ReaderAccountController

September 4, 2026

ReaderAccountController

Kind: Controller

Source: `atloria-monorepo/apps/api/src/reader-account/reader-account.controller.ts`

C3 reader-facing developer hub — the `p/:slugWithId/reader/⋯` family on the public surface (docs/architecture/reader-identity.md source 2 'account').

Same guard stack + ordering as PublicProjectController: OptionalJwtAuthGuard (member bypass for the A3 gate), ReaderAuthGuard (decode the ONE reader token → req.readerClaims), DocsAccessGuard (a gated project gates its dev hub too). ONLY the auth/start|verify|callback|confirm mint routes opt out via

`ReaderAccountController` serves reader-facing account routes under `p/:slugWithId/reader/...` for the public developer hub. It applies optional member authentication, reader-token decoding, and project documentation access checks in the same order as `PublicProjectController`, while allowing reader authentication minting flows to bypass the normal guard stack.

Diagram

```
graph LR
  Client[Reader browser or app] --> Route["/p/:slugWithId/reader/..."]

  Route --> OptionalJwtAuthGuard[OptionalJwtAuthGuard]
  OptionalJwtAuthGuard --> ReaderAuthGuard[ReaderAuthGuard]
  ReaderAuthGuard --> Claims["req.readerClaims"]
  Claims --> DocsAccessGuard[DocsAccessGuard]
  DocsAccessGuard --> AccountActions[Reader account actions]

  Client --> AuthRoutes["/reader/auth/start|verify|callback|confirm"]
  AuthRoutes --> MintReaderToken[Reader token minting flow]

  OptionalJwtAuthGuard -. Member bypass for A3 gate .-> DocsAccessGuard
```

Usage

```
// Example client call to begin a reader authentication flow.
// Auth minting routes intentionally opt out of the standard reader guard stack.
const response = await fetch(
  `/p/acme-platform--abc123/reader/auth/start`,
  {
    method: 'POST',
    headers: {
      'content-type': 'application/json',
    },
    body: JSON.stringify({
      email: 'reader@example.com',
    }),
  },
);

if (!response.ok) {
  throw new Error(`Unable to start reader authentication: ${response.status}`);
}

const result = await response.json();
console.log('Reader auth flow started:', result);
```

AI Coding Instructions

- Keep guard ordering aligned with `PublicProjectController`: `OptionalJwtAuthGuard`, then `ReaderAuthGuard`, then `DocsAccessGuard`.
- Treat `req.readerClaims` as the canonical decoded reader identity; do not decode or mint a second reader token in protected account handlers.
- Only reader authentication minting endpoints (`auth/start`, `auth/verify`, `auth/callback`, and `auth/confirm`) should opt out of the normal guard stack.
- Preserve the `p/:slugWithId/reader/...` route family so project resolution and gated developer-hub access remain consistent.
- Verify both member access and reader-token access paths when changing guards, decorators, or route-level authorization.

Relationships

- MODULE_DECLARES → `startMagicLink`
- MODULE_DECLARES → `verifyMagicLink`
- MODULE_DECLARES → `magicLinkCallback`
- MODULE_DECLARES → `magicLinkConfirm`

- MODULE_DECLARES → `usage`
- MODULE_DECLARES → `deleteAccount`
- MODULE_DECLARES → `listKeys`
- MODULE_DECLARES → `createKey`
- MODULE_DECLARES → `revokeKey`
- DEPENDS_ON → `ReaderMagicLinkService`
- DEPENDS_ON → `ReaderAccountService`
- DEPENDS_ON → `ReaderKeyService`

Referenced By

- `ReaderAccountModule` (MODULE_DECLARES)