4D Database Recovery Options – Part I

By Tai Bui, Technical Services Engineer, 4D Inc.

Technical Note 18-09

Table of Contents

Introduction

A 4D database can be an integral part of any system or business operations as it can simply be a tool to store data and go above and beyond to work with the data and provide services for users. However, as Murphy's Law states, "Anything that can go wrong will go wrong." As such it is always important to have a contingency plan and know what to do in cases of emergencies and accidents. There are a number of issues that can happen and there are a number of ways to handle them to reduce the amount of time the 4D database is down. This Technical Note will be the first of a two-part series. The first part will be a general informative overview that will go over 4D's features that can be utilized to cut downtime in case a database is damaged. These features are the backup system and how to set up and use a mirror. The following second part will go in-depth about the mirroring process and provide more details suggestions and examples of how to implement a mirror for different scenarios.

4D Database

The 4D database is usually a system that collects and organizes data. It can then be enhanced to provide additional features and services for users to interact with the data. The database is comprised of two main files: the structure file and the data file. The structure file can come in two formats interpreted with the ".4DB" file extension or compiled with the ".4DC" file extension while the data file only has one format with the ".4DD" file extension. There are also a number of other files that a 4D database can contain such as the data index file ".4DIndx".

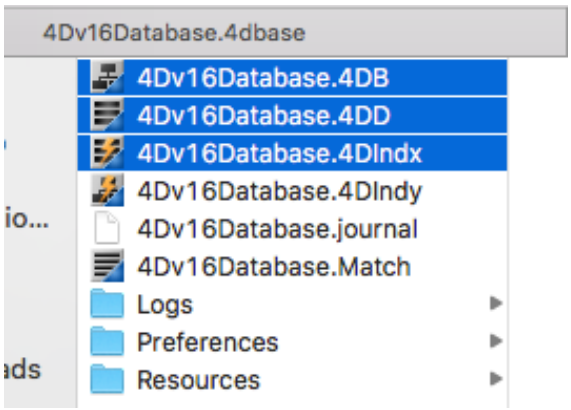

Image 1: Files of a 4D Database

Database Corruption or Damage

If any of a 4D database's files become damaged or corrupt it can prevent the database from opening up or cause errors during use. There are a number of unforeseen events that can cause this such as a crash, hardware failure, or even a power outage. These events can abruptly quit the application while it may be

in the middle of modifying something important, performing a flush, or modifying the data file. When an event like this occurs and the database is not able to run properly due to damages it becomes a battle against time to start up the database as soon as possible as users have no access to the data and services until it is back up again.

4D has a number of tools and features that can be done to bring back up the database with as much of the most recent data as possible. While it can take a bit of time and resources to set up, it is always better to be safe than sorry. The following content will explain ways to recover and what is needed to be prepared and implemented ahead of time in case the feature is needed. This will reduce the time a database is down and can prevent data loss conclusively saving the users and companies from losing more money and mitigate the negative impact of an event.

Maintenance and Security Center

One of the, if not the, first things that should be done when an issue occurs is to open up the database with the Maintenance and Security Center (MSC) and perform a verify on both the structure and the data to see if there are any issues that 4D can identify. If the database cannot be opened, the database can be opened in MSC without opening the database first by trying to open up MSC after starting 4D or 4D Server. This is done with the normal methods without a database opened through the "Help" menu or through the toolbar if it is displayed.

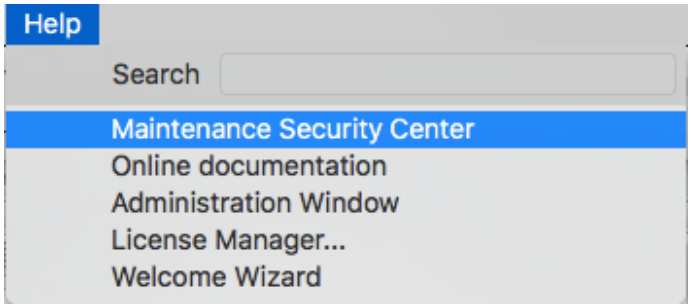

Image 2: Opening MSC through the Help Menu

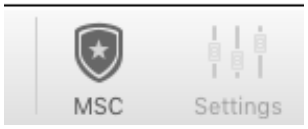

Image 3: MSC Icon in the Tool Bar, located on the left

The option can also be selected when opening a local database from the menu and changing how it is opened.

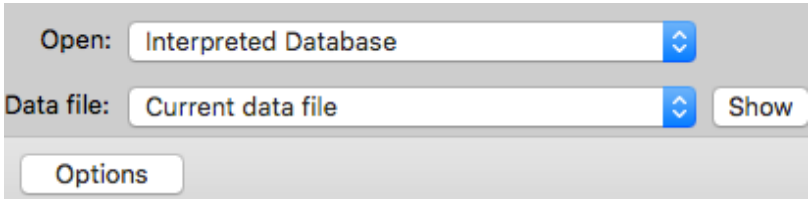

Image 4: Opening options dropdown when selecting a local database

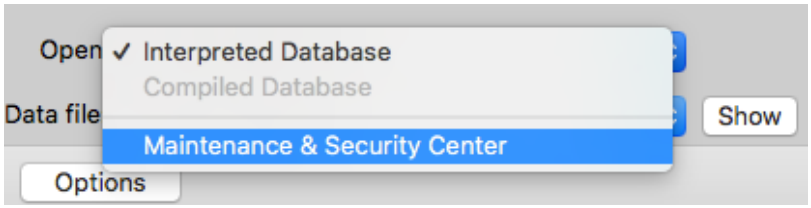

Image 5: The MSC option in the drop-down of the opening options

Verify

A verification can be performed on any of the major parts of the database, the data file (records), the data index file (indexes), and the structure file (application). The verification can return a detection of Errors, Warnings, or if everything appears to be ok from the verification an OK. The results will then be logged which can be viewed in XML format.

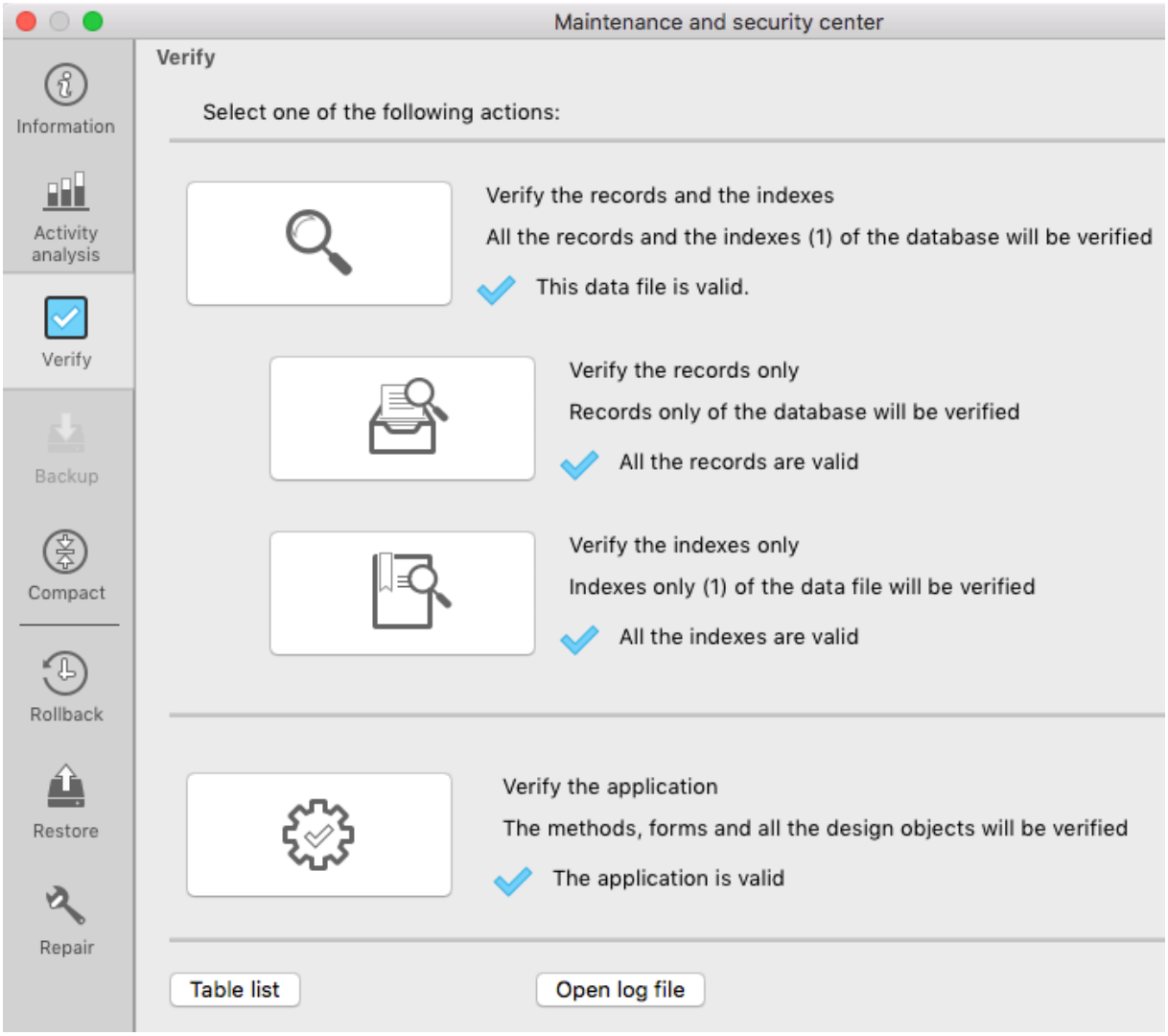

Image 6: Verification Tab of the MSC returning valid results

Repair

If there are issues found, they may be repairable through the MSC's repair functions. A repair can be performed on the data file and the structure file. Before a repair is performed, it can be useful to create a copy of the entire database. Depending on the errors found during the verification the associated repair should be performed. A repair can be performed even if the verification does not find any errors; in this scenario, it would save time to abstain from performing the verification. After each repair, a log is generated which can be reviewed also verification should be performed to confirm that the repair is fixing errors. In cases, repairs may result in new errors coming up due to that were not visible and repairable due to the prior damages, during these situations the process should be repeated until the errors are all fixed.

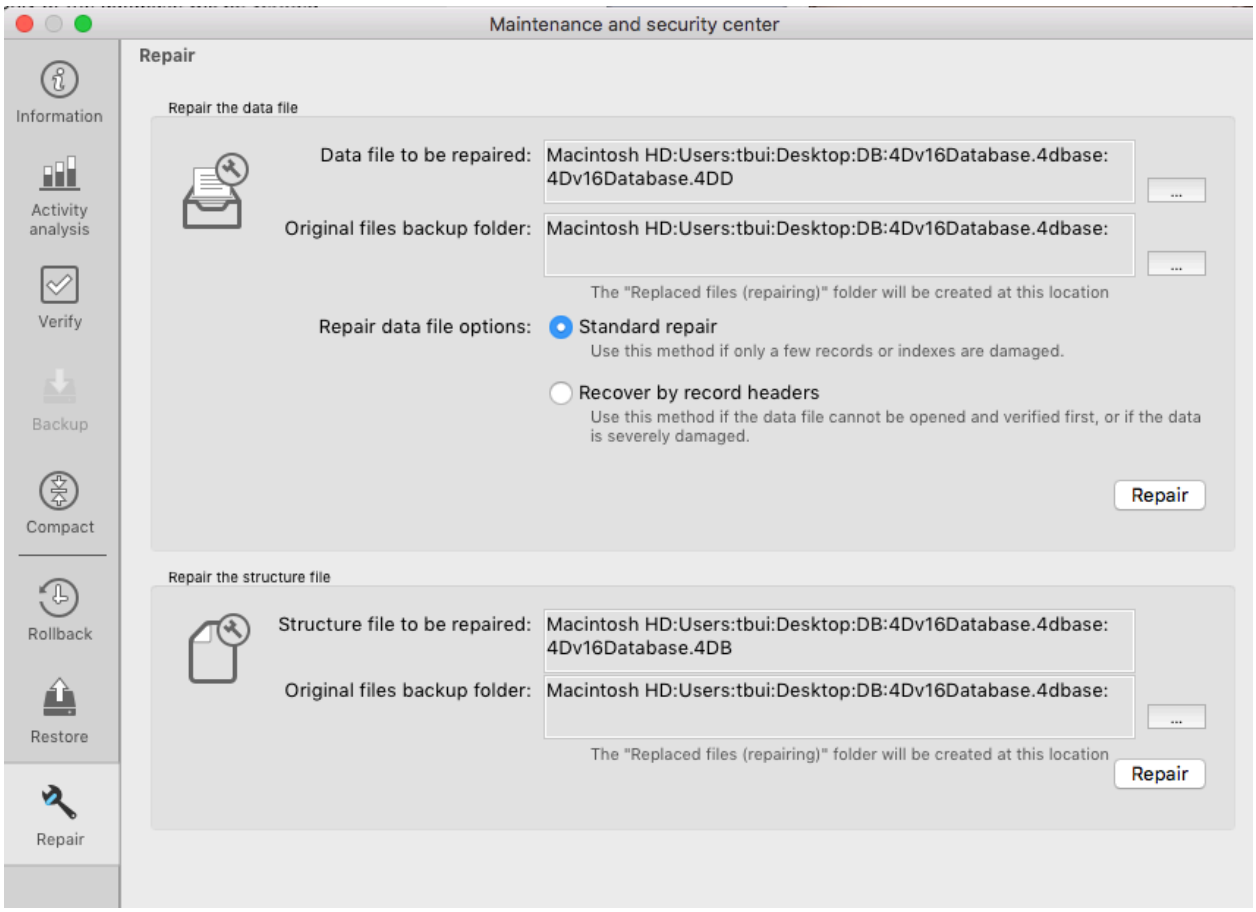

Image 7: Repair Tab of the MSC

The structure file only has one repair while the data file has two repair options. When performing repairs there will be a need for free hard drive space. When the structure is repaired, a copy of the structure file and the structure index is saved in a time-stamped "Replaced Files…" folder and then repaired and replaced in the main location. For the data file options, it is recommended that the Standard repair option is attempted first; if the repair does not work then the Recover by record headers should be the next option. When a repair on the data file is performed the data file, data index, and the journal files are placed in time-stamped "Replaced Files…" folder and then the data file is repaired and replaced. A new data index is not generated until the database is opened and a new journal file is not generated. The journal file can be found in the "Replaced Files…" folder and copied back. If the journal file is not found the database will generate a warning about the journal file and allow the options of creating a new journal file or locating and selecting an existing journal file.

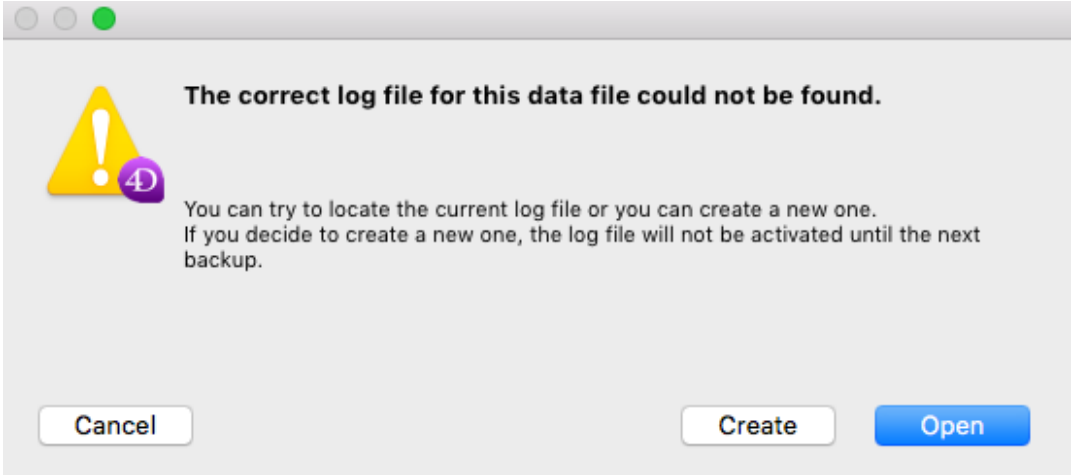

Image 8: Error Dialog when Log File is missing

With the MSC Verify and Repair features, they do not require any set up prior to an issue. After an issue arises, a verify can be performed or alternatively, the verify can be postponed until after an initial repair. In some occasions, multiple repairs may be needed to completely remove all errors as prior errors prevented the detection of other errors. The repair system is not a catchall solution. If the files are too damaged or corrupted they may not be able to be fixed with MSC's repair functions. In this case, it would have been beneficial to set up a "plan B".

Backup System

4D has a backup system that provides numerous options that allow the feature to be customized to better fit the application's use. The system has become even more robust and reliable since the rework of the logging system in 4Dv14 which uses primary keys as the main way to identify and keep track of records. The system generates logs of changes to the data file using the primary keys with the current log being the ".journal" file. When a backup is performed, a backup file ".4BK" and a backup of the journal file ".4BL" are generated.

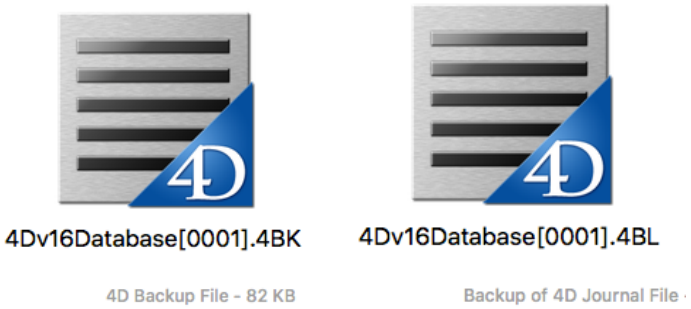

Image 9: Examples of a .4BK and .4BL

The backup file contains the backup files selected in the backup configuration of the database settings. By default, the data file and the structure file are included.

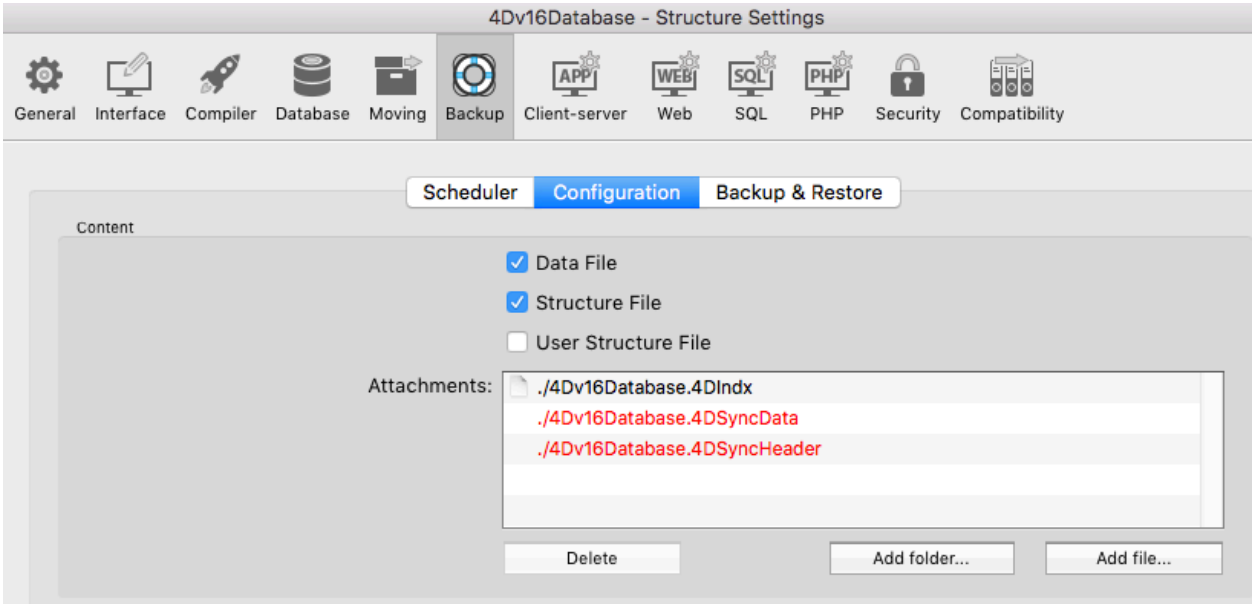

Image 10: Backup Configuration Page of the Structure Settings

When a major incident that cannot be remedied by the MSC's repair it is possible to restore from a backup and then integrate any logs to restore the most recent logged data file changes.

Preparing Backups

Backups are a feature that requires preparation. As explained, backups are restored from a backup file. If a backup does not exist it is not possible to restore from a backup. To prepare a database to be able to use backups and restores, there is not a lot of work required. First off, a backup does take up space as it contains a copy of the selected files in the options. A database with a large data file will use more space per backup file generated so space should be planned for the backup files. Backups can be generated in many ways not only be code through the **BACKUP** command but as depicted in the following images.

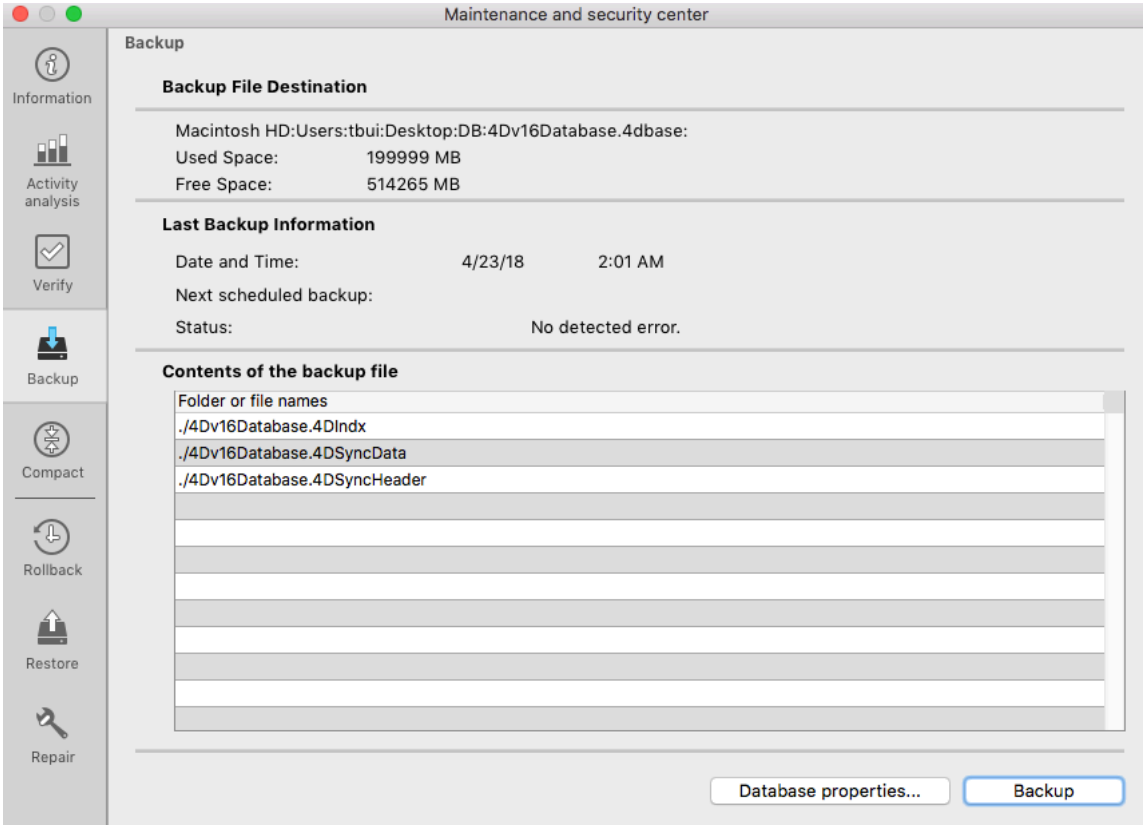

Image 11: Backup Tab of MSC

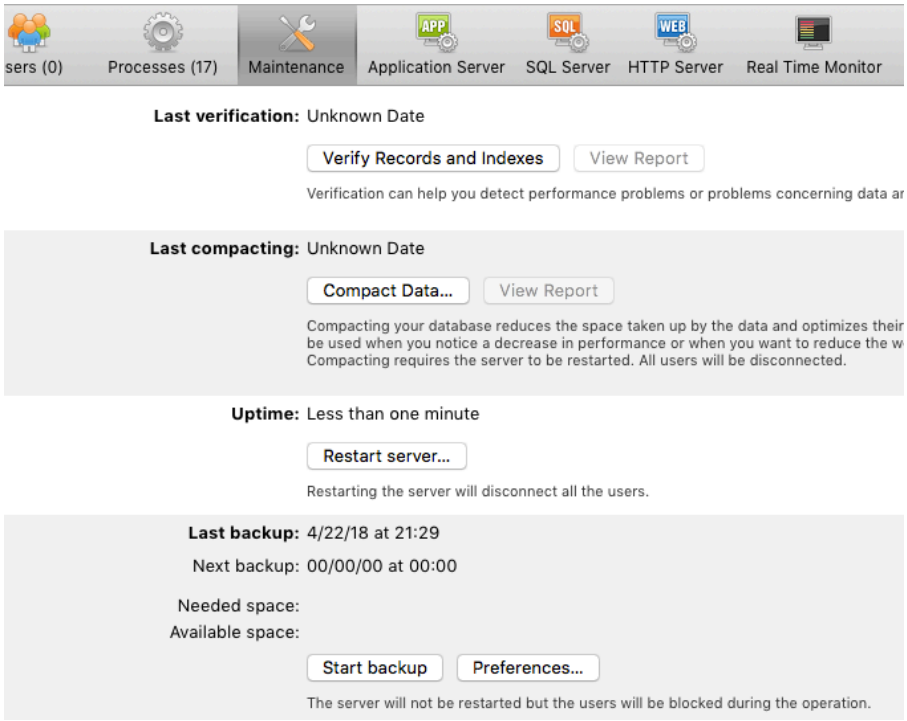

Image 12: Maintenance tab of the 4D Server's Admin Window

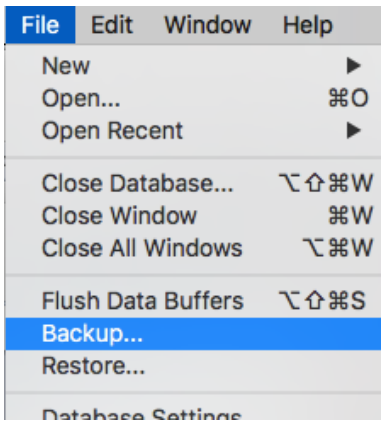

Image 13: The File menu

Backups can also be scheduled to occur automatically in the Backup Scheduler options of the database settings. Backups do lock the database during the process so with larger databases it is usually preferred to plan the backups to occur during a time of low activity.

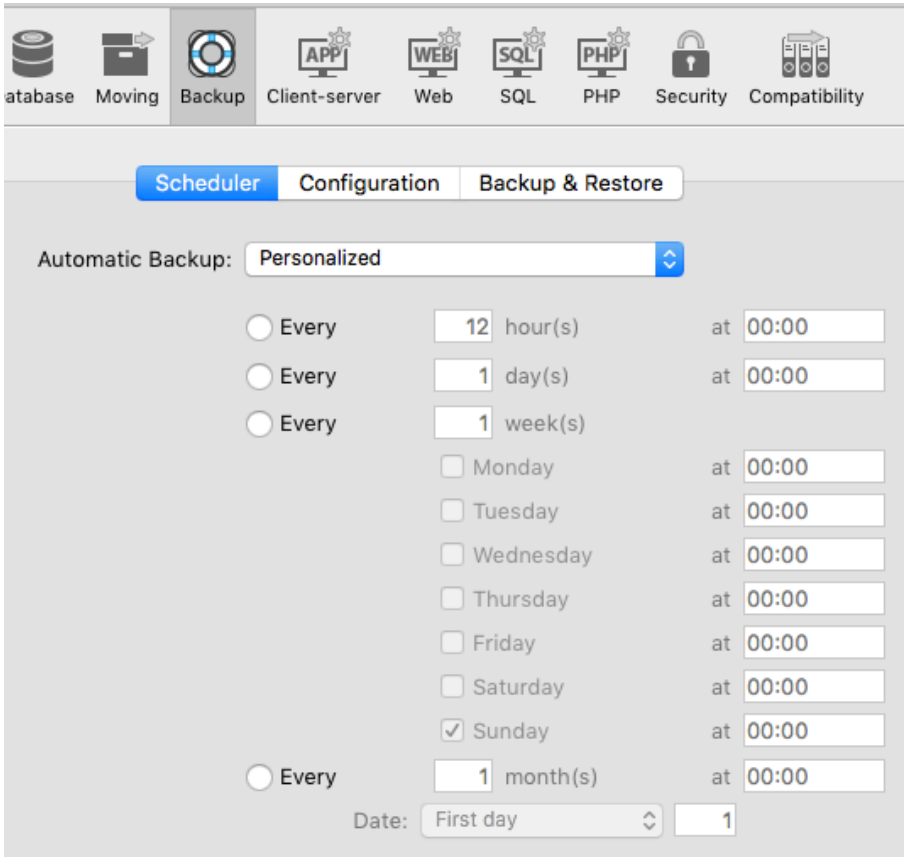

Image 14: Backup Scheduler Page of the Structure Settings

Setting up a database to be able to generate backups regularly should be a highly considered feature. All of the backup settings are stored in an external file located in the {database folder}\Preferences\Backup\Backup.xml so it is important to try to maintain this file during updates and restores. Having backups not only allow backups to be restored but can also allow a rollback to be performed. Both of these features are useful in a number of scenarios where an incident occurs.

Restoring and Rollbacks

With the availability of backups, there are now options for recovery. The two main functions that having backups can provide are the ability to restore a backup or perform a rollback. When restoring, the backup can be thought of as a compressed archive or zip file generated and restoring is unzipping the older version of the database where it was working properly before. A rollback can be considered as performing an undo on any actions on the data file.

Restoring from a Backup

When a database becomes damaged or something occurs where a return to an earlier state of the database is needed, a restore can be performed from one of the backups for the database. A restore can be performed from the file menu, by code, or through the MSC; which is the easiest way to perform a restore. In most of the methods of restoring, the files are restored to a designated location. The files are all in the state that they were at the time of generation of the selected backup. Usually, the most recent backup is selected. But in some cases, an older backup may be selected. Using the MSC will provide an option to integrate log file which will allow the latest operations that were not backed up to be integrated into the restored database reducing the amount of data loss.

Performing a Rollback

With backups used a rollback can be performed from the MSC. A rollback essentially restores a backup and then integrates changes up to a selected action. For example, if an accident occurs where a large number of records were accidentally deleted or changed, but there were still a couple of hours of work done. Opening up MSC and going to the rollback tab will list the actions performed on the data file and, if a prior backup exists, will allow an action to be selected to return to and undo the actions performed afterward. If a backup does not exist, the rollback page of the MSC will simply display the actions but will not allow a rollback to be performed.

A rollback can only be performed on the current log file, if a rollback is needed on an older log file, then the database must first be restored to the point where the desired action was in the current log file.

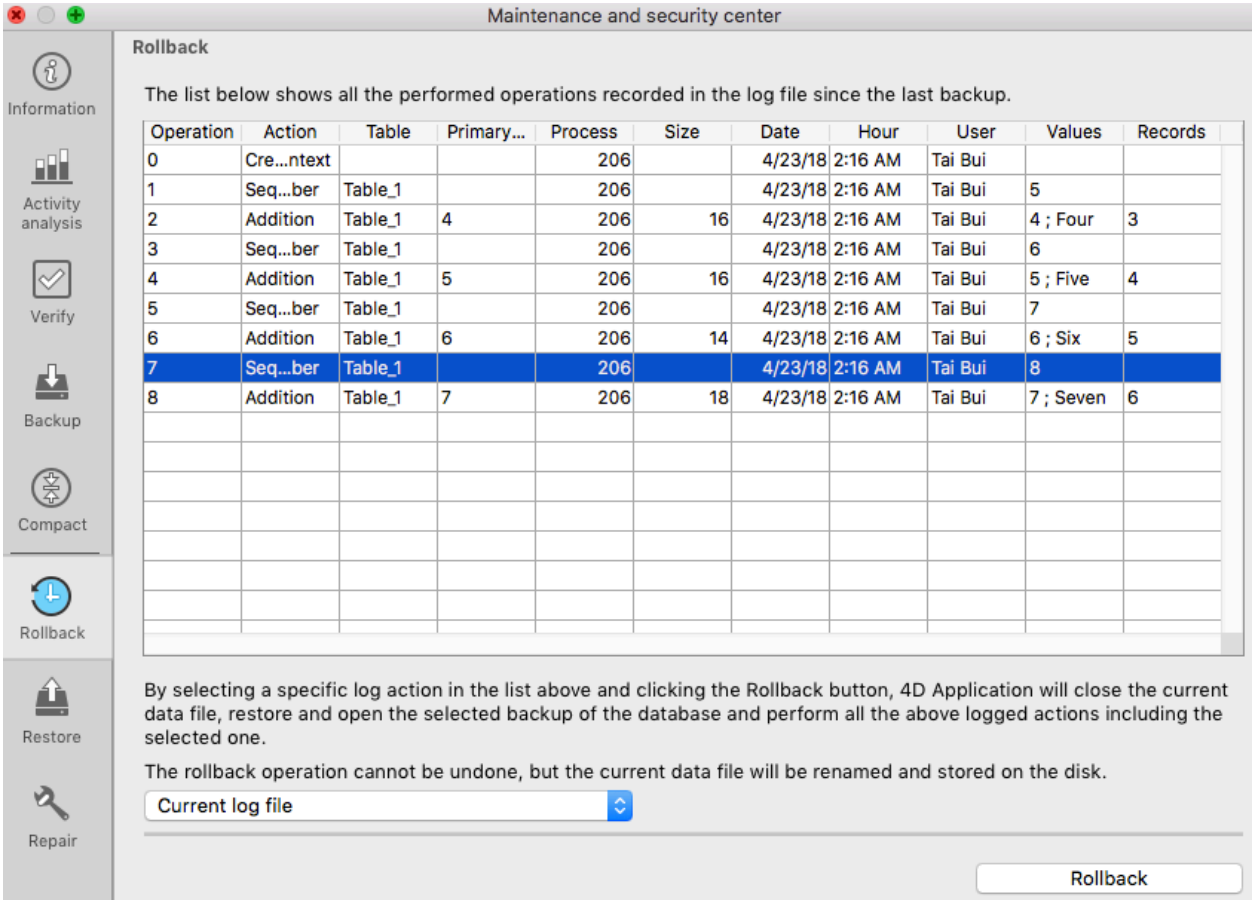

Image 15: Rollback Page of the MSC

Mirroring

Mirroring is a more advanced and possibly complex safety mechanism that can be implemented to any 4D database ran as a server. In 4D, mirroring is performed by setting up a second instance of a copy of the database. Then when any data actions are performed on the main server in use, the changes are "mirrored" on the second instance. This is done by generating log files and passing the logs to the mirror, which will integrate the changes. This allows one of the most rapid ways to bring back up a 4D database in case of an emergency if done correctly as a working database with the latest changes exists in an uncorrupted and undamaged condition. It also removes the issue of the database being locked during a backup process as there are no major locking processes in a mirror.

Settings up a Mirror

4D does not have settings and configurations to automatically prepare a mirror for use. Mirroring must be implemented ahead of time and coded into the database. There is not one best way to set up a mirror and it can be done in

many ways however there are a number of items that must be done to set up mirroring.

Identifying Self

To start with setting up mirroring, since the databases are the same database with the mirror being an additional copy of the database there needs to be a way for the databases to identify whether it is a mirror or the main database. This is best done on the On Server Startup database method. A simple way to do this is to check for a file that should only exist on the mirror such as an empty text file with a specific name. This can be more elaborate such as the existence of a folder or checking the contents of a configuration file. The main and mirror databases should be on two different machines in case of hardware issues.

For example:

```
      //On Server Startup Method:
      //…
C_TEXT($path_t)
$path_t:= Get 4D Folder(Current resources folder)
$path_t:= $path_t + Folder Separator
$path_t:= $path_t + "IsMirror.txt"

If(Test path name($path_t)=Is a document)
      //Is Mirror Server
Else
      //Is Main Server
End if
      //…
```

Generate Logs

If the database is the Main database that is being use it will need to generate logs. Generation of the logs can be done in many ways using the **New log file** command. The command generates a log file returning the full path to the file which is stored in the same location as the current log file appended by the number of log files generated since the start of the current chain of logs. Typically, it can be done procedurally during a short and frequent interval such as every 2-5 minutes. The more frequent the generation of the logs are made the more intact and recent the data on the mirror will be. Also keep in mind that stored procedures should be able to gracefully stop when the database needs to be stopped.

For example:

```
      // Mirror_GenLogs
      // Generates Logs for the mirror every 2 minutes
C_BOOLEAN(stopFlag_b)
```

```
C_TEXT($logPath_t)
stopFlag_b:= False

REPEAT
      $logPath_t:= New log file
      DELAY PROCESS(CURRENT PROCESS; 60*60*2)
      //60 ticks=1 second
UNTIL(stopFlag_b)| (Process aborted)
```

Sharing Logs with the Mirror

To integrate the logs into the mirror there must be a mechanism implemented to share the logs with the mirror. There are limitless ways to do this since this is a case of file transfer or file sharing. If the files are on a shared network it can be as simple as calling the **COPY DOCUMENT** command to copy the file over to the new location or can be more complex. It can be made the responsibility of the Main database to send the logs over or made the responsibility of the Mirror to obtain the logs. Usually, the main server will have the responsibility of sending the files as it knows when a log has been generated. Modifying the earlier example code, additional code can be added after the generation of the log file to call a method to perform that log transfer.

```
      // Mirror_GenLogs
      //…
REPEAT
      $logPath_t:= New log file
      Mirror_SendLog($logPath_t)
      //Sends log file to Mirror location
      DELAY PROCESS(CURRENT PROCESS; 60*60*2)
      //60 ticks=1 second
UNTIL(stopFlag_b)| (Process aborted)
      //…
```

Integrating the Logs

If the database is the mirror database, then it should be regularly trying to check to see if there are any new logs to be integrated and integrate them if so. From the previous implementation of a method to share the logs with the mirror, the mirror should be procedurally checking the location for a new log file in a short interval. The mirror should also have a system of keeping track of which logs have been integrated so can know which logs are new and need to be integrated and which logs are not.

When new logs are found, they will get integrated using the **INTEGRATE MIRROR LOG FILE** command. This command takes in two required parameters and two optional parameters. The first parameter is the path to the log that should be integrated. The second parameter is the number of the last operation integrated from the last log file and which gets updated with a

number of the last operation of the current execution and should be maintained unless the first log file is being integrated. The third parameter, which is optional, is to select whether the integration should be using strict mode or auto repair mode. In strict mode, if an error is found the integration stops and MSC should be used to investigate the case. This is the default and recommended mode to confirm the validity of the data. To maintain the mirror and continue the integration auto repair mode can be used to automatically repair any errors found. If errors are found, the information will be passed into the fourth optional object parameter of the command.

Preparation for Mirror as a Backup

If an issue does occur on the Main database, the mirror will need to step in and act as the Main database until the situation can be sorted out. Plans will be needed to reroute all connections from the main database's address to the mirror's address. This can be simple or complex depending on the number of connections and dependent systems from client connections to SQL server connections and more.

Running the Mirror

To run the databases first the main database needs to be started and backed up initiating the log file. Then the database should be closed, and all of the database and files should be copied over to the mirror location. The main database can then be started up running its processes to generate log files. At the same time, the mirror can then be started up. The log files will be sent to a location where the mirror database is observing for new log files in which the mirror will integrate the new log files resulting in a secondary database with the data as the main database.

If an emergency situation occurs on the Main database, the mirror server can be used its current machine to function as the main server until the main machine is cleared for use. In which case the steps should be repeated again starting from a backup.

The process of setting up a mirror allows for multiple implementations, but the core functions must be followed. The Main server should generate logs regularly and send them to the mirror server which regularly integrates them. Also, mirrors are not limited to one. Multiple mirrors can be implemented depending on the desired level of peace of mind desired in multiple ways. Mirrors can be chained up, with a mirror for the mirror, which provides an additional failover for the mirror acting as the main server or multiple mirrors can be set up for a single server. While setting up a mirror may be more difficult and require more resources it does provide a way to keep the database up 24/7 and provide a failover in case of an emergency such as a hospital database.

Conclusion

This Technical Note provided information on how to prepare and act in case of an emergency situation in which the database becomes unusable. In a case of a simple damage to the files, the MSC's repair functions may be able to repair the database. If the damage were more severe, then it would be helpful to have either a backup to restore from or a mirror to use while the situation is being handled. In a case of an accidental data change that cannot be easily reversed, having a backup can also allow a rollback to be performed. All of these actions will reduce the amount of downtime to bring back up the database with as much of the data as possible. If the idea of setting up a mirror is appealing and more information is needed, look for the second part of this two part Technical Note which will provide more details and examples on how to set up a mirror for 4D.