboard) that don't require the overhead of At-Least-Once delivery
- No side-effect WorkQueue and Interest-Based stream publish monitoring
- KV or Object Store update events as an alternative to watches, e.g. an option for cache invalidation

Design

If stream *RePublish* option is configured, a stream will evaluate each published message (that it ingests) against a *RePublish Source* subject filter. Upon match, the stream will re-publish the message (with special message headers as below) to a new *RePublish Destination* subject derived through a subject transformation.

> Re-publish occurs only after the original published message is ingested in the stream (with quorum for R>1 streams) and is *At-Most-Once* QoS.

RePublish Configuration Option

The RePublish option "republish" consists of three configuration fields:

Field	Description	JSON	Required	Default
Source	Published Subject-matching filter	src	N	>
Destination	RePublish Subject template	dest	Y	
Headers Only	Whether to RePublish only headers (no body)	headers_only	N	false

The following validation rules for RePublish option apply:

- A single token as `>` wildcard is allowed as the Source with meaning taken as any stream-ingested subject.
- Destination MUST have at least 1 non-wildcard token
- Destination MAY not match or subset the subject filter(s) of the stream
- Source and Destination must otherwise comply with requirements specified in ADR-30 Subject Transform.

Here is an example of a stream configuration with the RePublish option specified:

```
{
    "name": "Stream1",
    "subjects": [
        "one.>",
        "four.>"
    ],
    "republish": {
        "src": "one.>",
        "dest": "uno.>",
        "headers_only": false
    },
    "retention": "limits",
    ... omitted ...
}
```

In the configuration above, a published message at `one.foo.bar` will be ingested into `Stream1` as `one.foo.bar` and re-published as `uno.foo.bar`. Published messages at `four.foo.bar` will be ingested into `Stream1` but not re-published.

> RePublish option configuration MAY be edited after stream creation.

RePublish Transform

RePublish Destination, taken together with RePublish Source, form a valid subject token transform rule. The resulting transform is applied to each ingested message (that matches Source configuration) to determine the the concrete RePublish Subject.

See ADR-30 Subject Transform for description of subject transformation as used by RePublish.

RePublish Headers

Each RePublished Message will have the following message headers:

Header	Value Description
Nats-Stream	Stream name (in scope to stream's account)
Nats-Subject	Message's original subject as ingested into stream
Nats-Sequence	This message's stream sequence id

Header	Value Description
Nats-Last-Sequence	The stream sequence id of the last message ingested to the same original subject (or 0 if none or deleted)

If headers-only is "true", also:

Header	Value Description
Nats-Msg-Size	The size in bytes of the message's body

> Application-added headers in the original published message will be preserved in the re-published message.

Loop Prevention

Valid Destination configuration checks insures that re-published messages are not immediately ingested into the original stream (causing a loop). The scope of loop-detection is to the immediate stream only.

> Caution: It is possible to create a loop condition between two streams sharing an overlap in republish destinations and subject filters within a single account.