GakuNinRDM ストレージ
ダウンローダー・アップローダー
使用手引書

0 版

2024.07.12

データ管理システム開発ユニット）林　寛生

改訂履歴

版数	改訂日	改訂者	対象	改訂内容
0	2024.07.12	林寛生	全体	新規作成（Poetry 版より変更）

目次

1　はじめに

このソフトウェアは、GakuNinRDM で使用しているアドオンストレージに格納されたデータに関して、ローカルの環境にダウンロードもしくはローカルの環境からアップロードするための Python スクリプトです。

2　前提条件

Python（Version 3.10 以上）および Python のサードパーティ製モジュールである requests と pydantic-settings が利用できる環境であることが前提です。

3　インストール

① 本ソフトウェアの取得

wegt コマンドや Web ブラウザ等で以下の URL よりダウンロードしてください。

https://dmsutil.riken.jp/tool/grdm_storsync-0.1.0.tar.gz

② 本ソフトウェアのインストール

ダウンロードした本ソフトウェのパッケージを適当なフォルダに展開してください。Linux では以下のようにします。

例）

```
$ tar xvfz {ダウンロードしたフォルダ}/grdm_storsync-0.1.0.tar.gz
```

③ Python のアップデート

本ソフトウェアでは、**バージョン 3.10 以上**の Python が必要です。以下のように作業環境のバージョンを確認してください。

例）

```
$ python3 --version
```

もし、Python のバージョンが低い場合はアップデートしてください。（※環境によっては管理者権限が必要かもしれません）

④ 依存ライブラリのインストール

本ソフトウェアでは、Python の標準ライブラリの他に、サードパーティ製のモジュールである requests および pydantic-settings が必要です。Linux では以下のようにすると、これらのモジュールが作業環境で利用できる場合は、その名称とバージ

ョンが表示されます。

例）

```
$ pip list | grep -e requests -e pydantic-settings
```

もし、これらのモジュールが無い場合は、例えば以下のようにインストールしてください。（※環境によっては管理者権限が必要かもしれません）

例）

```
$ pip install requests
$ pip install pydantic-settings
```

4 Config ファイルの作成

ユーザごとの設定を記述した Config ファイルを作成します。vi 等のエディタで、以下のような .env ファイルを作成してください。

【.env ファイルの例】

```
GRDM_ACCESS_TOKEN="ABCDEFGHIJKLM0123456789NOPQRSTUVWXYZ"
GRDM_OSFAPI_BASEURL=" https://api.rdm.nii.ac.jp/v2/"
GRDM_WBAPI_BASEURL=" https://files.rdm.nii.ac.jp/"
GRDM_PROJECT_ID="xyz01"
GRDM_STORAGE_PROVIDER="s3compat"
GRDM_MOUNT_POINT=" /home/user/mnt_dir"
```

- ファイル名は .env とし、カレントフォルダに置いてください。
- 以下の６つの環境変数は必須です。

環境変数名	説明
GRDM_ACCESS_TOKEN	ユーザが GakuNinRDM で設定したパーソナルアクセストークン
GRDM_OSFAPI_BASEURL	GakuNinRDM の OSF API のベース URL
GRDM_WBAPI_BASEURL	GakuNinRDM の WaterButler API のベース URL
GRDM_PROJECT_ID	GakuNinRDM 上のプロジェクト ID（5 桁の英数字）
GRDM_STORAGE_PROVIDER	ダウンロード/アップロード対象のデータが格納される GakNin RDM のストレージ名
GRDM_MOUNT_POINT	マウントポイント = ダウンロード/アプロー

	ド対象のデータが格納されるローカルのフォルダのパス

- 上記以外のパラメタについては、付録 A. を参照してください。
- GRDM_STORAGE_PROVIDER の値が不明の場合は、付録 B. を参照してください。
- 何らかの理由でカレントフォルダに .env が作成できない場合は、任意のファイルに作成し、そのパスをコマンド実行時の引数で指定してください。(※カレントフォルダの .env より優先されます)

5 ダウンローダーの使用方法

GakuNin RDM のストレージからローカルの環境へデータをダウンロードするには、以下のように Python スクリプト grdm_stordl.py を実行します。

例)

```
$ python3 grdm_stordl.py {Config ファイルパス} -mp {マウントポイント}
```

- grdm_stordl.py が格納されているフォルダ以外で実行する場合は、grdm_stordl.py までのパスで指定してください。
- Config ファイルパスは省略できます。その場合は、カレントフォルダの .env が読まれます。(無い場合は grdm_stordl.py が格納されるフォルダの .env が読まれます)
- Config ファイルパスを指定した場合は、そのファイル中での設定が優先されます。
- -mp オプションでマウントポイントを指定した場合は、Config ファイル(.env 含む)でのマウントポイントの設定より優先されます。
- マウントポイントのフォルダは事前に作成してください。また、フォルダ内が空でない場合はダウンロードしません。

実行後、以下のようにダウンロードの概要が標準出力に表示されます。確認の上、問題がなければ **yes** を入力してダウンロード処理を進めてください。

```
----------------------------------------------------------------------------------------------------
Data file(s) will be downloaded from GakuNin RDM storage.
  GakuNin RDM project ID          : xyz01
  From (Remote storage folder)    : /s3compat/
  To (Local folder)               : /home/user/mnt_dir
----------------------------------------------------------------------------------------------------
Do you really want to proceed? [yes/no]
```

6 アップローダーの使用方法

ローカルの環境から GakuNin RDM のストレージへデータをアップロードするには、以下のように Python スクリプト grdm_storul.py を実行します。

例）

$ python3 grdm_storul.py {Config ファイルパス} -mp {マウントポイント}

- grdm_storul.py が格納されているフォルダ以外で実行する場合は、grdm_storul.py までのパスで指定してください。
- Config ファイルパスは省略できます。その場合は、カレントフォルダの .env が読まれます。（無い場合は grdm_storul.py が格納されるフォルダの .env が読まれます）
- Config ファイルパスを指定した場合は、そのファイル中での設定が優先されます。
- -mp オプションでマウントポイントを指定した場合は、Config ファイル（.env 含む）でのマウントポイントの設定より優先されます。

実行後、以下のようにアップロードの概要が標準出力に表示されます。確認の上、問題がなければ **yes** を入力してアップロード処理を進めてください。

```
----------------------------------------------------------------------------------------------------
Data file(s) will be uploaded to GakuNin RDM storage.
   GakuNin RDM project ID       : xyz01
   From (Local folder)          : /home/user/mnt_dir
   To (Remote storage folder)   : /s3compat/20241231_235959
   * Note: Remote folder will be created before uploading data file(s).
[GRDM_UPLOAD_REPLACE_FLAG=FALSE]
   * Note: Local data file(s) will be kept even after uploading.
[GRDM_LOCAL_DELETE_FLAG=FALSE]
----------------------------------------------------------------------------------------------------
Do you really want to proceed? [yes/no]
```

- フラグについては 付録 A. を参照してください（いずれもデフォルト値は FALSE）。
- デフォルトでは、アップロード先のフォルダ（ターゲットフォルダ）を上書きせず、ターゲットフォルダ以下に新規フォルダ（※フォルダ名はアップロード時の日付と時刻から；例えば"20241231_235959"）を作成し、そこにデータをアップロードします。上書きしたい場合は、GRDM_UPLOAD_REPLACE_FLAG の値を TRUE に設定してください。ただし、内容が同じでも上書きしてしまう（更新時刻等が変わる）ので注意が必要です。
- ローカルの環境にあるデータは GakuNinRDM のストレージへのアップロード後

も削除されることはありません。もし、アップロード後に自動的に削除したい場合は、GRDM_LOCAL_DELETE_FLAG の値を TRUE に設定してください。

付録

A. Config ファイルと設定パラメタ

Configファイルは、ユーザごとの設定を記述するためのファイルです。基本的には、カレントフォルダの .env ファイルに環境変数として設定します。設定の優先順位は以下の通りです(1 > 2 > 3)。

1. コマンド実行時の引数で指定されたファイル中の設定
2. .env ファイル中の設定
3. OS の環境変数設定

- .env ファイルは、カレントフォルダに無ければ、実行する Python スクリプトが置いてあるフォルダ（※）の .env を探します。（※本ソフトウェアのパッケージを展開したフォルダ）
- 本ソフトウェアでは、JSON 形式のファイルでも設定できます。その場合はファイルの拡張子を「.json」とし、コマンド実行時の引数でファイルパスを指定してください。また、キーは大文字ではなく小文字で記述してください。

本ソフトウェアで設定できるパラメタ以下の通りです。なお、一部のパラメタは設定を共有する別ソフトウェア(GakuNin RDM ストレージマウンター)でのみ利用され、本ソフトウェアでは利用されません。

パラメタ（環境変数）	型	説明
GRDM_ACCESS_TOKEN	str	ユーザが GakuNinRDM で設定したパーソナルアクセストークン
GRDM_OSFAPI_BASEURL	str	GakuNin RDM の OSF API のベース URL
GRDM_WBAPI_BASEURL	str	GakuNinRDM の WaterButler API のベース URL
GRDM_PROJECT_ID	str	GakuNinRDM 上のプロジェクト ID（5 桁の英数字）
GRDM_STORAGE_PROVIDER	str	ダウンロード/アップロード対象のデータが格納される GakNin RDM のストレージ名
GRDM_MOUNT_POINT	str	マウントポイント = ダウンロード/アップロード対象のデータが格納されるローカルのフォルダのパス
GRDM_TARGET_FOLDER	str	ダウンロード/アップロード対象のデータが格納される GakNin RDM のストレージ内のフォルダのパス［default=″/″（ストレージのトップ)］
GRDM_UPLOAD_CONCURRENT	int	データアップロード時の同時処理数［default=10］
GRDM_UPLOAD_REPLACE_FLAG	bool	データアップロード時に GakuNin RDM

		のストレージにある元データを上書きするかどうかのフラグ［default=FALSE (上書きしない)］
GRDM_LOCAL_DELETE_FLAG	bool	データアップロード後にローカルのデータを削除するかどうかのフラグ［default=FALSE (削除しない)］
GRDM_OUT_LOG	str	（※本ソフトウェアでは利用しない)コマンドのログの出力ファイルのパス

- GRDM_TARGET_FOLDER はストレージのトップ（〝/〝）を含む絶対パスで指定してください。
- GRDM_UPLOAD_REPLACE_FLAG が FALSE の場合は、アップロード先のフォルダ（ターゲットフォルダ）以下に新規フォルダ（※フォルダ名はアップロード時の日付と時刻から；例えば〝20241231_235959〝）を作成し、そこにデータをアップロードします。
- default 値が記載されていないパラメタが設定されない場合は、本ソフトウェアでは None として扱われ、処理がエラーになります。（GRDM_OUT_LOG は設定されなくても処理はエラーになりません）

B. ストレージプロバイダ名

GRDM_STORAGE_PROVIDER で指定する値（ストレージプロバイダ名）が不明な場合は、本ソフトウェアのパッケージに同梱してある Python スクリプト grdm_storls.py を以下のように実行してください。その実行結果として、GRDM_PROJECT_ID で指定する GakuNin RDM のプロジェクトにマウントされているストレージ名のリストが表示されますので、該当するものを選んでダウンローダー/アプローダーの GRDM_STORAGE_PROVIDER に指定してください。

例）

```
$ python3 grdm_storls.py {Config ファイルパス} -p {プロジェクト ID}
```

- grdm_storls.py が格納されているフォルダ以外で実行する場合は、grdm_storls.py までのパスで指定してください。
- ここで設定が必要な Config パラメタは、GRDM_ACCESS_TOKEN、GRDM_PROJECT_ID、および GRDM_OSFAPI_BASEURL の 3 つだけです。
- Config ファイルパスは省略できます。その場合は、カレントフォルダの .env が読まれます。
- Config ファイルパスを指定した場合は、そのファイル中での設定が優先されます。
- -p オプションでプロジェクト ID を指定した場合は、Config ファイル（.env 含む）でのプロジェクト ID の設定より優先されます。