

Authorization



- [Permissions Configuration Map](#)
 - [Permission Map](#)
 - [Allow Responses Map](#)
 - [Examples](#)
 - [Variables](#)
 - [Allow/Deny Specified](#)
 - [allow_responses](#)
 - [Queue Permissions](#)
-

The NATS server supports authorization using subject-level permissions on a per-user basis. Permission-based authorization is available with multi-user authentication via the `users` list.

Each permission specifies the subjects the user can publish to and subscribe to. The parser is generous at understanding what the intent is, so both arrays and singletons are processed. For more complex configuration, you can specify a `permission` object which explicitly allows or denies subjects. The specified subjects can specify wildcards as well. Permissions can make use of [variables](#).

A special field inside the authorization map is `default_permissions`. When present, it contains permissions that apply to users that do not have permissions associated with them.

Permissions Configuration Map

The `permissions` map specify subjects that can be subscribed to or published by the specified client.

Property	Description
<code>publish</code>	subject, list of subjects, or permission map the client can publish

Property	Description
<code>subscribe</code>	subject, list of subjects, or permission map the client can subscribe to. In this context it is possible to provide an optional queue name: <code><subject> <queue></code> to express queue group permissions. These permissions can also use wildcards such as <code>v2.*</code> or <code>></code> .
<code>allow_responses</code>	boolean or responses map , default is <code>false</code> . Enabling this implicitly denies publish to other subjects, however an explicit <code>publish</code> allow on a subject will override this implicit deny for that subject.

Permission Map

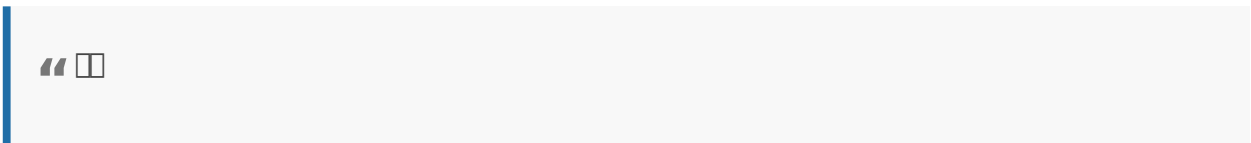
The `permission` map provides additional properties for configuring a `permissions` map. Instead of providing a list of allowable subjects and optional queues, the `permission` map allows you to explicitly list those you want to `allow` or `deny`. Both lists can be provided. In case of overlap `deny` has priority.

Property	Description
<code>allow</code>	List of subject names that are allowed to the client
<code>deny</code>	List of subjects that are denied to the client

Important Note It is important to not break request-reply patterns. In some cases (as shown [below](#)) you need to add rules for the `_INBOX.>` pattern. If an unauthorized client publishes or attempts to subscribe to a subject that has not been *allow listed*, the action fails and is logged at the server, and an error message is returned to the client. The [allow responses](#) option can simplify this.

Allow Responses Map

The `allow_responses` option dynamically allows publishing to reply subjects and is designed for [service](#) responders. When set to `true`, an implicit *publish allow* permission is enforced which enables the service to have temporary permission to publish to the `reply` subject during a request-reply exchange. If `true`, the client supports a one-time `publish`. If `allow_responses` is a map, it allows you to configure a maximum number of responses and how long the permission is valid.



Note, when `allow_responses` is enabled, the reply subject is not constrained to the `publish` allow or deny list. The implication of this is that a reply subject can be provided by a client to a service (responder) that does not have permission to explicitly publish on that subject, but is temporarily allowed given this option. If explicit control over which subjects a client is allowed to reply to, do not use `allow_responses` and instead define allow/deny lists under the `publish` permission map.

Property	Description
<code>max</code>	The maximum number of response messages that can be published.
<code>expires</code>	The amount of time the permission is valid. Values such as <code>1s</code> , <code>1m</code> , <code>1h</code> (1 second, minute, hour) etc can be specified. Default doesn't have a time limit.

When `allow_responses` is set to `true`, it defaults to the equivalent of `{ max: 1 }` and no time limit.

Important Note When using `nsc` to configure your users, you can specify the `--allow-pub-response` and `--response-ttl` to control these settings.

Examples

Variables

Here is an example authorization configuration that uses *variables* which defines four users, three of whom are assigned explicit permissions.

```
authorization {
  default_permissions = {
    publish = "SANDBOX.*"
    subscribe = ["PUBLIC.>", "_INBOX.>"]
  }
  ADMIN = {
    publish = ">"
    subscribe = ">"
  }
  REQUESTOR = {
    publish = ["req.a", "req.b"]
    subscribe = "_INBOX.>"
  }
  RESPONDER = {
    subscribe = ["req.a", "req.b"]
  }
}
```

```

publish = "_INBOX.>"
}
users = [
  {user: admin, password: $ADMIN_PASS, permissions: $ADMIN}
  {user: client, password: $CLIENT_PASS, permissions: $REQUESTOR}
  {user: service, password: $SERVICE_PASS, permissions: $RESPONDER}
  {user: other, password: $OTHER_PASS}
]
}

```

“ `default_permissions` is a special entry. If defined, it applies to all users that don't have specific permissions set.

- *admin* has `ADMIN` permissions and can publish/subscribe on any subject. We use the wildcard `>` to match any subject.
- *client* is a `REQUESTOR` and can publish requests on subjects `req.a` or `req.b`, and subscribe to anything that is a response (`_INBOX.>`).
- *service* is a `RESPONDER` to `req.a` and `req.b` requests, so it needs to be able to subscribe to the request subjects and respond to clients that can publish requests to `req.a` and `req.b`. The reply subject is an inbox. Typically inboxes start with the prefix `_INBOX.` followed by a generated string. The `_INBOX.>` subject matches all subjects that begin with `_INBOX.`.
- *other* has no permissions granted and therefore inherits the default permission set.

“ Note that in the above example, any client with permission to subscribe to `_INBOX.>` can receive *all* responses published. More sensitive installations will want to add or subset the prefix to further limit subjects that a client can subscribe. Alternatively, [Accounts](#) allow complete isolation limiting what members of an account can see.

Allow/Deny Specified

Here's an example without variables, where the `allow` and `deny` options are specified:

```

authorization: {
  users = [
    {
      user: admin

```

```

        password: secret
        permissions: {
            publish: ">"
            subscribe: ">"
        }
    }
    {
        user: test
        password: test
        permissions: {
            publish: {
                deny: ">"
            },
            subscribe: {
                allow: "client.>"
            }
        }
    }
}
]
}

```

allow_responses

Here's an example with `allow_responses`:

```

authorization: {
    users: [
        { user: a, password: a },
        { user: b, password: b, permissions: {subscribe: "q", allow_responses: true } },
        { user: c, password: c, permissions: {subscribe: "q", allow_responses: { max: 5,
expires: "1m" } } }
        { user: d, password: d, permissions: {subscribe: "q", publish: "x", allow_responses:
true } }
    ]
}

```

- User `a` has no restrictions.
- User `b` can listen on `q` for requests and can only publish once to reply subjects. *All other publish subjects are denied implicitly when `allow_responses` is set.*

- User `c` can listen on `q` for requests, but is able to return at most 5 reply messages, and the reply subject can be published at most for `1` minute.
- User `d` has the same behavior as user `b`, except that it can explicitly publish to subject `x` as well, which overrides the implicit deny from `allow_responses`.

Queue Permissions

User `a` can only subscribe to `foo` as part of the queue subscriptions `queue`. User `b` has permissions for queue subscriptions as well as plain subscriptions. You can allow plain subscriptions on `foo` but constrain the queues to which a client can join, as well as preventing any service from using a queue subscription with the name `*.prod`:

```
users = [
  {
    user: "a", password: "a", permissions: {
      sub: {
        allow: ["foo queue"]
      }
    }
  }
  {
    user: "b", password: "b", permissions: {
      sub: {
        # Allow plain subscription foo, but only v1 groups or *.dev queue groups
        allow: ["foo", "foo v1", "foo v1.>", "foo *.dev"]

        # Prevent queue subscriptions on prod groups
        deny: [ "> *.prod" ]
      }
    }
  }
]
```

📄 #3

📄 cocobook-sync 📄 2021-11-22 23:51:02 UTC

📄 cocobook-sync 📄 2025-08-22 01:43:03 UTC