

Subject-Based Messaging



- [Wildcards](#)
 - [Subject hierarchies](#)
 - [Subject usage best practices](#)
 - [Number of subjects](#)
 - [Subject-based filtering and security](#)
 - [Naming things](#)
 - [Matching a single token](#)
 - [Matching multiple tokens](#)
 - [Monitoring and wire taps](#)
 - [Mixing wildcards](#)
 - [Characters allowed and recommended for subject names](#)
 - [Pedantic mode](#)
-

NATS is a system for publishing and listening for messages on named communication channels we call `Subjects`. Fundamentally, NATS is an `interest-based` messaging system, where the listener has to `subscribe` to a subset of `subjects`.

In other middleware systems subjects may be called `topics`, `channels`, `streams` (Note that in NATS the term `stream` is used for a [JetStream](#) message storage).

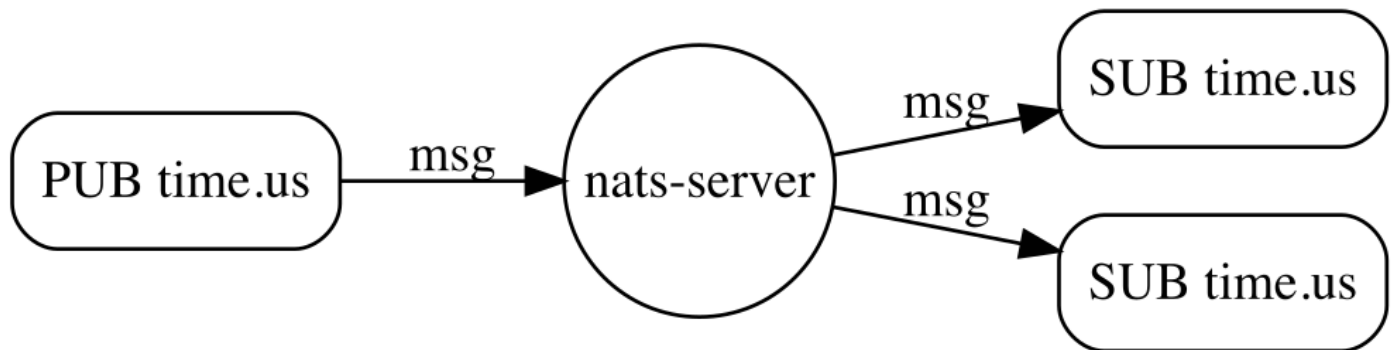
What is a subject? At its simplest, a subject is just a string of characters that form a name the publisher and subscriber can use to find each other. More commonly [subject hierarchies](#) are used to scope messages into semantic namespaces.

“ ”

Please check the [constraint and conventions](#) on naming for subjects here.

Location transparency Through subject-based addressing, NATS provides location transparency across a (large) cloud of routed NATS servers.

- Subject subscriptions are automatically propagated within the server cloud.
- Messages will be automatically routed to all interested subscribers, independent of location.
- Messages with no subscribers to their subject are automatically discarded (Please see the [JetStream](#) feature for message persistence).



Wildcards

NATS provides two *wildcards* that can take the place of one or more elements in a dot-separated subject. Publishers will always send a message to a fully specified subject, without the wildcard. While subscribers can use these wildcards to listen to multiple subjects with a single subscription.

Subject hierarchies

The `.` character is used to create a subject hierarchy. For example, a world clock application might define the following to logically group related subjects:

```
time.us
time.us.east
time.us.east.atlanta
time.eu.east
time.eu.east.warsaw
```

Subject usage best practices

There is no hard limit to subject size, but it is recommended to keep the maximum number of tokens in your subjects to a reasonable value. E.g. a maximum of 16 tokens and the subject length to less than 256 characters.

Number of subjects

NATS can manage 10s of millions of subjects efficiently, therefore, you can use fine-grained addressing for your business entities. Subjects are ephemeral resources, which will disappear when no longer subscribed to.

Still, subject subscriptions need to be cached by the server in memory. Consider when increasing your subscribed subject count to more than one million you will need more than 1GB of server memory and it will grow linearly from there.

Subject-based filtering and security

The message subject can be filtered with various means and through various configuration elements in your NATS server cluster. For example, but not limited to:

- Security - allow/deny per user
- Import/export between accounts
- Automatic transformations
- When inserting messages into JetStream streams
- When sourcing/mirroring JetStream streams
- When connecting leaf nodes (NATS edge servers)
- ...

A well-designed subject hierarchy will make the job a lot easier for those tasks.

Naming things

“ ”

There are only two hard problems in computer science: cache invalidation, naming things, and off-by-one errors. -- Unknown author

A subject hierarchy is a powerful tool for addressing your application resources. Most NATS users therefore encode business semantics into the subject name. You are free to choose a structure fit for your purpose, but you should refrain from over-complicating your subject design at the start of the project.

Some guidelines:

- Use the first token(s) to establish a general namespace.

```
factory1.tools.group42.unit17
```

- Use the final token(s) for identifiers

```
service.deploy.server-acme.app123
```

- A subject *should* be used for more than one message.
- Subscriptions *should* be stable (exist for receiving more than one message).
- Use wildcard subscriptions over subscribing to individual subjects whenever feasible.
- Name business or physical entities. Refrain from encoding too much data into the subject.
- Encode (business) intent into the subject, not technical details.

Pragmatic:

```
orders.online.store123.order171711
```

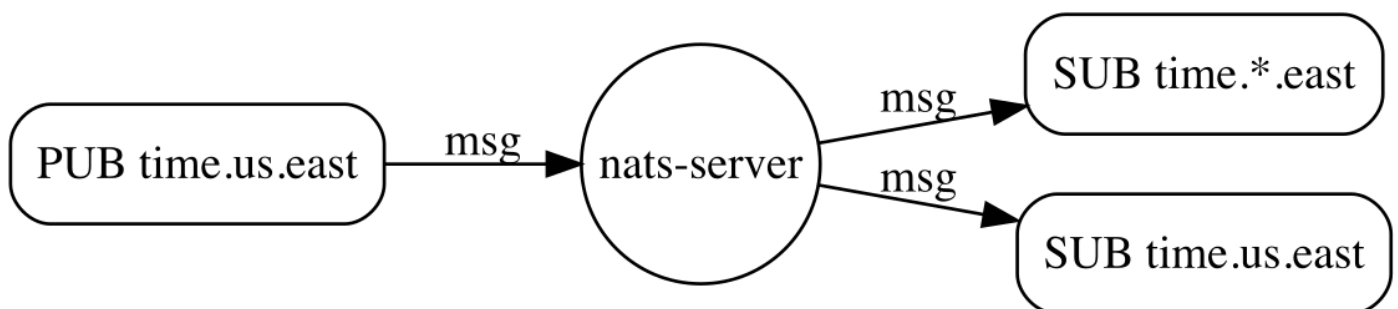
Maybe not so useful:

```
orders.online.us.server42.ccpayment.premium.store123.electronics.deliver-  
dhl.order171711.create
```

- NATS messages support headers. These can be used for additional metadata. There are subscription modes, which deliver headers only, allowing for efficient scanning of metadata in the message flow.

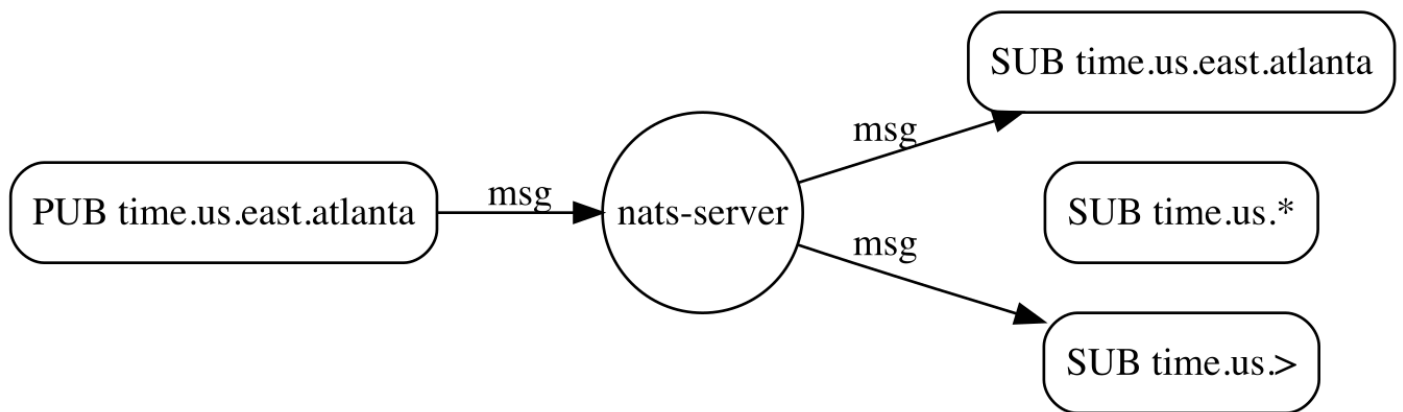
Matching a single token

The first wildcard is `*` which will match a single token. For example, if an application wanted to listen for eastern time zones, they could subscribe to `time.*.east`, which would match `time.us.east` and `time.eu.east`. Note that `*` can not match a substring within a token `time.New*.east`.



Matching multiple tokens

The second wildcard is `>` which will match one or more tokens, and can only appear at the end of the subject. For example, `time.us.>` will match `time.us.east` and `time.us.east.atlanta`, while `time.us.*` would only match `time.us.east` since it can't match more than one token.



Monitoring and wire taps

Subject to your security configuration, wildcards can be used for monitoring by creating something called a *wire tap*. In the simplest case, you can create a subscriber for `>`. This application will receive all messages -- again, subject to security settings -- sent on your NATS cluster.

Mixing wildcards

The wildcard `*` can appear multiple times in the same subject. Both types can be used as well. For example, `*.*.east.>` will receive `time.us.east.atlanta`.

Characters allowed and recommended for subject names

For compatibility across clients and ease of maintaining configuration files, we recommend using alphanumeric characters, `-` (dash) and `_` (underscore) ASCII characters for subject and other entity names created by the user.

UTF-8 (UTF8) characters are supported in subjects. Please use UTF-8 characters at your own risk. Using multilingual names for technical entities can create many issues for editing, configuration files, display, and cross-border collaboration.

The rules and recommendations here apply to ALL system names, subjects, streams, durables, buckets, keys (in key-value stores), as NATS will create API subjects that contain those names. NATS will enforce these constraints in most cases, but we recommend not relying on this.

- **Allowed characters:** Any Unicode character except `null`, space, `.`, `*` and `>`
- **Recommended characters:** (`a` - `z`), (`A` - `Z`), (`0` - `9`), `-` and `_` (names are case sensitive, and cannot contain whitespace).
- **Naming Conventions** If you want to delimit words, use either PascalCase as in `MyServiceOrderCreate` or `-` and `_` as in `my-service-order-create`
- **Special characters:** The period `.` (which is used to separate the tokens in the subject) and `*` and also `>` (the `*` and `>` are used as wildcards) are reserved and cannot be used.
- **Reserved names:** By convention subject names starting with a `$` are reserved for system use (e.g. subject names starting with `$SYS` or `$JS` or `$KV`, etc...). Many system subjects also use `_` (underscore) (e.g. `_INBOX` , `KV_ABC`, `OBJ_XYZ` etc.)

Good names

```
time.us
time.us2.east1
time.new-york
time.SanFrancisco
```

Deprecated subject names

```
location.Malmö
$location.Stockholm
_Subjects_.mysubject
```

Forbidden stream names

```
all*data
<my_stream>
service.stream.1
```

Pedantic mode

By default, for the sake of efficiency, subject names are not verified during message publishing. In particular, when generating subjects programmatically, this will result in illegal subjects which cannot be subscribed to. E.g. subjects containing wildcards may be ignored.

To enable subject name verification, activate `pedantic` mode in the client connection options.

```
//Java
Options options = Options.builder()
    .server("nats://127.0.0.1:4222")
    .pedantic()
    .build();
```

```
Connection nc = Nats.connect(options)
```

```
{} #4
```

```
{} cocobook-sync {} 2026-10-28 11:10:07 UTC
```

```
{} cocobook-sync {} 2026-10-29 03:58:09 UTC
```