Table of Contents

By default, this plugin renders the token in RFC 6750 Bearer Token format:

Sample Access token response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{

"access_token":"eyJhbGciOiJIUzI1NiJ9.eyJleHAiOjE0MjI5OTU5MjIsInN1YiI6ImppbWkiLCJyb
2xlcyI6WyJST0xFX0FETUlOIiwiUk9MRV9VU0VSIl0sImlhdCI6MTQyMjk5MjMyMn0.rA7A2Gwt14LaYMp
xNRtrCdO24RGrfHtZXY9fIjV8x8o",
    "token_type":"Bearer",
    "username": "john.doe",
    "roles": [
        "ROLE_ADMIN",
        "ROLE_USER"
    ]
}
```

As per the RFC, `access_token` and `token_type` property names cannot be customised.

The JSON structure can be customised with the following configuration keys:

Table 1. Token rendering configuration options

Config key	Default value
`grails.plugin.springsecurity.rest.token.rendering.usernamePropertyName`	`username`
`grails.plugin.springsecurity.rest.token.rendering.authoritiesPropertyName`	`roles`

E.g., with the following configuration:

```
grails.plugin.springsecurity.rest.token.rendering.usernamePropertyName = 'login'
grails.plugin.springsecurity.rest.token.rendering.authoritiesPropertyName =
'permissions'
```

The output will look like:

```
{

"access_token":"eyJhbGciOiJIUzI1NiJ9.eyJleHAiOjE0MjI5OTU5MjIsInN1YiI6ImppbWkiLCJyb
2xlcyI6WyJST0xFX0FETUlOIiwiUk9MRV9VU0VSIl0sImlhdCI6MTQyMjk5MjMyMn0.rA7A2Gwt14LaYMp
xNRtrCdO24RGrfHtZXY9fIjV8x8o",
    "token_type":"Bearer",
    "login": "john.doe",
    "permissions": [
        "ROLE_ADMIN",
        "ROLE_USER"
    ]
}
```

Disabling bearer tokens support for full response customisation

In order to fully customise the response, you need first to disable bearer tokens support by setting `grails.plugin.springsecurity.rest.token.validation.useBearerToken = false`. That will enable you to use this additional property:

Table 2. Token rendering configuration options

Config key	Default value
`grails.plugin.springsecurity.rest.token.rendering.tokenPropertyName`	`access_token`

Disabling bearer token support impacts the way tokens are extracted from the HTTP request. Please, read carefully the chapter about token validation first.

If you want your own implementation, simply create a class implementing AccessTokenJsonRenderer and wire it up in `resources.groovy` with name `accessTokenJsonRenderer`.

The principal object stored in the security context, and passed to the JSON renderer, is coming from the configured authentication providers. In most cases, this will be a `UserDetails` object retrieved using the `userDetailsService` bean.

If you want to render additional information in your JSON response, you have to:

1. Configure an alternative `userDetailsService` bean that retrieves the additional information you want, and put it in a principal object.
2. Configure an alternative `accessTokenJsonRenderer` that reads that information from the `restAuthenticationToken.principal` object.