Package 'notionapi'

September 2, 2025

Title Client for the 'Notion API'

Version 0.1.0

Description Enable programmatic interaction with 'Notion' pages, databases, blocks, comments, and users through the 'Notion API' <https://developers.notion.com/>. Provides both synchronous and asynchronous client interfaces for building workflows and automations that integrate with 'Notion' workspaces. Supports all 'Notion API' endpoints including content creation, data retrieval, and workspace management.

License MIT + file LICENSE

URL https://brenwin1.github.io/notionapi/, https://github.com/brenwin1/notionapi

BugReports https://github.com/brenwin1/notionapi/issues

Imports cli, httr2, jsonlite, R6, rlang

Suggests promises, testthat (>= 3.0.0), vcr (>= 2.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Brenwin Ang [aut, cre, cph]

Maintainer Brenwin Ang <brenwinalj@gmail.com>

Repository CRAN

Date/Publication 2025-09-02 21:10:02 UTC

Contents

`BlocksChildrenEndpoint`

R6 Class for Blocks children endpoint

Description

Handle all block children operations in the Notion API

Note: Access this endpoint through the client instance, e.g., `notion$blocks$children`. Not to be instantiated directly.

Value

A list containing the parsed API response.

Methods

Public methods:

- `BlocksChildrenEndpoint$new()`
- `BlocksChildrenEndpoint$retrieve()`
- `BlocksChildrenEndpoint$append()`

Method `new()`**:** Initialise block children endpoint. Not to be called directly, e.g., use `notion$pages$children` instead.

Usage:

```
BlocksChildrenEndpoint$new(client)
```

Arguments:

`client` Notion Client instance

Method `retrieve()`**:** Retrieve a block's children

Usage:

```
BlocksChildrenEndpoint$retrieve(block_id, start_cursor = NULL, page_size = 100)
```

Arguments:

`block_id` String (required). The ID for a Notion block.

`start_cursor` Character. For pagination. If provided, returns results starting from this cursor. If NULL, returns the first page of results.

`page_size` Integer. Number of items to return per page (1-100). Defaults to 100.

Details: Endpoint documentation

Method `append()`**:** Append block children

Usage:

```
BlocksChildrenEndpoint$append(block_id, children, after = NULL)
```

Arguments:

`block_id` String (required). The ID for a Notion block.

`children` List of lists (JSON array) (required). Block objects to append as children to the block.

`after` Character. The ID of the existing block after which the new children are appended.

Details: Endpoint documentation

Examples

```
notion <- notion_client()

# ----- append children to a block

notion$blocks$children$append(
  block_id = "23933ea0-c1e4-81d6-a6f6-dd5b57ad4aba",
  children = list(
    # add a level 2 heading called "Test Heading"
    list(
      object = "block",
      heading_2 = list(
        rich_text = list(list(
          text = list(content = "Test Heading")
        ))
      )
    )
  )
)


# ----- retrieve children of a block

notion$blocks$children$retrieve("23933ea0-c1e4-81d6-a6f6-dd5b57ad4aba")


# ----- iterate through paginated results
## Not run:
start_cursor <- NULL
has_more <- FALSE
resps <- list()
i <- 1

while (has_more) {
  resps[[i]] <- notion$blocks$children$retrieve(
    "2926b407e3c44b49a1830609abe6744f",
    start_cursor
```

```
  )
  has_more <- resps[[i]][["has_more"]]
  start_cursor <- resps[[i]][["next_cursor"]]
  i <- i + 1
}

## End(Not run)
```

`BlocksEndpoint` *R6 Class for Blocks Endpoint*

Description

Handle all block operations in the Notion API

Note: Access this endpoint through the client instance, e.g., `notion$blocks`. Not to be instantiated directly.

Value

A list containing the parsed API response.

Public fields

`children` Block children endpoint

Methods

Public methods:

- `BlocksEndpoint$new()`
- `BlocksEndpoint$retrieve()`
- `BlocksEndpoint$update()`
- `BlocksEndpoint$delete()`

Method `new()`**:** Initialise block endpoint. Not to be called directly, e.g., use `notion$blocks` instead.

Usage:

```
BlocksEndpoint$new(client)
```

Arguments:

`client` Notion Client instance

Method `retrieve()`**:** Retrieve a block

Usage:

```
BlocksEndpoint$retrieve(block_id)
```

Arguments:

`block_id` Character (required). The ID for a Notion block.

Details: Endpoint documentation

Method `update()`**:** Update a block

Usage:

```
BlocksEndpoint$update(block_id, archived = FALSE, ...)
```

Arguments:

`block_id` Character (required). The ID for a Notion block.

`archived` Boolean. Set to TRUE to archive (delete) a block. Set to FALSE to unarchive (restore) a block. Defaults to FALSE.

`...` <dynamic-dots> Block-specific properties to update. Each argument should be named after a block type (e.g., `heading_1`, `paragraph`) with a named list value containing the block configuration.

Details: Endpoint documentation

Method `delete()`**:** Delete a block

Usage:

```
BlocksEndpoint$delete(block_id)
```

Arguments:

`block_id` Character (required). The ID for a Notion block.

Details: Endpoint documentation

Examples

```
notion <- notion_client()

# ----- retrieve a block
notion$blocks$retrieve("23933ea0-c1e4-81dc-9f56-f3fa251a757f")


# ----- update a block

notion$blocks$update(
  "23933ea0-c1e4-81dc-9f56-f3fa251a757f",
  heading_2 = list(
    rich_text = list(
      list(
        text = list(
          content = "Updated Test Heading"
        )
      )
    )
  )
)
```

```
# ----- delete a block

notion$blocks$delete(block_id = "23933ea0-c1e4-81d6-a6f6-dd5b57ad4aba")
```

`CommentsEndpoint` *R6 Class for Comments Endpoint*

Description

Handle all comments operations in the Notion API

Note: Access this endpoint through the client instance, e.g., `notion$comments`. Not to be instantiated directly.

Value

A list containing the parsed API response.

Methods

Public methods:

- `CommentsEndpoint$new()`
- `CommentsEndpoint$create()`
- `CommentsEndpoint$retrieve()`

Method `new()`**:** Initialise comments endpoint. Not to be called directly, e.g., use `notion$comments` instead.

Usage:

```
CommentsEndpoint$new(client)
```

Arguments:

`client` Notion Client instance

Method `create()`**:** Create a comment

Usage:

```
CommentsEndpoint$create(
  parent = NULL,
  discussion_id = NULL,
  rich_text,
  attachments = NULL,
  display_name = NULL
)
```

Arguments:

`parent` List (JSON object). The parent page where comment is created. Required if `discussion_id` is not provided

`discussion_id` Character. The ID of the discussion thread for the comment. Required if `parent` is not provided.

`rich_text` List of lists (JSON array) (required). Rich text object(s) representing the comment content.

`attachments` List of lists (JSON array). Attachments to include in the comment.

`display_name` Named list (JSON object). Custom display name of the comment.

Details: Endpoint documentation

Method `retrieve()`**:** Retrieve comments for a block

Usage:

```
CommentsEndpoint$retrieve(block_id, start_cursor = NULL, page_size = NULL)
```

Arguments:

`block_id` Character. The ID for a Notion block.

`start_cursor` Character. For pagination. If provided, returns results starting from this cursor. If NULL, returns the first page of results.

`page_size` Integer. Number of items to return per page (1-100). Defaults to 100.

Details: Endpoint documentation

Examples

```
notion <- notion_client()

# ----- create comment

notion$comments$create(
  list(page_id = "23933ea0-c1e4-81d6-a6f6-dd5b57ad4aba"),
  rich_text = list(list(
    text = list(
      content = "Hello world"
    )
  ))
)


# ----- retrieve comments

notion$comments$retrieve(block_id = "23933ea0-c1e4-81d6-a6f6-dd5b57ad4aba")
```

`DatabasesEndpoint` *R6 Class for Databases Endpoint*

Description

Handle all databases operations in the Notion API

Note: Access this endpoint through the client instance, e.g., `notion$databases`. Not to be instantiated directly.

Value

A list containing the parsed API response.

Methods

Public methods:

- `DatabasesEndpoint$new()`
- `DatabasesEndpoint$create()`
- `DatabasesEndpoint$query()`
- `DatabasesEndpoint$retrieve()`
- `DatabasesEndpoint$update()`

Method `new()`**:** Initialise databases endpoint. Not to be called directly, e.g., use `notion$databases` instead.

Usage:

```
DatabasesEndpoint$new(client)
```

Arguments:

`client` Notion Client instance

Method `create()`**:** Create a database

Usage:

```
DatabasesEndpoint$create(parent, title, properties, ...)
```

Arguments:

`parent` Named list (JSON object) (required). The parent page where the database will be created.

`title` List of lists (JSON array). Database title as an array of rich text objects.

`properties` Named list (JSON object) (required). The properties of the database as key-value pairs.

`...` <`dynamic-dots`> Additional body parameters to include in the request body.

Details: Endpoint documentation

Method `query()`**:** Query a database

Usage:

```
DatabasesEndpoint$query(
  database_id,
  filter_properties = NULL,
  filter = NULL,
  sorts = NULL,
  start_cursor = NULL,
  page_size = 100,
  ...
)
```

Arguments:

`database_id` String (required). The ID of a Notion database.

`filter_properties` Character vector. Property value IDs to include in the response schema.

`filter` Named list (JSON object). Filter conditions to apply to the query.

`sorts` List of lists (JSON array). Sort conditions to apply to the query.

`start_cursor` Character. For pagination. If provided, returns results starting from this cursor. If NULL, returns the first page of results.

`page_size` Integer. Number of items to return per page (1-100). Defaults to 100.

`...` Reserved for future use.

Details: Endpoint documentation

Method `retrieve()`**:** Retrieve a database

Usage:

```
DatabasesEndpoint$retrieve(database_id)
```

Arguments:

`database_id` String (required). The ID of a Notion database.

Details: Endpoint documentation

Method `update()`**:** Update a database

Usage:

```
DatabasesEndpoint$update(
  database_id,
  title = NULL,
  description = NULL,
  properties = NULL
)
```

Arguments:

`database_id` String (required). The ID of a Notion database.

`title` List of lists (JSON array). Database title as an array of rich text objects.

`description` List of lists (JSON array). Database description as an array of rich text objects.

`properties` Named list (JSON object). Database properties to update as key-value pairs.

Details: Endpoint documentation

Examples

```
notion <- notion_client()

# ----- create a database

notion$databases$create(
  parent = list(page_id = "23933ea0-c1e4-81d6-a6f6-dd5b57ad4aba"),
  title = list(
    list(
      type = "text",
      text = list(
        content = "Grocery list"
      )
    )
```

```
  ),
  properties = list(
    Name = list(
      title = no_config()
    ),
    `In stock` = list(
      checkbox = no_config()
    )
  )
)


# ----- retrieve a database

notion$databases$retrieve(
  "23933ea0-c1e4-8136-b37b-fa235c6f2a71"
)


# ----- update a database

notion$databases$update(
  "23933ea0-c1e4-8136-b37b-fa235c6f2a71",
  list(list(
    text = list(
      content = "Today's grocery list"
    )
  ))
)


# ----- query a database

notion$databases$query(
  database_id = "23933ea0-c1e4-8136-b37b-fa235c6f2a71",
  filter = list(
    or = list(
      list(
        property = "In stock",
        checkbox = list(equals = TRUE)
      ),
      list(
        property = "Name",
        title = list(contains = "kale")
      )
    )
  ),
  sorts = list(list(
    property = "Name",
    direction = "ascending"
  ))
)
```

```
# ----- iterate through paginated results
## Not run:
i <- 1
resps <- list()
has_more <- FALSE
start_cursor <- NULL

while (has_more) {
  resps[[i]] <- notion$databases$query(
    "22833ea0c1e481178e9cf1dcba79dbca",
    start_cursor = start_cursor
  )

  has_more <- resps[[i]][["has_more"]]
  start_cursor <- resps[[i]][["next_cursor"]]
  i <- i + 1
}

## End(Not run)
```

`notion_token_exists` *Check if Notion token is set*

Description

Checks if the `NOTION_TOKEN` environment variable is set.

Usage

```
notion_token_exists()
```

Value

`TRUE` if the token exists, `FALSE` otherwise.

Examples

```
notion_token_exists()
```

`no_config` *Create empty property configuration*

Description

Helper function that creates an empty named list for property configurations that require no additional settings.

Many database properties like text, checkbox and date do not need configuration settings. This function returns the empty configuration (`{}` in JSON) that these properties expect.

Usage

```
no_config()
```

Value

An empty `named list` that serialises to `{}` in JSON

Examples

```
notion <- notion_client()

# ----- create a database

notion$databases$create(
  parent = list(page_id = "23933ea0-c1e4-81d6-a6f6-dd5b57ad4aba"),
  title = list(
    list(
      type = "text",
      text = list(
        content = "Grocery list"
      )
    )
  ),
  properties = list(
    Name = list(
      title = no_config()
    ),
    `In stock` = list(
      checkbox = no_config()
    )
  )
)
```

PagesEndpoint *R6 Class for Pages Endpoint*

Description

Handle all pages operations in the Notion API

Note: Access this endpoint through the client instance, e.g., `notion$pages`. Not to be instantiated directly.

Value

A list containing the parsed API response.

Public fields

`properties` Pages properties endpoint

Methods

Public methods:

- `PagesEndpoint$new()`
- `PagesEndpoint$create()`
- `PagesEndpoint$retrieve()`

Method `new()`**:** Initialise pages endpoint. Not to be called directly, e.g., use `notion$pages` instead.

Usage:

```
PagesEndpoint$new(client)
```

Arguments:

`client` Notion Client instance

Method `create()`**:** Create a page

Usage:

```
PagesEndpoint$create(
  parent,
  properties,
  children = NULL,
  icon = NULL,
  cover = NULL
)
```

Arguments:

`parent` Named list (JSON object) (required). The parent page or database where the new page is inserted.

`properties` Named list (JSON object) (required). Key-value pairs representing the properties of the page.

`children` List of lists (JSON array). Block objects to append as children to the page.

`icon` Named list (JSON object). An icon for the page.

`cover` Named list (JSON object). A cover image for the page.

Details: Endpoint documentation

Method `retrieve()`**:** Retrieve page properties

Usage:

```
PagesEndpoint$retrieve(page_id, filter_properties = NULL)
```

Arguments:

`page_id` Character (required.). The ID for a Notion page.

`filter_properties` Character. Page property value IDs to include in the response schema. If NULL, all properties are returned.

Details: Endpoint documentation

Examples

```
notion <- notion_client()

# ----- create a page
notion$pages$create(
  list(page_id = "22f33ea0c1e480b99c77d1ab72aedff9"),
  list(
    title = list(list(
      text = list(
        content = "Test Page for notionapi"
      )
    ))
  )
)


# ----- retrieve a page

notion$pages$retrieve("23933ea0-c1e4-81d6-a6f6-dd5b57ad4aba")
```

`PagesPropertiesEndpoint`

R6 Class for Pages Properties Endpoint

Description

Handle all pages properties operations in the Notion API

Note: Access this endpoint through the client instance, e.g., `notion$pages$properties`. Not to be instantiated directly.

Value

A list containing the parsed API response.

Methods

Public methods:

- `PagesPropertiesEndpoint$new()`
- `PagesPropertiesEndpoint$retrieve()`
- `PagesPropertiesEndpoint$update()`

Method `new()`**:** Initialise pages properties endpoint. Not to be called directly, e.g., use `notion$pages$properties` instead.

Usage:

```
PagesPropertiesEndpoint$new(client)
```

Arguments:

`client` Notion Client instance

Method `retrieve()`**:** Retrieve a page property item

Usage:

```
PagesPropertiesEndpoint$retrieve(
  page_id,
  property_id,
  page_size = 100,
  start_cursor = NULL
)
```

Arguments:

`page_id` Character (required). The ID for a Notion page.

`property_id` Character (required). The ID of the property to retrieve.

`page_size` Integer. Number of items to return per page (1-100). Defaults to 100.

`start_cursor` Character. For pagination. If provided, returns results starting from this cursor. If NULL, returns the first page of results.

Details: Endpoint documentation

Method `update()`**:** Update a page property

Usage:

```
PagesPropertiesEndpoint$update(
  page_id,
  properties = NULL,
  in_trash = NULL,
  icon = NULL,
  cover = NULL
)
```

Arguments:

`page_id` Character (required). The ID for a Notion page.

`properties` Named list (JSON object). Page properties to update as key-value pairs.

`in_trash` Boolean. Set to TRUE to move the block to trash (delete). Set to FALSE to restore the block Defaults to FALSE.

`icon` Named list (JSON object). An icon for the page.

`cover` Named list (JSON object). A cover image for the page.

Details: Endpoint documentation

Examples

```
notion <- notion_client()

# ----- retrieve a page property

notion$pages$properties$retrieve(
  "23933ea0-c1e4-8104-897b-f5a09269e561",
  property_id = "q;L^"
)


# ----- update a page property

notion$pages$properties$update(
  "23933ea0-c1e4-8104-897b-f5a09269e561",
  list(
    `In stock` = list(
      checkbox = TRUE
    )
  )
)
```

`print.notion_response` *Print the result of a Notion API call*

Description

Print the result of a Notion API call

Usage

```
## S3 method for class 'notion_response'
print(x, ...)
```

Arguments

`x`	The result object.
`...`	Ignored.

Value

The JSON result.

`UsersEndpoint` *R6 Class for Users Endpoint*

Description

Handle all users operations in the Notion API

Note: Access this endpoint through the client instance, e.g., `notion$users`. Not to be instantiated directly.

Value

A list containing the parsed API response.

Methods

Public methods:

- `UsersEndpoint$new()`
- `UsersEndpoint$list()`
- `UsersEndpoint$retrieve()`
- `UsersEndpoint$me()`

Method `new()`**:** Initialise users endpoint. Not to be called directly, e.g., use `notion$users` instead.

Usage:

```
UsersEndpoint$new(client)
```

Arguments:

`client` Notion Client instance

Method `list()`**:** List all users

Usage:

```
UsersEndpoint$list(start_cursor = NULL, page_size = NULL)
```

Arguments:

`start_cursor` Character. For pagination. If provided, returns results starting from this cursor. If NULL, returns the first page of results.

`page_size` Integer. Number of items to return per page (1-100). Defaults to 100.

Details: Endpoint documentation

Method `retrieve()`**:** Retrieve a user

Usage:

```
UsersEndpoint$retrieve(user_id)
```

Arguments:

`user_id` Character (required). The ID of the user to retrieve.

Details: Endpoint documentation

Method `me()`**:** Retrieve the bot User associated with the API token

Usage:

```
UsersEndpoint$me()
```

Details: Endpoint documentation

Examples

```
notion <- notion_client()

# ----- list all users

notion$users$list()


# ----- retrieve a user

notion$users$retrieve(user_id = "fda12729-108d-4eb5-bbfb-a8f0886794d1")


# ----- retrieve the bot User associated with the API token

notion$users$me()
```

Index