Catalog-API

The Catalog-API is the main interface for apps and frontends to communicate with the Purple backend.

> ⚠ This documentation is intended for technical users (integrators and developers).

Overview

The Catalog-API is based on GraphQL. This allows for a type-safe and clearly defined schema while also allowing to query only the data you need.

> ⚠ This documentation assumes familiarity with GraphQL. Please read the official website's Introduction to GraphQL to learn about the concepts if you have never worked with GraphQL before.

To get started working with our API, you can use the online editor GraphiQL. It allows building and testing your queries without having to setup a local development environment.

We also offer an interactive visualization of all the types and their relations using the GraphQL Voyager tool.

You can view inline documentation of all the available types in both tools and your local development environment as the GraphQL schema offers direct support for documentation.

The endpoint that GraphQL clients have to use is: https://catalog.purplemanager.com/graphql

Queries

The Catalog-API offers both the *Query* and *Mutation* root types.

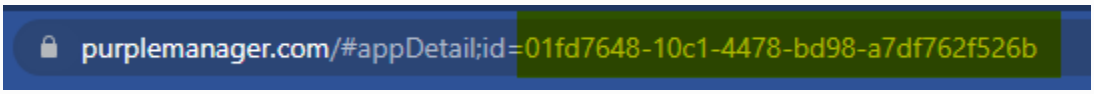
The most important part of the Catalog-API is the ability to query the contents and it's metadata of your app or website.

The entrypoint for this is the *catalog* field in the *Query* root type. The required arguments define the basic information that all other queries use.

Common request parameters

AppInfo

The *AppInfo* type holds the reference to your *appId* (the ID of your app or website in our system), the *appVersion* (the version of your app) and the *preview* flag (determines if you want preview or live contents).

> ⓘ You can find the appId in the Purple Manager when editing the app. It is currently only visible in the URL:
>
> 
>
>
> Here the appId is "01fd7648-10c1-4478-bd98-a7df762f526b"

DeviceInfo

The *DeviceInfo* type holds information about the requesting system. The *deviceId* is used to uniquely identify the system and allow access to previously anonymously purchased contents, e.g. in app stores.

> ⚠ If you are requesting data from a website you should generate a unique ID, e.g. an UUID, and store it in the local storage of the browser.

The *platform* defines the platform that is being used to request dats. Currently we differentiate between *ANDROID*, *IOS* and *WEB*. It is also used to filter the available contents, e.g. if you want certain contents to be only available on one platform or deliver different version to different platforms.

The *deviceModel* and *deviceOSs* fields are used to determine the device class, e.g. tables or phone for iOS, and for statistical and auditing purposes. Please provide accurate data about the model, e.g. the model name of the device or browser name, and the version of the device or browser.

The *smallestScreenWidthDp* field is only used for the *ANDROID* platform to determine the device class.

Authorization

The *Authorization* type is used to provide access tokens and subscription codes (coupon codes) to determine access and purchase status for contents.

Pagination

Most queries for data support paging using the GraphQL Cursor Connections Specification (arguments *first* and *after*). You can identify this by the *Connection* suffix at the fieldnames which support paging.

> ⚠ All APIs limit the maximum number of entries to **200 per query**. If you need to request more data you will need to request pages using the *after* argument.

Filtering and sorting

All connections support filtering and sorting the results.

Filtering

To filter the results you can use the *filter* parameter. The filters provide matchers for many fields of the Connections return type. You can however only use one field matcher at a time. To match based on multiple fields you have to combine multiple filters using the *AND* and *OR* fields.

Example: Filter for name or description containing the word "Sun"

```
{
  "filter": {
    "OR": [
      {
        "name": {
          "operation": "CONTAINS",
          "value": "Sun"
        }
      },
      {
        "description": {
          "operation": "CONTAINS",
          "value": "Sun"
        }
      }
    ]
  }
}
```

Sorting

The filtered results can be sorted using the *sort* parameter. Each comparator type allows defining the field that is used to compare for the sort and the direction (ascending or descending). Multiple comparators can also be chained, e.g. to sort by publication date and then name.

Example: Sort by publicationDate and name

```
{
  "sort": [
    {
      "publicationDate": {
        "direction": "DESC"
      }
    },
    {
      "name": {
        "direction": "ASC"
      }
    }
  ]
}
```

Contents (Issues, Posts and Bundles)

The *contentsConnection* can be used to query issues, posts and bundles of an app. The results can be filtered and sorted using the *filter* and *sort* arguments.

The contents have three major types that share the common *Content* interface.

Issue

Issues are magazine style contents. They usually have multiple pages and use our Storytelling Engine and can display animations, PDFs and other media.

Post

Posts are (news) article contents produced using our Purple Hub.

Bundle

Bundles are collections of Posts and are also produced using our Purple Hub.

Example queries

- All Posts
- All Issues
- All Bundles
- All Posts which have a category "news"
- All Posts which have a tag "news"
- All contents which have a custom property "cover" with value "true"

Publications

Publications are the main container for contents. Every content is assigned to one publication.

The *publicationsConnection* can be used to query the publications of an app. The results can be filtered and sorted using the *filter* and *sort* arguments.

Example queries

- All Publications
- All Newsfeeds (internally called CHANNEL)
- All Newsstands (internally called KIOSK)

Subscriptions

Apps can offer subscriptions to their users. Currently only native app store subscriptions are supported. Subscriptions are only unlocking contents for publications which there are assigned to.

The *subscriptionsConnection* can be used to query the subscriptions of an app. The results can be filtered and sorted using the *filter* and *sort* arguments.

Example queries

- All Subscriptions (ANDROID)
- All Subscriptions (IOS)

Publication Products

Publication products allows publishers, depending of the type of the product, to offer users to purchase all previously published content or use the same (consumable) app store product for multiple purchases of different contents. Publication products are only unlocking contents for publications which there are assigned to.

The *publicationProductsConnection* can be used to query the available product of an app. The results can be filtered and sorted using the *filter* and *sort* arguments.

Example queries

- All Publication products (ANDROID)
- All Publication products (IOS)

Taxonomies

The *taxonomiesConnection* can be used to query the taxonomies, e.g. tags and categories, of an app. The results can be filtered and sorted using the *filter* and *sort* arguments.

Example queries

- All Taxonomies

Collections

Collections are curated lists of posts/articles.

The *collectionsConnection* can be used to query the collections of an app. The results can be filtered and sorted using the *filter* and *sort* arguments.

Example queries

- All Collections

Menus

Menus are currently only used for websites. They allow dynamically configuring the menu of a Purple-based website.

The *menusConnection* can be used to query the menus of an app. The results can be filtered using the *filter* argument.

Example queries

- All Menus

Search

The API allows to perform full-text searches across all contents of an app.

The *contentSearchConnection* can be used to perform full-text searches.

Example queries°

- Search for contents that contain the word "Lorem"
- Search for contents that contain the words "Lorem" and "article" (using the option findAllWords)
- Search for contents that contain the words "Lorem" or "issue"