Datafile OS – Getting Started

Table of Contents

Datafile OS – Getting Started

1. Operating System

This installation assumes that you are running on a clean, late copy of Debian, Ubuntu LTS or Mint LTS. If you are running a different distribution you will need to edit the script to suit your distro.

The script to install is called dfinstall.sh.

For the version for the Debian family, download the appropriate .tgz from the web site.

There is now a separate script for installing on Red Hat 8.7 - use the download from the web site : http://www.projectcomputers.co.uk/datafileos.html and untar/zip. It has not been tested on other versions of Red Hat which may require further modification for the LAPP stack.

It is assumed that neither Apache or Postgres are pre installed. If this is not the case then you should inspect the installation script carefully to ensure that your existing systems will not be compromised.

If you have a pre-existing Postgresql installation and have modified the postgres password you will need to modify the script accordingly.

If you need to run on a different database, web server etc. please contact us - we will be happy to discuss your requirements.

Please be aware that the database user "pcl" password defaults to "parrot". This is not considered a secure password and should be modified after installation. You will also need to update the "secfile.json" and “secfile” files under the /home/datafile/pcldat directory – see **Section 4** below.

2. Creating a User, Granting Rights

The installation script assumes that you have followed these short steps to create the Linux user **datafile** and made the user a member of the sudoers group.

```
sudo adduser datafile
 (with password of parrot)

sudo usermod -aG sudo datafile

su datafile

cd ~
```

3. Download and Installation

The installation script assumes that you have followed these short steps above to create the Linux user **datafile** and made the user a member of the sudoers group.

Datafile OS – Getting Started

It is assumed that you are now in the newly created datafile directory.

Download the dfinstall.tgz file to the home/datafile directory and un-tar it in situ.

```
tar -xzvf dfinstall.tgz
```

or, in the case of Red Hat 8.7, rh-dfinstall.tgz

Then run the installation script :

```
sh dfinstall.sh
```

or for Red Hat :

```
sh rh-dfinstall.sh
```

If at any stage in running the script there is an error, such as passwords failing to match, you should abort the script and start it again. Any packages already installed will be ignored. Failure to install certain packages will cause problems later in the script.

Note that you will need to press the “y” button to confirm acceptance of the package downloads/installations as the question occurs throughout the script.

Also **note** that you may see “Completed !” as various package installations are completed. This does not mean the script has run to end – wait for the message below before assuming the script has finished :

```
You should now consult the documentation …
```

Please also pay attention to any notes in the script.

When you see a section beginning “Created symlink … ” enter the pcl role password.

```
Created symlink ….
Enter password for new role :
```

(this is the PostgreSQL) role “pcl” in the PostgreSQL database) – if you are still using the standard “parrot” password, enter this here, or following the instructions in Section 1 to change the password in the script and configuration files. You WILL need to remember this to enter into the <secfile.json>.

In psql you will be presented with the “postgresql=#” prompt. Ensure that you enter the following :

```
postgresql=# grant all privileges on database datafile to pcl;
```

You should see :

```
postgresql=# GRANT
```

when this has been completed.

Now enter the following to exit psql and return to the remainder of the script :

```
postgresql=# \q
```

and hit Return.

Ignore any messages similar to these :

```
To activate you need to run:
systemctl reload apache2
```

You should then see the final note from the script :

```
You should now consult the documentation …
```

“Documentation” refers to this document.

Before continuing, either reboot (if you are using Red Hat, ensure you reboot) or log out and then login as the Linux user datafile.

Go to a terminal again and create a new user for the Datafile OS system :

```
dfadduser <your chosen id>   and then password of your choosing
```

e.g.

```
dfadduser ops
```

You should now be in a position to run up the Datafile OS system – but at the moment you won’t have any data to look at.

Red Hat

In order to test the system on **Red Hat 8.7 only** you must turn selinux off by using the command :

```
setenforce 0
```

Note that setenforce 0 is not persistent so if you wish to permanently disable selinux you must do so by editing its configuration file. Alternatively,you may wish to implement an SELinux policy which permits your users and supervisors/operators the access they require. This is outside the scope of this document. Datafile will be running on an intranet server so access to it will also be controlled by your firewall and login policies, so there may be no requirement for the additional level of security and complexity offered by SELinux.

4. Configuration and Indexing Data

You will now need to configure your system and copy in your reports in preparation for indexing, which makes them available to users via the browser-based Datafile OS.

There is more information on the nature of the data/reports in Section 2 of the document Datafile **OS – Functionality, Configuration and Setup** also available from the web site.

Here we assume that you are using EBCIDIC PRCTL data transferred from an IBM host machine, that the line lengths of the reports is 132 columns and that the format has a 4-byte leader. The files MUST be in original EBCDIC format.   (Files such as .TTO are normally used to configure this – see the instructions available on your host system for more detail.)

Note re json file format

json file format is simple but any errors simply cause the process to fail, usually doing nothing in a rather grumpy way. The JSONLint validation tool is extremely helpful in ironing out any issues - https://jsonlint.com/

Security files for database access

Datafile requires 2 security files, (both in the */home/datafile/pcldat directory),* to be configured using a text editor :

- secfile (for historical compatibility)
- secfile.json

Both contain the same information :

```
{
    "database":
    {
        "host": "localhost",
        "dbName": "datafile",
        "port": "5432",
        "userName": "pcl",
        "password": "parrot"
    }
}
```

Again, these files will need to be modified if you are using your own password.

System configuration

The system is configured using 2 files (both in the */home/datafile/pcldat directory)* :

dfsites.json

Your dfsites.json file must contain an entry for your sysid. The one in the download is pointing at an id called **fin** and there is a <sysid>.json called **fin.json**.

This has a line for the data :

dataDir   Beneath the home path of the special Linux user **datafile** a directory for raw and indexed data

e.g.

```
{
        "dataDir": "/home/datafile/fin",
```

Copy your raw report data to this directory. The file names should be in lower case. All reports of identical basic format (e.g. header, report id, line length) should be contained a single file :

- download.136

All other reports of differing formats should be downloaded as single files, one for each format e.g.

- statements.136
- advices.136

Once these files are present in the correct directory you can index them, from the command line as Linux user **datafile**. Note that indexing from the command line gives the additional security that it requires the user to utilise the Linux login

```
indexing <sysid> <YYYYMMDD date>
e.g. indexing bnk 20220924
```

You should see the report identifiers, start and end offsets and number of pages found zipping by. Once the process is complete you may log into via the browser, using the Datafile OS id you created earlier at the end of Section 3 (e.g. **ops**).

Details on how to check that the indexing process has processed your reports as you would wish, identifying the correct reports by identifier are contained in Section 5 of Datafile OS – **Functionality, Configuration and Setup** document.

5. Existing Datafile Sites

If you have an earlier version of Datafile you will need to migrate by :

- copying the data (processed reports)
- convert the indices

in order to have access to your archived reports.

Copying report data (previously indexed)

The data downloaded from the host system remains unchanged, as it has always in Datafile. Previously indexed files are renamed to NNNNN.dat by older versions and the format remains the same.

Data Directory

Your old system will have a path to the data (applies to both indexed and un-indexed raw data). Copy the *.DAT files to the Data Directory path specified in the <sysid>.json file (see **Section 4.**

Index Directory

Your old index files must be converted, using the **idxconv** program which loads old index files into the PostgreSQL database.

Recommended Migration

Many existing clients have been using the Datafile system for many years. You may choose to prune the data you wish to retain at this point but the principle remains the same :

- load the indices using the utility **idxconv**
- copy the matching report files
- check recall of indices and display of reports

Start with 1 month at a time – to test the process

Copy one month's worth of indices from the Index Directory path on your old system to the Index Directory on your new system (as specified in <sysid>.json).

Run idxconv on that month's worth of indices, using the 2nd parameter :

idxconv <sysid> <month directory for indices>

e.g.

idxconv fin home/datafile/kapi/202302

In order to test the indices for a previously archived month, you will need the .DAT files which correspond to that month's indices. You can identify these any of these 3 ways :

- if your .DAT files have the original dates (rather than any dates which related to copying data around) then you can simply identify these by date from the Data Directory specified in your old system (Workstation / …..)
- or you can recall an entire month's worth of reports with your Windows version, noting the first and last file numbers
- or you can use the new browser recall system to do the same thing

and then copy the relevant .DAT files across to the new system. Recalls should then display the reports as expected.

Once you have tested successfully, delete the processed month directories from the Index Directory i.e. if you have processed September 2022 then remove the 202209 directory or you may get duplicate key errors which will include this :

“duplicate key value violates UNIQUE CONSTRAINT” indicates that the indices you are attempting to convert have already been loaded into the database.

Once you are happy with a series of 1 month tests …

you can use the same process for a longer period. We suggest that you clear the Index Directory on the new system and then, for example, copy 3 month’s of index directories.

You can use the index conversion to process all index directories :

```
idxconv fin kapi
```

Deal with the related .DAT files in the same way and then you can progress.

The speed at which you convert and check data is entirely up to the customer. This is merely a suggestion to ensure that you have all files converted and in the right places.

INTERIM STORAGE

Please note that the method used by historical versions of Datafile - having a master copy of .DAT files in the Data Directory and another copy for immediate access in the INTERIM directory is no longer relevant. This structure was originally designed to speed up access when clients were using optical disks, and in particular optical jukeboxes.

BACKUPS

All data is now stored solely in the Data Directory and backing up this data and the database are the responsibility of the client.

DATABASE BACKUPS

Old and new indices are stored in the PostgreSQL database - **datafile**. Backups of this can easily be made using PostgreSQL pg_dump to produce a text file which can be backed up and could, if required, be reloaded to the database with either psql or pg_restore.