Refresh Token Rotation (RTR)

FOLIO will implement refresh token rotation (RTR) as an important step in improving its overall security in Poppy. Within the Poppy and Quesnelia releases, the current token regime will be deprecated, and be fully replaced by the new RTR-based approach in Ramsons. Affected modules will need to upgrade to the new regime during the transition period. During the transition period, both the current and new methods of obtaining a token will be supported, but not after.

In the Ramsons release, legacy, non-expiring tokens will no longer be supported.

The epic for all of the work associated with RTR is here:

- FOLIO-3627 - Poppy 2023 R2 - Implement refresh token rotation (RTR) in all affected modules

The Core Platform team has implemented the changes to the core platform authentication and authorization modules to support RTR. RTR has been reviewed for security by Skott Klebe, and his report has been shared with the Core Platform team and the Security Council.

See the epic for a list of all the affected modules and associated stories for the work to upgrade to RTR.

This page is to serve as a guide for implementing RTR in modules and scripts and 3rd party integrations. It also provides configuration options for sys ops.

How RTR works

In RTR, the user provides a set of credentials (such as a username and password) and then receives two tokens back: a refresh token (RT) and an access token (AT). Both the RT and AT have a time to live (TTL) specified in the JWT claims (the properties of the JWT) as the \texttt{exp} claim. The user can provide the AT in subsequent requests to authorize access to a given resource so long as the AT hasn't expired. The user can provide the RT to get another AT when the current AT expires, which provides a new RT and AT set. This is the "refresh" part of RTR.

The cycle continues until the user logs out. Usually in RTR regimes the AT is short lived (in 10s of minutes) and the RT is longer lived (in days).

In the context of a web-browser, the user can stay logged in indefinitely (without a login interruption) so long as they come back to their browser within the TTL of RT since client code can work in the background to refresh the tokens and since our cookies will persist between sessions.

Single use

A common feature of RTR regimes is RT single use. A given RT may only be presented once to the authentication server (AS) to get a new AT/RT pair. If it is presented more than once this is considered a "leaked token" (a compromised or stolen token) and triggers a revocation of all current tokens associated with the given user. This has the effect of logging the user out on all of their devices and causes the user to have to log in again. It also allows for the compromise to be logged as a security event.

Keeping RTs safe

Because the RT allows you to get new ATs, it is important to keep RTs secure. This is usually accomplished by storing the tokens in the browser's cookie jar, with the \texttt{httpOnly} flag to keep the cookie isolated from JavaScript. Furthermore, RTs should only be used for the purpose of obtaining a new set of tokens; for regular API calls, only the AT is to be provided on the call.

The ability to revoke or invalidate tokens and having a TTL on tokens are both essential ingredients in an RTR regime.

How RTR will change FOLIO authentication and authorization

In the current FOLIO system, after logging in with a username and password, the response includes a non-expiring JWT token. Subsequent requests can include this token and the authorization server (mod-authtoken) will allow those requests to succeed provided the user has the correct permissions for the resource and provided the token is signed correctly. It will allow this token to succeed indefinitely, provided that it is signed by the current signing key of mod-authtoken.

This will change in the following ways:

- The RT and AT will have an expiration claim which will be checked on each use. If either one has expired it is rejected and a 403 is returned.
- The RT and AT will each be returned in their own \texttt{Set-Cookie} header only. No token is returned in the body of the response.
- The body of the response will contain the TTL of the RT and the AT in separate JSON properties to allow clients which cannot access the cookie \texttt{exp} claim (browser clients) to determine when the tokens need to be refreshed.
- The endpoint for logging in will change from \texttt{authn/login} to \texttt{authn/login-with-expiry}. For a transition period endpoints supporting the old and the new regime will co-exist. This arrangement will not be in a given FOLIO release however. Only the new endpoint will be supported once the transition is complete.
- A new \texttt{authn/refresh} endpoint is available for submitting an RT to get a new AT/RT pair and getting back a response similar to the one returned from login.
A brief guide for implementing RTR for module clients

Any module which relies on mod-login's authn/login endpoint should now use the newer authn/login-with-expiry endpoint. This will return the new tokens using the new Set-Cookie headers with a response body containing the TTL of each in ISO 8601 format.

It is the responsibility of each client to keep its RT and AT current. Clients can adopt different strategies to ensure that a valid AT is submitted with every request. These strategies depend on UX considerations and performance considerations.

Endpoints involved in RTR in mod-login:

- authn/login-with-expiry - Returns the RT and AT pair when given a username and password.
- authn/refresh - Returns a refreshed RT and AT pair when provided a valid RT.
- authn/logout - Invalidates the user's current RT not allowing it to be used anymore.
- authn/logout-all - Logs the user out on all of their devices (all of their RTs that are currently valid are now invalidated).

Please see the current implementation of the API (the RAML) the master branch of mod-login. Mod-users-bl also has its own login-with-expiry endpoint which is a proxy for mod-login's, but it provides a different response body (a CompositeUser object).

In general, a backend module need only concern itself with the login endpoints to get a new token when a token has or is about to expire. Refreshing the token via the refresh endpoint should only be needed for browser-based clients.

A guide for non-module clients such as scripts or other integrations

Any scripts or 3rd party integrations with FOLIO that relied on the legacy, non-expiring tokens will need to be changed to use the new authn/login-with-expiry endpoint. Unlike the legacy endpoint, the new endpoint only returns the AT in the form of a header in the HTTP response. To get the AT, client code needs to parse the cookie string to extract the AT since the cookie string contains other elements besides the token. What follows is an example of the cookie contents of a Set-Cookie header returned from the login function:

```
eyrzdW1loiJ02XN0a7nXx2FkbNlu7iwi7xdXhi19pZCI6IjczEtzEjgsYTF1L7kOZGMeNdkjS1lyjylLTM3YW1ONUoNm7iyi1sInR5cGUoi0iJhYfN1c3MiicCjleNai0jE207UsOTWwNTAxe11hdc16MTYJNTM5MjixwMCwiZGVzdEXpYjE0In0.
PqMCQ9ghQPKzdxolYdYd09tOnu0QT4C10ifNkmn2n0; Max-Age=600; Expires=Fri, 22 Sep 2023 14:30:10 GMT; Path=/; Secure; HTTPOnly; SameSite=None
Set-Cookie: folioRefreshToken=eyJlbmNlOiJLittleCQwijwiYXN0IjoiIZ0GlyIn0..hA3f_Sa2xV7V07n+p9Po0qtsgs0r48hKnAf5-M8JRRxXXhYJJ24eLCFNA8ee2EZ5PQ9P9EGktq9KHOFVsrQewLwXF04-
O2xlg6i8b63Nv8w7T8NmyqgWkAZg3UUXS81Qs3fYDgs6H7TV_c8hU77knHKGzo3YLbNv5-
Qs7en7EhzCE0y79k7hQFfUrgr7Y7bKixXxKkVRvsh7c+-n52S20T44-H-KC-vmkVn7yLu9W-pCCU4E37NFjBWVv9jV6NiOa-H-
92fF01p6wTHfprcKXZ4LVRAmsVJLxMQYdt02P7s01888.v6Pe4Wm-k0P8jmJaiZ32kQ; Max-Age=604800; Expires=Fri, 29 Sep 2023 14:20:10 GMT; Path=/authn; Secure; HTTPOnly; SameSite=None
```

Note that the response from login in contains two Set-Cookie headers, one for the RT and one for the AT. Client scripts only need concern themselves with the AT. If a new AT is needed, client code should request a new one by logging in again and not bother using the refresh endpoint.

The AT is the part after the folioAccessToken= and before the ; character.

![Warning icon] The default expiration of an AT is 10 minutes. If client code needs to use this token after this 10 minute period is up, client code should request a new AT by logging in again. Note that once the AT reaches the FOLIO system the AT is converted into a non-expiring token. So a long-running operation that takes more than 10 minutes won't be subject to the expiration of the original AT.

Configuration guide for sys ops

Systems operators have the ability to configure the TTLs on the AT and the RT. For configuration instructions please see mod-authn-token's `readme` here. Configuration can override the default expiration for the AT, the RT or both. Configuration can apply globally to the cluster or per tenant.

Depending on the host name of front-end and backend the SameSite attribute of the AT and RT cookie needs to be configured using the LOGIN_COOKIE_ SAMESITE environment variable of mod-login (README) and mod-login-saml (README).

For system operators who wish to take full advantage of the new more secure tokens, there is a new system property and environment variable called Legacy_TOKEN_TENANTS. When not set (the default), in Poppy, all tenants are legacy token tenants. When set to a list of tenant ids, only those tenants will be considered legacy token tenants. When set to an empty string, no tenants are legacy tenants. A legacy tenant is defined as a tenant for which the legacy endpoint authn/login will not return a 404.