Elmer A. Sia Follow
Last edited by Elmer A. Sia on Jul 2, 2025
Contributors to

Bibendo API Documentation Enhancement Proposal

1. Overview

During the migration and improvement of Bibendo's API documentation for Readme.com and OpenAPI 3.0 compliance, we identified additional opportunities to enhance the quality, clarity, and developer experience of the API Reference.

This document summarizes key areas of improvement and proposes next steps.

2. Global Improvement Areas

- Authentication handled via Firebase tokens — confirmed, but better standardization of `securitySchemes` could be reinforced.
- Inconsistent HTTP status code usage (`200 OK` vs `204 No Content`) across similar operations.
- Certain `GET` endpoints accepting request bodies — non-standard in REST guidelines.
- Schema-level inconsistencies (e.g., missing field descriptions).
- Opportunities to improve developer onboarding with more examples and consistent error messaging.

Duplicate or Redundant Endpoints

Observation: During documentation we found that `/game/{gameId}` and `/game/{gameId}/unauth` appear to expose nearly identical data, but with slight differences in authentication and request body expectations. The authenticated version uses an `EnhancedUser` payload while the unauthenticated one does not.

Why This Matters: Clarifying endpoint purpose and overlap helps developers choose the right call for their use case, avoids confusion in front-end implementation, and ensures cleaner maintenance for future versions.

Suggested Next Step: Confirm whether both versions are needed. If redundant, deprecate the less clear variant and update documentation to direct developers to the correct, canonical path.

Consistent Use of Request Body in `GET` and `DELETE`

Pattern Observed:

- Some `GET` endpoints accept optional request bodies (e.g., `EnhancedUser`).
- Some `DELETE` endpoints, like `/game/{gameId}` , do as well — but inconsistently, and sometimes they don't do anything meaningful with it.
- This is a REST anti-pattern that can confuse both developers and OpenAPI tooling.

Proposed Improvement:

- Audit all `GET` and `DELETE` methods that have request bodies.
- Remove unused or redundant bodies.
- Shift context to query parameters if necessary, for REST-aligned calls.

Ambiguous Scopes for Player vs User

Pattern Observed:

- We flagged during documentation that `Player` and `User` are not consistently scoped or defined.
- For example, `Run Player Management` vs `Player Contacts` vs `User Management` — the boundary of what a player is versus a user with broader platform privileges remains hazy.

Proposed Improvement:

- Add clear definitions to the API's Getting Started or Concepts guide:
- What is a User?
- What is a Player?
- What data do they share?
- Where are they distinct?
- Add consistent tags and path naming to reinforce that boundary.

Parameter Consistency

Observation:
Some endpoints (especially legacy ones) include OpenAPI-specific parameter fields like `style: simple` and `explode: false` . However, in modern OpenAPI 3.0 usage, these defaults are implied for path parameters and do not need to be explicitly defined. Removing redundant fields reduces visual clutter, avoids confusion, and improves spec readability.

Proposed Improvement:

- Clean up redundant `style` and `explode` fields where the default behavior applies.
- Apply this as a convention for any future endpoints added to the OAS file.
- Consider a linting pass using tools like Spectral to enforce parameter style consistency.

3. Targeted Resource Improvements

3.1. Accounts API Enhancements

3.1.1. Schema Refinements

Schema	Issue Identified	Proposed Improvement
`Account`	No field descriptions; sensitive fields exposed	Add clear field descriptions; consider masking or flagging sensitive fields (`password` , `notPasswordFalse`)
`EnhancedUser`	No descriptions; structure unclear for devs	Add descriptions; validate necessity of fields
`JsonMap`	Undefined structure; lacks examples	Define expected key-value pairs; provide example
`Password`	No password complexity enforced	Add `minLength` , regex pattern validation to `password` field
`AccountList`	Fields like `error` , `type` lack descriptions	Add meaningful field-level descriptions to improve API consumer clarity
`Invitation`	Missing descriptions; vague structure	Add descriptions; suggest providing examples or value hints for `Invitation` objects

3.1.2. Endpoint Enhancements

Endpoint	Issue	Proposed Improvement
`DELETE /account/{fullId}`	Returns full object after delete	Suggest `204 No Content` or lightweight deletion metadata
`GET /account/{fullId}`	`GET` with required body (non-RESTful)	Suggest reworking to avoid body in GET
`GET /accounts/{fullIds}`	`GET` with required body (non-RESTful)	Suggest reworking to avoid body in GET
`POST /account/setPw/{token}`	Password field lacks validation	Add stricter password validation in schema
`POST /account/updateOnce`	Returns `200 OK` without content	Suggest `204 No Content`
`GET /account/deleteMe`	Returns `200 OK` without content	Suggest `204 No Content`
`POST /account/resetPw/{email}`	Returns `200 OK` without content	Suggest `204 No Content`
`DELETE /account/remove/{id}`	Returns `200 OK` without content	Suggest `204 No Content`
`POST /account/{fullId}/expiration/{dateExpect}`	Returns `200 OK` with no response body	Suggest `204 No Content`
`POST /account/create`	Request body not strictly required	Evaluate enforcing non-empty request body
`POST /account/update`	Request body not strictly required	Evaluate enforcing non-empty request body

3.2. Media API Enhancements

3.1.1. Schema Refinements

Schema	Issue Identified	Proposed Improvement
`MediaLibraryFile`	Lacks field descriptions	Add field-level descriptions for clarity
`CollectionResponse_MediaLibraryFile`	Minimal pagination metadata	Add pagination fields like `totalCount` , `pageSize` to enhance UX

3.1.2. Endpoint Enhancements

Endpoint	Issue	Proposed Improvement
`DELETE /media/{assetId}`	Returns `200 OK` with no content	Suggest `204 No Content`
`GET /media/list`	Pagination only via `nextPageToken`	Consider enhancing pagination metadata
`POST /media`	No field-level validation in schema	Add stricter validation to `MediaLibraryFile` fields if necessary

3.3. Organization API Enhancements

3.3.1. Schema Refinements

Schema	Issue Identified	Proposed Improvement
`Organization`	No field-level descriptions	Add descriptions for `id` , `name` , `expirationDate`
`CollectionResponse_Organization`	`nextPageToken` lacks clarification	Clarify meaning (pagination cursor?) and add field descriptions

3.3.2. Endpoint Enhancements

Endpoint	Issue	Proposed Improvement
`DELETE /organization/{id}`	Returns full object after delete	Suggest `204 No Content` or lightweight deletion metadata
`GET /organization/list`	Returns large lists without pagination hints	Add query parameters `limit` , `cursor` for pagination
`GET /organization/{id}/games`	No query parameters for pagination	Add `limit` , `cursor` support for scalable responses
`POST /organization/{organizationId}/reup/{expirationDate}`	No input validation on date	Add schema constraints (e.g., expiration date must be in the future)

4. Cross-Cutting Developer Experience Enhancements

The following recommendations apply across all resources and endpoints to ensure a consistent, developer-friendly experience throughout the Bibendo API. These enhancements aim to improve clarity, usability, and maintainability, aligning the API with modern best practices.

- **Add Example Responses**: Ensure all endpoints and schemas include realistic example responses to improve developer understanding and enable better SDK generation.
- **Field-Level Schema Descriptions**: Add concise, developer-friendly descriptions for all schema fields to improve API documentation readability and usability.
- **Standardize HTTP Status Codes**:
  - Use `204 No Content` where applicable for successful `DELETE` or non-content `POST` operations.
  - Ensure consistent use of `200 OK` only when response payloads are returned.
- **Consistent Authentication Documentation**:
  - Standardize the use of `securitySchemes` across all secured endpoints.
  - Clearly document authentication flows, expected tokens, and scopes where relevant.
- **Avoid Request Bodies in GET Requests**:
  - Review endpoints using request bodies in `GET` methods and migrate to `POST` where necessary for REST compliance.
- **Optimize DELETE Responses**:
  - Return lightweight deletion metadata (e.g., `{ "id": 12345, "deleted": true }`) or no content (`204 No Content`) instead of full resource objects on `DELETE` operations.
- **Pagination and Large Payload Handling**:
  - Introduce optional pagination parameters (e.g., `limit` , `cursor` , `nextPageToken`) on list endpoints to avoid large, unmanageable payloads.
  - Clarify usage of fields like `resumptionToken` for consistent pagination behavior.
- **Error and Metadata Field Clarifications**:
  - Add clear documentation for vague fields such as `error` , `errorCode` , and `type` across schemas to improve API consumer trust and clarity.

5. Proposed Deliverables

- Full audit of all schemas referenced in the OpenAPI spec.
- Field-level improvements:
  - Descriptions for all properties.
  - Naming consistency corrections.
  - Sensitive field review (security hygiene).
- Addition of example responses where applicable.
- Improved inline parameter descriptions.
- Delivery of a fully enhanced, developer-friendly OpenAPI 3.0.1 spec, ready for future scaling and SDK generation.

6. Estimated Timeline and Pricing

- **Timeline**: depends on the scope
- **Pricing**: depends on the scope

7. Benefits to Bibendo

- Improved onboarding time for developers integrating with the API.
- Reduced support requests due to better self-service documentation.
- Cleaner path for auto-generating SDKs and future API maintenance.
- Improved developer trust and brand perception.

8. Next Steps

Here's what we suggest to keep the momentum going:

1. **Quick Review**
   - Take a look at the ideas and improvements we've outlined.
   - Let us know what looks good and if you have any tweaks in mind.
2. **Confirm & Kick Off**
   - Once you're happy with the plan, we'll get started on the improvements.
   - We can adjust the scope or timeline based on what works best for you.
3. **Ongoing Collaboration**
   - We'll share updates as we go — no surprises, just steady progress.
   - Happy to adjust based on your feedback along the way.
4. **Final Touches & Handoff**
   - Once everything is polished, we'll deliver the updated OpenAPI spec.
   - We'll also make sure everything's smooth in Readme and hand over the files ready for future updates.

🚀 Ready to dive in as soon as we get the thumbs-up!