FMK | FileMaker Konferenz

Shell-Scripting am FMS

Philipp A. Puls - 72solutions GmbH

Mag. Philipp A. Puls

- geschäftsführender Gesellschafter und Mitbegründer der 72solutions GmbH
- Hintergrund in theoretischer Physik von der Universität Wien und der University of Edinburgh.
- 72solutions GmbH ist Platinum-Partner von Claris FileMaker und ein „SBA“-Partner mit ihren Lösungen «base72 Toolbox» und «onexio».
- FileMaker-Karriere 1992 mit Version 2.1 begonnen und seit 2015 durchgehend zertifiziert
- Seit 2025 Veranstalter der “Vienna Calling” Unconference in Wien.

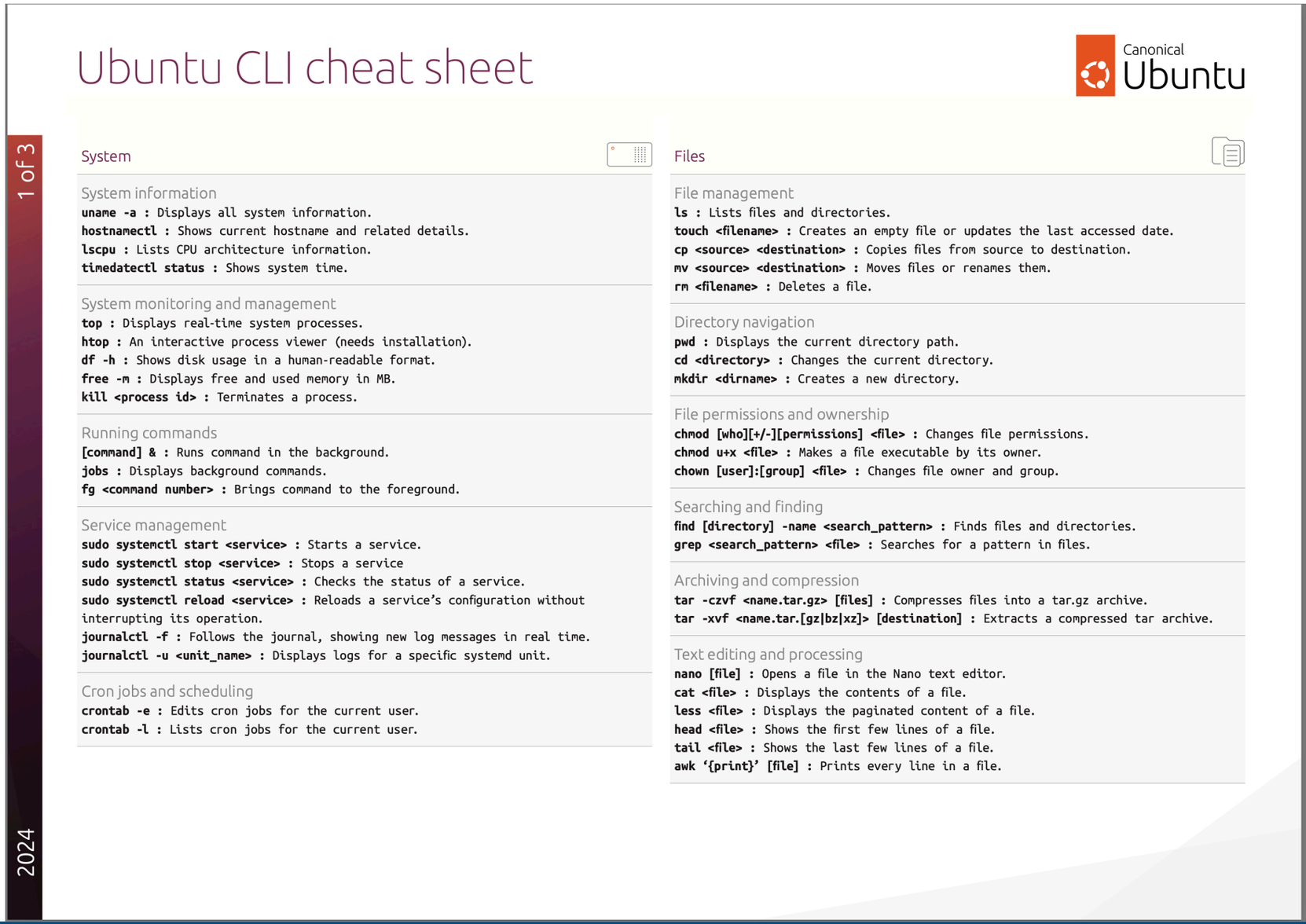
Zertifiziert für:

“Muschel was…?”

Command Line Interface

- Das offizielle Ubuntu CLI cheat sheet

(https://assets.ubuntu.com/v1/3bd0daaf-Ubuntu Server CLI cheat sheet 2024 v6.pdf)



Ubuntu CLI cheat sheet

Canonical Ubuntu

1 of 3

2024

System

System information

uname -a : Displays all system information.
hostnamectl : Shows current hostname and related details.
lscpu : Lists CPU architecture information.
timedatectl status : Shows system time.

System monitoring and management

top : Displays real-time system processes.
htop : An interactive process viewer (needs installation).
df -h : Shows disk usage in a human-readable format.
free -m : Displays free and used memory in MB.
kill <process id> : Terminates a process.

Running commands

[command] & : Runs command in the background.
jobs : Displays background commands.
fg <command number> : Brings command to the foreground.

Service management

sudo systemctl start <service> : Starts a service.
sudo systemctl stop <service> : Stops a service
sudo systemctl status <service> : Checks the status of a service.
sudo systemctl reload <service> : Reloads a service's configuration without interrupting its operation.
journalctl -f : Follows the journal, showing new log messages in real time.
journalctl -u <unit_name> : Displays logs for a specific systemd unit.

Cron jobs and scheduling

crontab -e : Edits cron jobs for the current user.
crontab -l : Lists cron jobs for the current user.

Files

File management

ls : Lists files and directories.
touch <filename> : Creates an empty file or updates the last accessed date.
cp <source> <destination> : Copies files from source to destination.
mv <source> <destination> : Moves files or renames them.
rm <filename> : Deletes a file.

Directory navigation

pwd : Displays the current directory path.
cd <directory> : Changes the current directory.
mkdir <dirname> : Creates a new directory.

File permissions and ownership

chmod [who][+/-][permissions] <file> : Changes file permissions.
chmod u+x <file> : Makes a file executable by its owner.
chown [user]:[group] <file> : Changes file owner and group.

Searching and finding

find [directory] -name <search_pattern> : Finds files and directories.
grep <search_pattern> <file> : Searches for a pattern in files.

Archiving and compression

tar -czvf <name.tar.gz> [files] : Compresses files into a tar.gz archive.
tar -xvf <name.tar.[gz|bz|xz]> [destination] : Extracts a compressed tar archive.

Text editing and processing

nano [file] : Opens a file in the Nano text editor.
cat <file> : Displays the contents of a file.
less <file> : Displays the paginated content of a file.
head <file> : Shows the first few lines of a file.
tail <file> : Shows the last few lines of a file.
awk '{print}' [file] : Prints every line in a file.

Was kann das CLI

- Operationen des Filesystems (mv, cp, scp, rsync, rm, chmod, chown, …)
- Operationen des Betriebsystems (usermng, u-/mount, crontab, systemctl, …)
- fmsadmin

(https://support.claris.com/s/article/FileMaker-Server-Command-Line-Reference-15036930659 18?language=en_US#cli)

und vieles mehr aber auch:

- Installation und Ausführung beliebiger Programme, ob über apt, snap oder auch per Download aus einer beliebigen Webadresse!

Was ist Shell-Scripting?

- CLI Befehle in Gruppen in einem Text-File hinterlegen
- Wird direkt am Server ausgeführt
- Automatisiert repetitive Prozesse

Muss ich da aufpassen?

SICHERHEIT!

- **User** fmserver **Gruppe** fmsadmin - was der kann, kann im schlimmsten Fall jeder Filemaker User!

Aber wie geht das denn?

Man kann ein File.sh schreiben und es ausführen…

- wenn FM-Benutzer Zugriff auf Scripts haben (per PSOS einen Conainerinhalt ablegen oder mit einem «Write to Data File» direkt schreiben)
- wenn FM-Benutzer einen Text editieren können, der per PSOS evaluiert wird

… MBS macht das leichter, es geht aber auch ohne (z.b. indem man den Inhalt eines FMS-SystemScript ersetzt — wenn es schon eines gibt).

Was wir damit machen:

Lernen am Beispiel

Plugin Installation

- /opt/FileMaker/FileMaker Server/Data/Scripts$

```
#!/bin/bash

fmsadmin stop fmse -y

rm "/opt/FileMaker/FileMaker Server/Database Server/Extensions/MBS*”

cd "/opt/FileMaker/FileMaker Server/Data/Documents/“

wget https://www.monkeybreadsoftware.com/filemaker/files/gzip/MBS.fmx.gz

gzip -d “MBS.fmx.gz"

mv "MBS.fmx" "/opt/FileMaker/FileMaker Server/Database Server/Extensions/“

chown -R fmserver:fmsadmin "/opt/FileMaker/FileMaker Server/Database Server/Extensions/“

chmod -R 775 "/opt/FileMaker/FileMaker Server/Database Server/Extensions/*”

fmsadmin restart fmse -y

# EOF
```

AuditLog File-Rotation

- /opt/FileMaker/FileMaker Server/Data/Scripts$

Was wir tun wollen:

1. Vom gehosteten Log-File einen Clone erzeugen
2. Das gehostete Log-File schließen
3. Das letztgültige Log-File in ein Sub-Verzeichnis verschieben und umbenennen
4. Den Clone umbenennen und in das Live-Verzeichnis verschieben
5. Alle Dateien wieder hosten

Mein Proof-Of-Concept

```
#!/bin/bash
TODAY=$(date +"%Y-%m-%d")
# ====== Define paths ======
ARCH_DIR="/opt/FileMaker/FileMaker Server/Data/Databases/AuditLogArchiv/"
FILE="/opt/FileMaker/FileMaker Server/Data/Databases/ExternesLog.fmp12"
BACKUP_FM="filelinux:/opt/FileMaker/FileMaker Server/Data/Backups/"
BACKUP_DIR="/opt/FileMaker/FileMaker Server/Data/Backups/"
# ====== Ensure the backup directory exists ======
mkdir -p "$ARCH_DIR"
# ====== Get the password from the first argument ======
PASSWORD="$1"
# ====== Run the backup command ======
fmsadmin backup "ExternesLog.fmp12" -d "$BACKUP_FM" -k 0 -n -u fmsadmin -p "$PASSWORD"
fmsadmin close -u fmsadmin -p "$PASSWORD" ExternesLog.fmp12 -y
mv /opt/FileMaker/FileMaker\ Server/Data/Databases/ExternesLog.fmp12 /opt/FileMaker/FileMaker\ Server/
  Data/Databases/AuditLogArchiv/ExternesLog_bis_$TODAY.fmp12
mv /opt/FileMaker/FileMaker\ Server/Data/Backups/Cloned_by_FMS/Databases/ExternesLog\ Clone.fmp12 /opt/
  FileMaker/FileMaker\ Server/Data/Databases/ExternesLog.fmp12
sleep 10
fmsadmin open -u fmsadmin -p “$PASSWORD”
# EOF
```

Das Endprodukt

Wenn man die Profis rannläßt …

```
#!/bin/bash

# set -x

# Setting up variables

TODAY=$(date -I)

# Path definitions

FILEMAKER_BASE_DATA_DIR="/opt/FileMaker/FileMaker Server/Data"

FILEMAKER_BASE_DATA_DATABASES_DIR="${FILEMAKER_BASE_DATA_DIR}/Databases"

FILEMAKER_BACKUP_DIR="${FILEMAKER_BASE_DATA_DIR}/Backups"

FILEMAKER_BACKUP_CLONE_DIR="${FILEMAKER_BACKUP_DIR}/Cloned_by_FMS/Databases"

AUDITLOGARCHIVE_DIR="${FILEMAKER_BASE_DATA_DATABASES_DIR}/AuditLogArchiv"

EXTERNAL_LOG_FILE="ExternesLog.fmp12"

FILEMAKER_EVENT_LOGFILE="/opt/FileMaker/FileMaker Server/Logs/Event.log"

SCRIPT_LOGFILE_PARTIAL=$(basename "${0}") || true

SCRIPT_LOGFILE="/opt/FileMaker/FileMaker Server/Logs/${SCRIPT_LOGFILE_PARTIAL}.log"

# EXTERNAL_LOG_FILENAME=$(basename -- "$fullfile")

EXTERNAL_LOG_EXTENSION="${EXTERNAL_LOG_FILE##*.}"

EXTERNAL_LOG_FILENAME="${EXTERNAL_LOG_FILE%.*}"

EXTERNAL_LOG_ARCHIVE_FILENAME=$(basename -- "${EXTERNAL_LOG_FILENAME}")"_bis_${TODAY}.$
  {EXTERNAL_LOG_EXTENSION}"

EXTERNAL_LOG_CLONE_FILE=$(basename -- "${EXTERNAL_LOG_FILENAME}")" Clone.$
  {EXTERNAL_LOG_EXTENSION}"

FILEMAKER_USER="fmsadmin"
```

```
#!/bin/bash
# set -x
# Setting up variables
TODAY=$(date -I)

# Path definitions
FILEMAKER_BASE_DATA_DIR="/opt/FileMaker/FileMaker Server/Data"
FILEMAKER_BASE_DATA_DATABASES_DIR="${FILEMAKER_BASE_DATA_DIR}/Databases"
FILEMAKER_BACKUP_DIR="${FILEMAKER_BASE_DATA_DIR}/Backups"
FILEMAKER_BACKUP_CLONE_DIR="${FILEMAKER_BACKUP_DIR}/Cloned_by_FMS/Databases"
AUDITLOGARCHIVE_DIR="${FILEMAKER_BASE_DATA_DATABASES_DIR}/AuditLogArchiv"
EXTERNAL_LOG_FILE="ExternesLog.fmp12"

FILEMAKER_EVENT_LOGFILE="/opt/FileMaker/FileMaker Server/Logs/Event.log"
SCRIPT_LOGFILE_PARTIAL=$(basename "${0}") || true
SCRIPT_LOGFILE="/opt/FileMaker/FileMaker Server/Logs/${SCRIPT_LOGFILE_PARTIAL}.log"

# EXTERNAL_LOG_FILENAME=$(basename -- "$fullfile")
EXTERNAL_LOG_EXTENSION="${EXTERNAL_LOG_FILE##*.}"
EXTERNAL_LOG_FILENAME="${EXTERNAL_LOG_FILE%.*}"

EXTERNAL_LOG_ARCHIVE_FILENAME=$(basename -- "${EXTERNAL_LOG_FILENAME}")"_bis_${TODAY}.${EXTERNAL_LOG_EXTENSION}"

EXTERNAL_LOG_CLONE_FILE=$(basename -- "${EXTERNAL_LOG_FILENAME}")" Clone.${EXTERNAL_LOG_EXTENSION}"
FILEMAKER_USER="fmsadmin"
```

```
function log_to_files (){
    if [[ -z "$1" ]] # Did we get a message to log?
    then # Yes, file does exist AND is writable
        LOG_MESSAGE="${0} ${date}: ${1}" || true
        echo "${LOG_MESSAGE}" >> "${SCRIPT_LOGFILE}"
        echo "${LOG_MESSAGE}" >> "${FILEMAKER_EVENT_LOGFILE}"
    fi
}

# Ensure Logfiles
touch "${SCRIPT_LOGFILE}"
touch "${FILEMAKER_EVENT_LOGFILE}"

# Obtain Passwort from Environment OR when passed as first parameter.
# If neither is set, log error message and bail out.
if [[ -z ${FMSADMIN_PASSWORD} ]]
then
    FMSADMIN_PASSWORD="${1}"
    unset 1
else
    log_to_files "FMSADMIN_PASSWORD required but neither set in environment nor passed as first invocation parameter"
    exit 254
fi

# Ensure the backup directory exists
mkdir -p "${FILEMAKER_BACKUP_DIR}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create ${FILEMAKER_BACKUP_DIR}"
fi
unset return_value

# Ensure the Archive directory exists
mkdir -p "${AUDITLOGARCHIVE_DIR}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create ${AUDITLOGARCHIVE_DIR}"
fi
unset return_value

# Run the backup command
# fmsadmin backup "ExternesLog.fmp12" -e -u fmsadmin -p "$PASSWORD"
fmsadmin backup "${EXTERNAL_LOG_FILE}" --dest "filelinux:${FILEMAKER_BACKUP_DIR}/" --keep 0 --clone --username "$
{FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create Filemaker Backup of ${EXTERNAL_LOG_FILE} to ${FILEMAKER_BACKUP_DIR}"
else
    log_to_files "Created backup of ${EXTERNAL_LOG_FILE}"
fi
unset return_value

# Close the Audit Log File
fmsadmin close "${EXTERNAL_LOG_FILE}" --yes --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to close ${EXTERNAL_LOG_FILE}"
else
    log_to_files "Closed ${EXTERNAL_LOG_FILE}"
fi
unset return_value

# Move Audit Log to Archive
mv "${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"  "${AUDITLOGARCHIVE_DIR}/$
{EXTERNAL_LOG_ARCHIVE_FILENAME}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to move ${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE} to ${AUDITLOGARCHIVE_DIR}/$
{EXTERNAL_LOG_ARCHIVE_FILENAME}"
fi
unset return_value


# Move empty clone to Databases
mv "${FILEMAKER_BACKUP_CLONE_DIR}/${EXTERNAL_LOG_CLONE_FILE}" "${FILEMAKER_BASE_DATA_DATABASES_DIR}/$
{EXTERNAL_LOG_FILE}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to move ${FILEMAKER_BACKUP_CLONE_DIR}/${EXTERNAL_LOG_CLONE_FILE} to $
{FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"
fi
unset return_value

# This could take longer than 10s but we have no way to find out properly.
sleep 10

# Future addition: Should check if this is no other open file, since it's likely that this will fail if there is none
other file with the same storage-at-rest passphrase set.

# Open all Filemaker files again. Not just the one we recently closed.
fmsadmin open --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}" >> "${FILEMAKER_EVENT_LOGFILE}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to open files"
else
    log_to_files "Started opening all database files"
fi
unset return_value

# Future addition: Check if both of the new files are actually opened and log an error or retry when they're not?

# EOF
```

```
# Obtain Passwort from Environment OR when passed as first parameter.
# If neither is set, log error message and bail out.
if [[ -z ${FMSADMIN_PASSWORD} ]]
then
    FMSADMIN_PASSWORD="${1}"
    unset 1
else
    log_to_files "FMSADMIN_PASSWORD required but neither set in environment nor passed
  as first invocation parameter"
    exit 254
fi


# Ensure the backup directory exists
mkdir -p "${FILEMAKER_BACKUP_DIR}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create ${FILEMAKER_BACKUP_DIR}"
fi
unset return_value
```

```
#!/bin/bash
# set -x
# Setting up variables
TODAY=$(date -I)

# Path definitions
FILEMAKER_BASE_DATA_DIR="/opt/FileMaker/FileMaker Server/Data"
FILEMAKER_BASE_DATA_DATABASES_DIR="${FILEMAKER_BASE_DATA_DIR}/Databases"
FILEMAKER_BACKUP_DIR="${FILEMAKER_BASE_DATA_DIR}/Backups"
FILEMAKER_BACKUP_CLONE_DIR="${FILEMAKER_BACKUP_DIR}/Cloned_by_FMS/Databases"
AUDITLOGARCHIVE_DIR="${FILEMAKER_BASE_DATA_DATABASES_DIR}/AuditLogArchiv"
EXTERNAL_LOG_FILE="ExternesLog.fmp12"

FILEMAKER_EVENT_LOGFILE="/opt/FileMaker/FileMaker Server/Logs/Event.log"
SCRIPT_LOGFILE_PARTIAL=$(basename "${0}") || true
SCRIPT_LOGFILE="/opt/FileMaker/FileMaker Server/Logs/${SCRIPT_LOGFILE_PARTIAL}.log"

# EXTERNAL_LOG_FILENAME=$(basename -- "$fullfile")
EXTERNAL_LOG_EXTENSION="${EXTERNAL_LOG_FILE##*.}"
EXTERNAL_LOG_FILENAME="${EXTERNAL_LOG_FILE%.*}"

EXTERNAL_LOG_ARCHIVE_FILENAME=$(basename -- "${EXTERNAL_LOG_FILENAME}")"_bis_${TODAY}.${EXTERNAL_LOG_EXTENSION}"

EXTERNAL_LOG_CLONE_FILE=$(basename -- "${EXTERNAL_LOG_FILENAME}")" Clone.${EXTERNAL_LOG_EXTENSION}"
FILEMAKER_USER="fmsadmin"

function log_to_files (){
    if [[ -z "$1" ]] # Did we get a message to log?
    then # Yes, file does exist AND is writable
        LOG_MESSAGE="${0} ${date}: ${1}" || true
        echo "${LOG_MESSAGE}" >> "${SCRIPT_LOGFILE}"
        echo "${LOG_MESSAGE}" >> "${FILEMAKER_EVENT_LOGFILE}"
    fi
}

# Ensure Logfiles
touch "${SCRIPT_LOGFILE}"
touch "${FILEMAKER_EVENT_LOGFILE}"

# Obtain Passwort from Environment OR when passed as first parameter.
# If neither is set, log error message and bail out.
if [[ -z ${FMSADMIN_PASSWORD} ]]
then
    FMSADMIN_PASSWORD="${1}"
    unset 1
else
    log_to_files "FMSADMIN_PASSWORD required but neither set in environment nor passed as first invocation parameter"
    exit 254
fi

# Ensure the backup directory exists
mkdir -p "${FILEMAKER_BACKUP_DIR}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create ${FILEMAKER_BACKUP_DIR}"
fi
unset return_value

# Ensure the Archive directory exists
mkdir -p "${AUDITLOGARCHIVE_DIR}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create ${AUDITLOGARCHIVE_DIR}"
fi
unset return_value

# Run the backup command
# fmsadmin backup "ExternesLog.fmp12" -e -u fmsadmin -p "$PASSWORD"
fmsadmin backup "${EXTERNAL_LOG_FILE}" --dest "filelinux:${FILEMAKER_BACKUP_DIR}/" --keep 0 --clone --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create Filemaker Backup of ${EXTERNAL_LOG_FILE} to ${FILEMAKER_BACKUP_DIR}"
else
    log_to_files "Created backup of ${EXTERNAL_LOG_FILE}"
fi
unset return_value

# Close the Audit Log File
fmsadmin close "${EXTERNAL_LOG_FILE}" --yes --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to close ${EXTERNAL_LOG_FILE}"
else
    log_to_files "Closed ${EXTERNAL_LOG_FILE}"
fi
unset return_value

# Move Audit Log to Archive
mv "${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"  "${AUDITLOGARCHIVE_DIR}/${EXTERNAL_LOG_ARCHIVE_FILENAME}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to move ${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE} to ${AUDITLOGARCHIVE_DIR}/${EXTERNAL_LOG_ARCHIVE_FILENAME}"
fi
unset return_value


# Move empty clone to Databases
mv "${FILEMAKER_BACKUP_CLONE_DIR}/${EXTERNAL_LOG_CLONE_FILE}" "${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to move ${FILEMAKER_BACKUP_CLONE_DIR}/${EXTERNAL_LOG_CLONE_FILE} to ${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"
fi
unset return_value

# This could take longer than 10s but we have no way to find out properly.
sleep 10

# Future addition: Should check if this is no other open file, since it's likely that this will fail if there is none other file with the same storage-at-rest passphrase set.

# Open all Filemaker files again. Not just the one we recently closed.
fmsadmin open --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}" >> "${FILEMAKER_EVENT_LOGFILE}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to open files"
else
    log_to_files "Started opening all database files"
fi
unset return_value

# Future addition: Check if both of the new files are actually opened and log an error or retry when they're not?

# EOF
```

```
# Run the backup command
# fmsadmin backup "ExternesLog.fmp12" -e -u fmsadmin -p "$PASSWORD"
fmsadmin backup "${EXTERNAL_LOG_FILE}" --dest "filelinux:${FILEMAKER_BACKUP_DIR}/" --
  keep 0 --clone --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create Filemaker Backup of ${EXTERNAL_LOG_FILE} to $
  {FILEMAKER_BACKUP_DIR}"
else
    log_to_files "Created backup of ${EXTERNAL_LOG_FILE}"
fi
unset return_value
```

```
#!/bin/bash
# set -x
# Setting up variables
TODAY=$(date -I)

# Path definitions
FILEMAKER_BASE_DATA_DIR="/opt/FileMaker/FileMaker Server/Data"
FILEMAKER_BASE_DATA_DATABASES_DIR="${FILEMAKER_BASE_DATA_DIR}/Databases"
FILEMAKER_BACKUP_DIR="${FILEMAKER_BASE_DATA_DIR}/Backups"
FILEMAKER_BACKUP_CLONE_DIR="${FILEMAKER_BACKUP_DIR}/Cloned_by_FMS/Databases"
AUDITLOGARCHIVE_DIR="${FILEMAKER_BASE_DATA_DATABASES_DIR}/AuditLogArchiv"
EXTERNAL_LOG_FILE="ExternesLog.fmp12"

FILEMAKER_EVENT_LOGFILE="/opt/FileMaker/FileMaker Server/Logs/Event.log"
SCRIPT_LOGFILE_PARTIAL=$(basename "${0}") || true
SCRIPT_LOGFILE="/opt/FileMaker/FileMaker Server/Logs/${SCRIPT_LOGFILE_PARTIAL}.log"

# EXTERNAL_LOG_FILENAME=$(basename -- "$fullfile")
EXTERNAL_LOG_EXTENSION="${EXTERNAL_LOG_FILE##*.}"
EXTERNAL_LOG_FILENAME="${EXTERNAL_LOG_FILE%.*}"

EXTERNAL_LOG_ARCHIVE_FILENAME=$(basename -- "${EXTERNAL_LOG_FILENAME}")"_bis_${TODAY}.${EXTERNAL_LOG_EXTENSION}"

EXTERNAL_LOG_CLONE_FILE=$(basename -- "${EXTERNAL_LOG_FILENAME}")" Clone.${EXTERNAL_LOG_EXTENSION}"
FILEMAKER_USER="fmsadmin"

function log_to_files (){
    if [[ -z "$1" ]] # Did we get a message to log?
    then # Yes, file does exist AND is writable
        LOG_MESSAGE="${0} $(date): ${1}" || true
        echo "${LOG_MESSAGE}" >> "${SCRIPT_LOGFILE}"
        echo "${LOG_MESSAGE}" >> "${FILEMAKER_EVENT_LOGFILE}"
    fi
}

# Ensure Logfiles
touch "${SCRIPT_LOGFILE}"
touch "${FILEMAKER_EVENT_LOGFILE}"

# Obtain Passwort from Environment OR when passed as first parameter.
# If neither is set, log error message and bail out.
if [[ -z ${FMSADMIN_PASSWORD} ]]
then
    FMSADMIN_PASSWORD="${1}"
    unset 1
else
    log_to_files "FMSADMIN_PASSWORD required but neither set in environment nor passed as first invocation parameter"
    exit 254
fi

# Ensure the backup directory exists
mkdir -p "${FILEMAKER_BACKUP_DIR}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create ${FILEMAKER_BACKUP_DIR}"
fi
unset return_value

# Ensure the Archive directory exists
mkdir -p "${AUDITLOGARCHIVE_DIR}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create ${AUDITLOGARCHIVE_DIR}"
fi
unset return_value

# Run the backup command
# fmsadmin backup "ExternesLog.fmp12" -e -u fmsadmin -p "$PASSWORD"
fmsadmin backup "${EXTERNAL_LOG_FILE}" --dest "filelinux:${FILEMAKER_BACKUP_DIR}/" --keep 0 --clone --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create Filemaker Backup of ${EXTERNAL_LOG_FILE} to ${FILEMAKER_BACKUP_DIR}"
else
    log_to_files "Created backup of ${EXTERNAL_LOG_FILE}"
fi
unset return_value

# Close the Audit Log File
fmsadmin close "${EXTERNAL_LOG_FILE}" --yes --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to close ${EXTERNAL_LOG_FILE}"
else
    log_to_files "Closed ${EXTERNAL_LOG_FILE}"
fi
unset return_value

# Move Audit Log to Archive
mv "${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"  "${AUDITLOGARCHIVE_DIR}/${EXTERNAL_LOG_ARCHIVE_FILENAME}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to move ${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE} to ${AUDITLOGARCHIVE_DIR}/${EXTERNAL_LOG_ARCHIVE_FILENAME}"
fi
unset return_value


# Move empty clone to Databases
mv "${FILEMAKER_BACKUP_CLONE_DIR}/${EXTERNAL_LOG_CLONE_FILE}" "${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to move ${FILEMAKER_BACKUP_CLONE_DIR}/${EXTERNAL_LOG_CLONE_FILE} to ${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"
fi
unset return_value

# This could take longer than 10s but we have no way to find out properly.
sleep 10

# Future addition: Should check if this is no other open file, since it's likely that this will fail if there is none other file with the same storage-at-rest passphrase set.

# Open all Filemaker files again. Not just the one we recently closed.
fmsadmin open --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}" >> "${FILEMAKER_EVENT_LOGFILE}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to open files"
else
    log_to_files "Started opening all database files"
fi
unset return_value

# Future addition: Check if both of the new files are actually opened and log an error or retry when they're not?

# EOF
```

```
# Move Audit Log to Archive
mv "${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"   "$
  {AUDITLOGARCHIVE_DIR}/${EXTERNAL_LOG_ARCHIVE_FILENAME}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to move ${FILEMAKER_BASE_DATA_DATABASES_DIR}/$
  {EXTERNAL_LOG_FILE} to ${AUDITLOGARCHIVE_DIR}/${EXTERNAL_LOG_ARCHIVE_FILENAME}"
fi
unset return_value
```

```
#!/bin/bash
# set -x
# Setting up variables
TODAY=$(date -I)

# Path definitions
FILEMAKER_BASE_DATA_DIR="/opt/FileMaker/FileMaker Server/Data"
FILEMAKER_BASE_DATA_DATABASES_DIR="${FILEMAKER_BASE_DATA_DIR}/Databases"
FILEMAKER_BACKUP_DIR="${FILEMAKER_BASE_DATA_DIR}/Backups"
FILEMAKER_BACKUP_CLONE_DIR="${FILEMAKER_BACKUP_DIR}/Cloned_by_FMS/Databases"
AUDITLOGARCHIVE_DIR="${FILEMAKER_BASE_DATA_DATABASES_DIR}/AuditLogArchiv"
EXTERNAL_LOG_FILE="ExternesLog.fmp12"

FILEMAKER_EVENT_LOGFILE="/opt/FileMaker/FileMaker Server/Logs/Event.log"
SCRIPT_LOGFILE_PARTIAL=$(basename "${0}") || true
SCRIPT_LOGFILE="/opt/FileMaker/FileMaker Server/Logs/${SCRIPT_LOGFILE_PARTIAL}.log"

# EXTERNAL_LOG_FILENAME=$(basename -- "$fullfile")
EXTERNAL_LOG_EXTENSION="${EXTERNAL_LOG_FILE##*.}"
EXTERNAL_LOG_FILENAME="${EXTERNAL_LOG_FILE%.*}"

EXTERNAL_LOG_ARCHIVE_FILENAME=$(basename -- "${EXTERNAL_LOG_FILENAME}")"_bis_${TODAY}.${EXTERNAL_LOG_EXTENSION}"

EXTERNAL_LOG_CLONE_FILE=$(basename -- "${EXTERNAL_LOG_FILENAME}")" Clone.${EXTERNAL_LOG_EXTENSION}"
FILEMAKER_USER="fmsadmin"

function log_to_files (){
    if [[ -z "$1" ]] # Did we get a message to log?
    then # Yes, file does exist AND is writable
        LOG_MESSAGE="${0} $(date): ${1}" || true
        echo "${LOG_MESSAGE}" >> "${SCRIPT_LOGFILE}"
        echo "${LOG_MESSAGE}" >> "${FILEMAKER_EVENT_LOGFILE}"
    fi
}

# Ensure Logfiles
touch "${SCRIPT_LOGFILE}"
touch "${FILEMAKER_EVENT_LOGFILE}"

# Obtain Passwort from Environment OR when passed as first parameter.
# If neither is set, log error message and bail out.
if [[ -z ${FMSADMIN_PASSWORD} ]]
then
    FMSADMIN_PASSWORD="${1}"
    unset 1
else
    log_to_files "FMSADMIN_PASSWORD required but neither set in environment nor passed as first invocation parameter"
    exit 254
fi

# Ensure the backup directory exists
mkdir -p "${FILEMAKER_BACKUP_DIR}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create ${FILEMAKER_BACKUP_DIR}"
fi
unset return_value

# Ensure the Archive directory exists
mkdir -p "${AUDITLOGARCHIVE_DIR}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create ${AUDITLOGARCHIVE_DIR}"
fi
unset return_value

# Run the backup command
# fmsadmin backup "ExternesLog.fmp12" -e -u fmsadmin -p "$PASSWORD"
fmsadmin backup "${EXTERNAL_LOG_FILE}" --dest "filelinux:${FILEMAKER_BACKUP_DIR}/" --keep 0 --clone --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create Filemaker Backup of ${EXTERNAL_LOG_FILE} to ${FILEMAKER_BACKUP_DIR}"
else
    log_to_files "Created backup of ${EXTERNAL_LOG_FILE}"
fi
unset return_value

# Close the Audit Log File
fmsadmin close "${EXTERNAL_LOG_FILE}" --yes --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to close ${EXTERNAL_LOG_FILE}"
else
    log_to_files "Closed ${EXTERNAL_LOG_FILE}"
fi
unset return_value

# Move Audit Log to Archive
mv "${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"  "${AUDITLOGARCHIVE_DIR}/${EXTERNAL_LOG_ARCHIVE_FILENAME}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to move ${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE} to ${AUDITLOGARCHIVE_DIR}/${EXTERNAL_LOG_ARCHIVE_FILENAME}"
fi
unset return_value

# Move empty clone to Databases
mv "${FILEMAKER_BACKUP_CLONE_DIR}/${EXTERNAL_LOG_CLONE_FILE}" "${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to move ${FILEMAKER_BACKUP_CLONE_DIR}/${EXTERNAL_LOG_CLONE_FILE} to ${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"
fi
unset return_value

# This could take longer than 10s but we have no way to find out properly.
sleep 10

# Future addition: Should check if this is no other open file, since it's likely that this will fail if there is none other file with the same storage-at-rest passphrase set.

# Open all Filemaker files again. Not just the one we recently closed.
fmsadmin open --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}" >> "${FILEMAKER_EVENT_LOGFILE}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to open files"
else
    log_to_files "Started opening all database files"
fi
unset return_value

# Future addition: Check if both of the new files are actually opened and log an error or retry when they're not?

# EOF
```

```
# Open all Filemaker files again. Not just the one we recently closed.
fmsadmin open --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}" >> "$
  {FILEMAKER_EVENT_LOGFILE}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to open files"
else
    log_to_files "Started opening all database files"
fi
unset return_value

# Future addition: Check if both of the new files are actually opened and log an error
  or retry when they're not?

# EOF
```

```
#!/bin/bash
# set -x
# Setting up variables
TODAY=$(date -I)

# Path definitions
FILEMAKER_BASE_DATA_DIR="/opt/FileMaker/FileMaker Server/Data"
FILEMAKER_BASE_DATA_DATABASES_DIR="${FILEMAKER_BASE_DATA_DIR}/Databases"
FILEMAKER_BACKUP_DIR="${FILEMAKER_BASE_DATA_DIR}/Backups"
FILEMAKER_BACKUP_CLONE_DIR="${FILEMAKER_BACKUP_DIR}/Cloned_by_FMS/Databases"
AUDITLOGARCHIVE_DIR="${FILEMAKER_BASE_DATA_DATABASES_DIR}/AuditLogArchiv"
EXTERNAL_LOG_FILE="ExternesLog.fmp12"

FILEMAKER_EVENT_LOGFILE="/opt/FileMaker/FileMaker Server/Logs/Event.log"
SCRIPT_LOGFILE_PARTIAL=$(basename "${0}") || true
SCRIPT_LOGFILE="/opt/FileMaker/FileMaker Server/Logs/${SCRIPT_LOGFILE_PARTIAL}.log"

# EXTERNAL_LOG_FILENAME=$(basename -- "$fullfile")
EXTERNAL_LOG_EXTENSION="${EXTERNAL_LOG_FILE##*.}"
EXTERNAL_LOG_FILENAME="${EXTERNAL_LOG_FILE%.*}"

EXTERNAL_LOG_ARCHIVE_FILENAME=$(basename -- "${EXTERNAL_LOG_FILENAME}")"_bis_${TODAY}.${EXTERNAL_LOG_EXTENSION}"

EXTERNAL_LOG_CLONE_FILE=$(basename -- "${EXTERNAL_LOG_FILENAME}")" Clone.${EXTERNAL_LOG_EXTENSION}"
FILEMAKER_USER="fmsadmin"

function log_to_files (){
    if [[ -z "$1" ]] # Did we get a message to log?
    then # Yes, file does exist AND is writable
        LOG_MESSAGE="${0} ${date}: ${1}" || true
        echo "${LOG_MESSAGE}" >> "${SCRIPT_LOGFILE}"
        echo "${LOG_MESSAGE}" >> "${FILEMAKER_EVENT_LOGFILE}"
    fi
}

# Ensure Logfiles
touch "${SCRIPT_LOGFILE}"
touch "${FILEMAKER_EVENT_LOGFILE}"

# Obtain Passwort from Environment OR when passed as first parameter.
# If neither is set, log error message and bail out.
if [[ -z ${FMSADMIN_PASSWORD} ]]
then
    FMSADMIN_PASSWORD="${1}"
    unset 1
else
    log_to_files "FMSADMIN_PASSWORD required but neither set in environment nor passed as first invocation parameter"
    exit 254
fi

# Ensure the backup directory exists
mkdir -p "${FILEMAKER_BACKUP_DIR}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create ${FILEMAKER_BACKUP_DIR}"
fi
unset return_value

# Ensure the Archive directory exists
mkdir -p "${AUDITLOGARCHIVE_DIR}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create ${AUDITLOGARCHIVE_DIR}"
fi
unset return_value

# Run the backup command
# fmsadmin backup "ExternesLog.fmp12" -e -u fmsadmin -p "$PASSWORD"
fmsadmin backup "${EXTERNAL_LOG_FILE}" --dest "filelinux:${FILEMAKER_BACKUP_DIR}/" --keep 0 --clone --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to create Filemaker Backup of ${EXTERNAL_LOG_FILE} to ${FILEMAKER_BACKUP_DIR}"
else
    log_to_files "Created backup of ${EXTERNAL_LOG_FILE}"
fi
unset return_value

# Close the Audit Log File
fmsadmin close "${EXTERNAL_LOG_FILE}" --yes --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to close ${EXTERNAL_LOG_FILE}"
else
    log_to_files "Closed ${EXTERNAL_LOG_FILE}"
fi
unset return_value

# Move Audit Log to Archive
mv "${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"  "${AUDITLOGARCHIVE_DIR}/${EXTERNAL_LOG_ARCHIVE_FILENAME}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to move ${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE} to ${AUDITLOGARCHIVE_DIR}/${EXTERNAL_LOG_ARCHIVE_FILENAME}"
fi
unset return_value


# Move empty clone to Databases
mv "${FILEMAKER_BACKUP_CLONE_DIR}/${EXTERNAL_LOG_CLONE_FILE}" "${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to move ${FILEMAKER_BACKUP_CLONE_DIR}/${EXTERNAL_LOG_CLONE_FILE} to ${FILEMAKER_BASE_DATA_DATABASES_DIR}/${EXTERNAL_LOG_FILE}"
fi
unset return_value

# This could take longer than 10s but we have no way to find out properly.
sleep 10

# Future addition: Should check if this is no other open file, since it's likely that this will fail if there is none other file with the same storage-at-rest passphrase set.

# Open all Filemaker files again. Not just the one we recently closed.
fmsadmin open --username "${FILEMAKER_USER}" --password "${FMSADMIN_PASSWORD}" >> "${FILEMAKER_EVENT_LOGFILE}"
return_value=$?
if [[ ${return_value} -ne 0 ]]
then
    log_to_files "Failed to open files"
else
    log_to_files "Started opening all database files"
fi
unset return_value

# Future addition: Check if both of the new files are actually opened and log an error or retry when they're not?

# EOF
```

Viel Spass beim Coden!

Vielen Dank unseren Sponsoren und Konferenz-Partnern

Claris

FileMaker Magazin

24U SOFTWARE

CamelCase
camelcase.com

CAOS
culture agency office software