Claudius Iacob

**pdFetch 1.0.0 Documentation**

**pdFetch** is a Node.js application designed to export Knowledge Base (KB) articles from a ServiceNow instance into PDF files. It provides various modes of operation to handle different use cases, including listing articles, identifying changes, downloading listed articles as PDFs and keeping them up to date. It also provides special storage modes such as archiving or merging the individual PDF files.

**Scenarios**

Typical scenarios where **pdFetch** can be employed include:

1. **Exporting KB Articles to PDF**: For sharing knowledge base articles in a portable document format that can be easily viewed, printed, or archived.
2. **Change Detection**: Identifying changes in KB articles over time, useful for maintaining up-to-date off-site documentation.
3. **Integration with Text Processing Applications**: PDF files can be used as an interchange format for various unattended text processing applications that accept PDF input.

**Audience**

**pdFetch** can be used:

1. **Manually by a Human User**: A user can execute the application from the command line, providing the necessary arguments to perform the desired operation.
2. **Mechanically by an Unattended System**: Once a configuration file is set up, the application can be run automatically, such as through a cronjob, to perform regular exports when articles are added removed or updated in a remote ServiceNow instance.

**How to Install**

**Prerequisites**

1. **Node.js**: Ensure that Node.js is installed on your system. You can download the latest LTS version from [nodejs.org](https://nodejs.org/).

**Installation Steps**

1. **Unzip the Application**: Unzip the provided **pdFetch** package to your desired directory.
2. **Install Dependencies**: Navigate to the application directory and run:

> npm install

1. **Create a shortcut**: Create a batch file or shell script to run the application easily. Name it pdFetch.bat or pdFetch.sh, as applicable. Add this script file to your system's PATH. The script should contain:

node /path/to/application/directory/index.js

You can then simply type in your console, e.g.:

> pdFetch --h

**How to Use It**

**Command-Line Arguments**

Here are the command-line arguments that **pdFetch** accepts:

* --output\_dir=path/to/dir: The working directory of the program. All downloads and content generations (except the configuration file) will be done in this directory. It must be an absolute path to an existing folder. (Mandatory)
* --sn\_instance\_name=instance\_name: The name of the ServiceNow instance to connect to, e.g., acme in acme.service-now.com. (Mandatory)
* --sn\_user\_name=user\_name: The username for connecting to the ServiceNow instance using a local account, e.g., john.doe. (Mandatory)
* --sn\_pass=password: The password for the user. (Mandatory)
* --help: Displays information about the input parameters of the program and stops execution, even if other parameters have been provided.
* --init\_config: Initializes an empty configuration file in the user's home directory and stops execution, even if other parameters have been provided. The configuration file name is pdFetch.config.
* --config\_profile=profile\_name: Loads default data from a configuration profile, assuming it has been defined and there is a configuration file.
* --sn\_query=query: Additional filter to add to the default filter used for obtaining articles.
* --storage\_mode=mode: Determines how the downloaded/generated content is stored in the working directory. Possible values are:
  + files: Creates PDF files in a PDFs subfolder (default).
  + archived\_files: Creates an archive of the downloaded/generated PDF files.
  + single\_file: Creates a combined PDF file of all the downloaded/generated PDF files.
* --operation\_mode=mode: The operations the application performs. Possible values are:
  + list: Only downloads a list of available articles without downloading the files.
  + list\_changes: Creates a list of changes based on an older list of available articles.
  + list\_files: Downloads/generates a PDF file for each listed entry.
  + list\_changes\_files: Updates the PDFs subfolder to reflect the current state of articles.

**Examples**

1. **Listing Articles**:

> node index.js --output\_dir=/path/to/dir --sn\_instance\_name=acme --sn\_user\_name=john.doe --sn\_pass=letmein1234 --operation\_mode=list

1. **Listing Articles with Changes**:

> node index.js --output\_dir=/path/to/dir --sn\_instance\_name=acme --sn\_user\_name=john.doe --sn\_pass=letmein1234 --operation\_mode=list\_changes

1. **Listing Articles and Downloading PDFs**:

> node index.js --output\_dir=/path/to/dir --sn\_instance\_name=acme --sn\_user\_name=john.doe --sn\_pass=letmein1234 --operation\_mode=list\_files

**Default Values and Profiles**

* **Default Profile**: The application first loads settings from a default profile if it exists.
* **Specified Profile**: If a profile is specified using --config\_profile, the values it defines override those in the default profile settings.
* **Command-Line Arguments**: Finally, any given command-line arguments override both profile settings.

**Validating Arguments**

If mandatory arguments are missing, the application will exit early with an error message.

If an argument given on the command line is not understood, the application will exit early as well.

**Caveats**

**Session Lock File**

A operation\_in\_progress.lock file is created in the target directory when the application starts and is removed when it exits. The lock file ensures that the application does not run multiple instances simultaneously on the same directory, which could lead to data corruption or unexpected behavior.

* **Caveat**: If the application is stopped abruptly, the lock file will remain, and the application will refuse to work on subsequent runs.
* **Solution**: Manually delete the lock file. It is recommended to delete everything in the target directory in such cases.

**Bash Examples**

1. **Running the Application**:

pdFetch --output\_dir=/path/to/dir --sn\_instance\_name=acme \

--sn\_user\_name=john.doe --sn\_pass=letmein1234 --operation\_mode=list\_files

1. **Handling Exit Status**:

if pdFetch --output\_dir=/path/to/dir --sn\_instance\_name=acme \

--sn\_user\_name=john.doe --sn\_pass=letmein1234 --operation\_mode=list\_files; then

echo "pdFetch ran successfully"

else

echo "pdFetch encountered an error"

fi

1. **Running from a Cronjob**: Add the following line to your crontab (edit with crontab -e and enter everything on just one line):

0 2 \* \* \* pdFetch --output\_dir=/path/to/dir --sn\_instance\_name=acme --sn\_user\_name=john.doe --sn\_pass=letmein1234 --operation\_mode=list\_changes\_files --storage\_mode=files

This will run **pdFetch** every day at 2 AM, listing changes and downloading files as needed.

**Built-in documentation**

Refer to the --help argument for an abridged version of this file.