API Design Rules Module: Pagination 1.0.0

Logius Guide
Consultation version November 05, 2026

This version:
https://gitdocumentatie.logius.nl/publicatie/api/mod-pagination/1.0.0/

Latest published version:
https://gitdocumentatie.logius.nl/publicatie/api/mod-pagination/

Latest editor's draft:
https://logius-standaarden.github.io/API-mod-pagination/

Previous version:
https://gitdocumentatie.logius.nl/publicatie/api/mod-pagination//

Editor:
Logius Standaarden (Logius)

Author:
Logius Standaarden (Logius)

Participate:
GitHub Logius-standaarden/API-mod-pagination
All issues
File an issue
Commit history
Pull requests

This document is also available in these non-normative format: PDF

This document is licensed under
Creative Commons Attribution 4.0 International Public License

Status of This Document

This is a proposed recommendation approved by TO. Comments regarding this document may be sent to api@logius.nl

Table of Contents

Status of This Document

Conformance

Abstract

1. Introduction
1.1 Static collections
1.2 Dynamic collections
1.3 Pagination formats

2. Design rules
2.1 Summary
2.2 Rules

A. Linter configuration

B. References
B.1 Normative references

§Conformance

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words *MAY*, *MUST*, and *SHOULD* in this document are to be interpreted as described in BCP 14 [*RFC2119*] [*RFC8174*] when, and only when, they appear in all capitals, as shown here.

Abstract

This document is part of the *Nederlandse API Strategie*.

The Nederlandse API Strategie consists of a set of distinct documents.

Status	Description & Link
Informative	Inleiding NL API Strategie
Informative	Architectuur NL API Strategie
Informative	Gebruikerswensen NL API Strategie
Normative	NLgov REST API Design Rules (ADR v2.2)
Normative	Open API Specification (OAS 3.0)

Status	Description & Link
Normative	NLgov Assurance profile for OAuth 2.0
Normative	Digikoppeling REST API koppelvlak specificatie
Normative module	GEO module v1.0

Before reading this document it is advised to gain knowledge of the informative documents, in particular the Architecture.

This document describes the *Pagination module*, containing rules for paginating results from large datasets.

1. Introduction

This document is a module as part of the *API Design Rules*.

Resource collections can be unreasonably large for a single response. With pagination, a segment of the resources can be returned while providing a method to retrieve other segments (or "pages"). This module provides design rules which attempt to strike a balance between enhanced predictability for clients and flexibility to suit different data collection types.

A distinction between two categories of collections is used throughout this module: static and dynamic.

1.1 Static collections

Static collections are stable or at least unlikely to change while a client is retrieving segments of the resources. Page-number pagination would suffice for these collections. If no mutation occurred between retrieving pages, sequential pages will have no gaps or overlap, e.g. if item *n* is the last of a page, item *n+1* will be the first item on the next page.

If a resource collection is not completely static, no pagination method is reliable for synchronisation.

§1.2 Dynamic collections

Dynamic collections see frequent mutations. While a client jumps from one page to the next, an item may have been inserted or removed in the previous pages, causing a shift that results in an overlap or a gap, respectively. At the cost of complexity, cursor pagination is more robust by providing a pointer to a listed item for page boundaries which remain consistent even if the collection shifts.

§1.3 Pagination formats

In order to support both collection categories, this module allows two pagination formats: page-number pagination (using `page` and `pageSize`) and cursor pagination (using `cursor` and `limit`).

§2. Design rules

§2.1 Summary

Design rules are technical rules, which should be tested automatically, and functional rules, which should be considered when designing and building the API.

List of technical rules

- /pagination/format: Use standard pagination format
- /pagination/limits: Document the page size limits
- /pagination/links: Use a Link header for navigation

List of functional rules

- /pagination/sorting: Use deterministic ordering in paginated responses

§2.2 Rules

Technical

/pagination/format: Use standard pagination format

Statement

A paginated request *MUST* use one (and only one) pagination method from the following list:

- Page-number pagination using query keys `page` and `pageSize`:
  - `page`: requested page number (1-based)
  - `pageSize`: maximum number of items to return per page
- Cursor pagination using query keys `cursor` and `limit`:
  - `cursor`: opaque pointer to current position in dataset
  - `limit`: maximum number of items to return per page

> NOTE
>
> An endpoint *MAY* define additional aliases such as, but not limited to, `size` or `offset` for backwards compatibility.

Rationale

The defined formats cover the main usage patterns of pagination with the most commonly used names for query keys.

How to test

Technical

/pagination/limits: Document the page size limits

Statement

The OpenAPI definition document *MUST* include a default value for the returned page size (`pageSize` or `limit`) through the query parameter's `schema`, with `integer` as `type` and a `default` key.

The `schema` *SHOULD* also include a `maximum` value. A requested value larger than the maximum *MUST* result in HTTP status code `400` (Bad Request).

EXAMPLE 1: Page size limits in OAS

```
{
  "name": "pageSize",
  "in": "query",
  "schema": {
    "type": "integer",
    "default": 20,
    "maximum": 100
  }
}
```

Rationale

By documenting a default, the client can rely on that limit without having to request one or infer it from the presence of a link to another page.

By documenting a maximum, the client can avoid sending an unreasonable request.

How to test

This rule can be automatically checked and an example test is shown in the linter configuration.

Technical

/pagination/links: Use a Link header for navigation

Statement

A paginated response *MUST* include an HTTP `Link` header [*RFC8288*] for navigation to other result segments according to the following list:

- Page-number pagination *MUST* use link relation types `first`, `prev`, and `next`, and *MAY* use `last`.
- Cursor pagination *MUST* use link relation type `next` and *MAY* use `prev`.

A link relation that has no result (e.g. `prev` on the first page) *MUST* be omitted.

NOTE

The link relation types are part of the Link Relations registry maintained by IANA.

Rationale

A `Link` header provides clients with a convenient and consistent method of navigation.

The link relation type `last` is optional, as determining it requires counting the collection, which can be a costly endeavour.

Dynamic collections, like feeds, can experience mutations at the head. These can cause inconsistency with page alignment when reached with `prev`. Therefore, the backwards link is merely optional for cursor pagination.

An absent link relation allows the client to recognise the limits of the collection, without having to resolve a value without a target.

How to test

Functional

/pagination/sorting: Use deterministic ordering in paginated responses

Statement

Items in a paginated response *MUST* use deterministic ordering.

Rationale

If the order can change beyond the influence of the client, items are likely to be missed or appear multiple times while traversing pages.

A. Linter configuration

This section is non-normative.

▶ Details

Open the above details to inspect the linter configuration. You can also open the configuration file in separate tab

B. References

§B.1 Normative references

[ADR]

API Design Rules. Jasper Roes; Joost Farla. Logius. URL: https://gitdocumentatie.logius.nl/publicatie/api/adr/

[RFC2119]

Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF. March 1997. Best Current Practice. URL: https://www.rfc-editor.org/info/rfc2119/

[RFC8174]

Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. B. Leiba. IETF. May 2017. Best Current Practice. URL: https://www.rfc-editor.org/info/rfc8174/

[RFC8288]

Web Linking. M. Nottingham. IETF. October 2017. Proposed Standard. URL: https://httpwg.org/specs/rfc8288.html

↥