Technical documentation

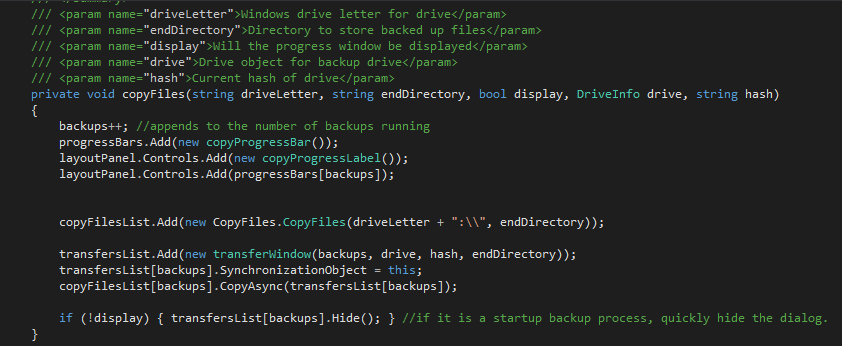
# System Overview

Saviour Backup System is a simple, lightweight and easy to use backup system for portable USB drives. Using this software, allowing back up of portable devices to a computer, so that if the drive is damaged or lost in any way, all the data is retained.

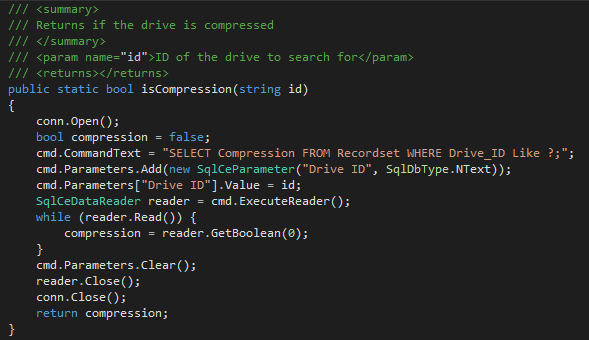
The system stores its data in an SQLCE database located inside the installation directory. This database is encrypted using a password to stop unauthorised editing. The database holds records for the drives, and backup information for the drive, along with records of every backup performed. The software is written in C# using the .NET runtime version 4.5 and compiled using visual studio.

# Algorithms

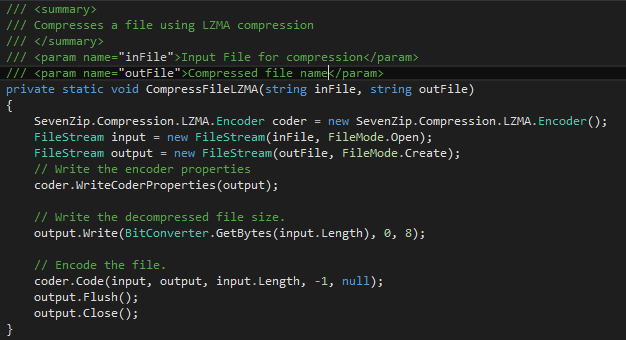
## Copy Files to system (CurrentTransfers Line 67)



## Check if user wants compression (DatabaseTools Line 198)



# Compress data on drive (Compression Line 55)



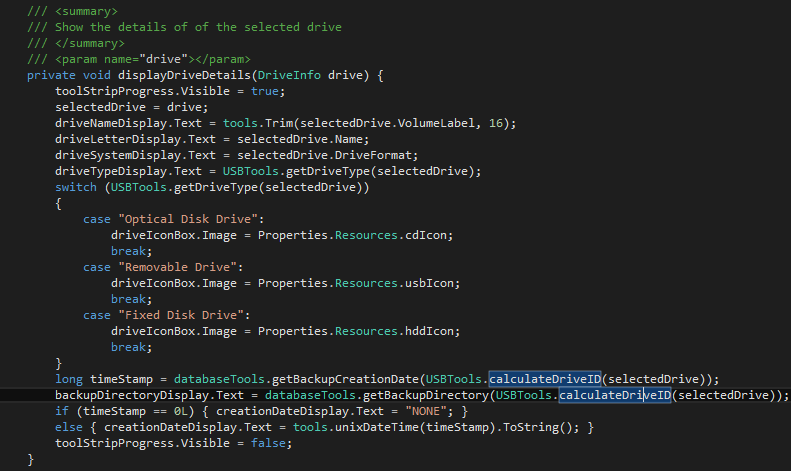
## Detect logged in user (Setup line 23)



## Check if backup record exists (MainWindow line 55)

## 

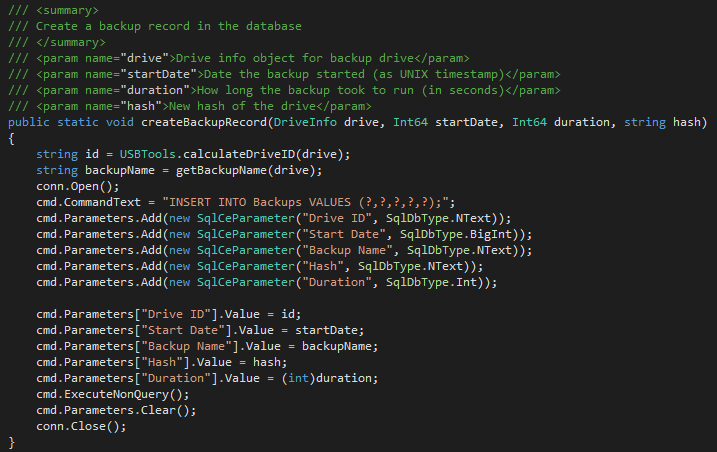
## Get drive details from USB (MainWindow Line 55)



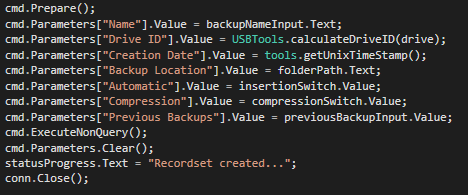
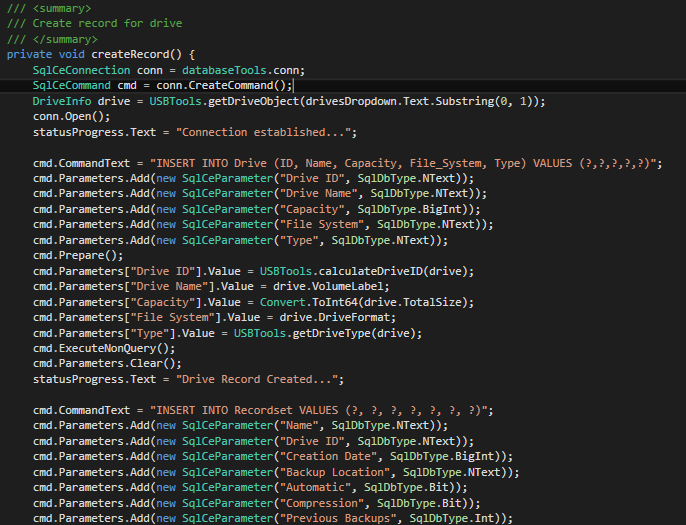
## Check a database exists



## Store information for backup (DatabaseTools line 146)



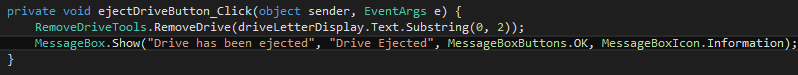
## Store information for new backup record (addbackupwizard line 155)



## Generate Drive ID (USBTools line 105)



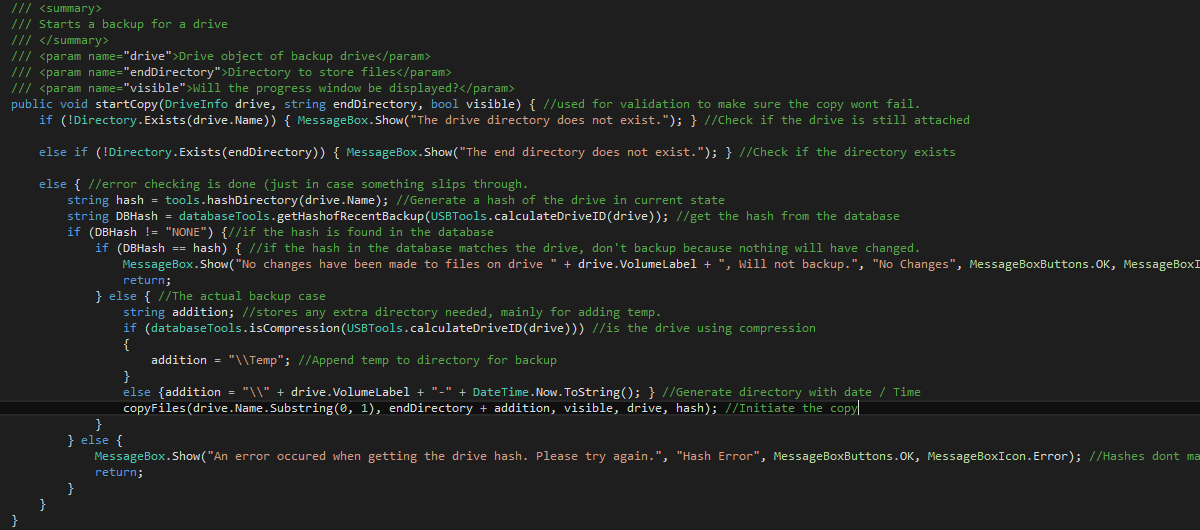
## Eject Drive (mainWindow Line 277)



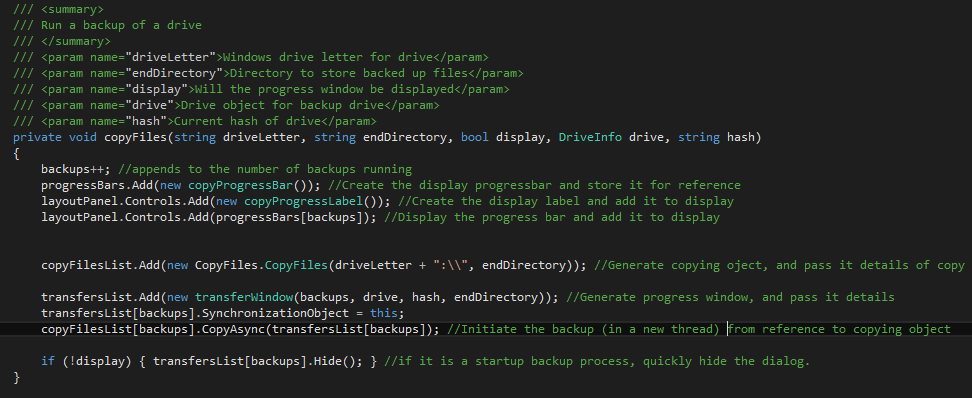
# Important code sections

## Backup Code

### Validation - currentTransfers.cs



### Execution – currentTransfers.cs



## Adding Backup Record set – addBackupWizard.cs

## 

# Procedure list

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Procedure Name** | **Description** | **Parameters** | **Output Type** | **Output Value** | **File Location** |
| **Compress** | Runs pre-checks on data before compression starts | * String directory * String output file * Driveinfo drive | void |  | Compression.cs |
| **Interface7Zip** | Interfaces with built in 7-zip program to compress using that |  | void |  | Compression.cs |
| **Compression7Zip** | Begins compression using LZMA and 7zip library in program |  | void |  | Compression.cs |
| **CompressFileLZMA** | Compresses the file using LZMA algorithm | * String infile * String outfile | void |  | Compression.cs |
| **DecompressFileLZMA** | Decompresses the file using LZMA algorithm | * String infile * String outfile | void |  | Compression.cs |
| **startCopy** | Checks entered data for validity, then starts backup | * Driveinfo drive * String end directory * Bool visible | void |  | currentTransfers.cs |
| **copyFiles** | Starts copying the files from drive to end directory | * String drive letter * String end directory * Bool display * Driveinfo drive * String hash | void |  | currentTransfers.cs |
| **copyDatabase** | Copies the database from executable to install directory |  | void |  | databaseTools.cs |
| **getDriveName** | Returns the drive name from an ID | * String ID | string | Name of the drive | databaseTools.cs |
| **getBackupDirectory** | Returns the backup directory for a drive | * String ID | string | Directory for backup | databaseTools.cs |
| **getBackupCreationDate** | Returns the creation date for a backup record | * String ID | Int64 | Timestamp for date | databaseTools.cs |
| **getAutomaticBackups** | Returns ID’s for all backups using automatic backup |  | String array | Array of IDs | databaseTools.cs |
| **getBackupName** | Returns the name of the backup | * Driveinfo drive | string | Backup name | databaseTools.cs |
| **createBackupRecord** | Records the information of a backup process | * Driveinfo drive * Int64 startDate * Int64 duration * String hash | void |  | databaseTools.cs |
| **getHashofRecentBackup** | Return the most recent hash for a drive | * String ID | string | Latest hash of backup | databaseTools.cs |
| **isCompression** | Returns if the drive is compressed | * String id | bool | compression | databaseTools.cs |
| **getAllDriveBackups** | Returns information on all drives with records |  | datatable | Table of information | databaseTools.cs |
| **deleteDriveRecord** | Deletes a drive record | * Int64 creationdate | void |  | databaseTools.cs |
| **updateDriveRecord** | Updates a drive record in the database | * String backupName * String backupLocation * Bool automatic * Bool compression * Int previousBackups * Int64 creationDate | void |  | databaseTools.cs |
| **removeDisplay** | Wrapper to hide the window |  | void |  | mainWindow.cs |
| **showDisplay** | Wrapper to show the window |  | void |  | mainWindow.cs |
| **refreshDriveList** | Updates the list of connected drives |  | void |  | mainWindow.cs |
| **displayDriveDetails** | Formats the details of the selected drive and displays them in the window | * Driveinfo drive | void |  | mainWindow.cs |
| **formatDriveCapacity** | Animates the progressbar for the drive capacity, and displayed the text |  | void |  | mainWindow.cs |
| **clearDriveDetails** | Removes all the information from the drive details |  | void |  | mainWindow.cs |
| **startupBackups** | Runs the backups that are set to run on startup |  | void |  | Setup.cs |
| **trim** | Formats a long string to add ‘…’ if its beyond a certain length | * String value * Int maxLength | string | Shortened string with ‘…’ | Tools.cs |
| **hash** | Converts string to MD5 hash | * String input | string | Hash of input string | Tools.cs |
| **getUnixTimeStamp** | Creates a unix time stamp of the current time (of execution) |  | Int64 | Time stamp | Tools.cs |
| **Hashdirectory** | Creates a hash of a directory | * String path | string | Hash of directory | Tools.cs |
| **unixDateTime** | Converts a unix time stamp to a usable object | * Long unixTimeStamp | datetime | Datetime object from timestamp | Tools.cs |
| **driveScanTick** | Updates the list of connected drives | * Object sender * Elpsedeventargs e | void |  | USBTools.cs |
| **getConnectedDrives** | Gets a list of the currently connected drives |  | driveInfo array | List of connected drives | USBTools.cs |
| **getDriveType** | Formats the drive type to something the user will understand | * Driveifo selectedDrive | string | String of type of drive | USBTools.cs |
| **spacePercentage** | Turns the capacity of the drive into a percentage | * Driveinfo drive | Int | Percentage of used space | USBTools.cs |
| **calculateDriveID** | Calculates a unique ID for the drive based off its properties | * Driveinfo drive | string | ID for the drive | USBTools.cs |
| **finaliseCopy** | Runs after backup, locks progressbars, and starts compression |  | void |  | USBTools.cs |
| **update** | Updates copy progressbars for window | * Int32 totalfiles * Int32 copiedfiles * Int64 copiedbytes * Int64 copiedbytes * String currentfilename | void |  | USBTools.cs |

# File List (Red are from external sources)

|  |  |
| --- | --- |
| File Name | File Description |
| addBackupWizard.cs | Window for adding a drive for backup to the system |
| backupViewer.cs | Window for displaying and editing currently added backup records |
| compression.cs | Contains functions for compressing backup output |
| currentTransfers.cs | Contains code for running backups, and displaying progress of all running backups |
| databaseTools.cs | Constains functions for interacting with the database, inserting and pulling data |
| ejectDrive.cs | Contains code for ejecting drive from computer |
| mainWindow.cs | Displays main UI screen, and contains the code to adapt |
| notificationIcon.cs | Contains the code to display an icon in the notification tray, and interact with it |
| OSInfo.cs | Contains code to get detailed information on installed OS |
| Program.cs | Start of program is here – CS generated |
| Settings.cs | Contains global settings – CS generated |
| setup.cs | Contains code to initialise the software, and start automatic backups |
| splashScreen.cs | Window for splash screen. So screen isn’t blank while program initialises |
| tools.cs | Contains global tools for the software to use |
| transferWindow.cs | Window for displaying progress of backup it’s attached to. |
| USBTools.cs | Contains code for interacting with connected devices |

# Used Libraries

In order to make use of other people’s code, to make the software more efficient, and cleaner code, external libraries have been used. Some in the form of DLLs, others as direct code files.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Library Name | Use | Creator | Format | Location |
| DotNetBar | More UI controls, and fancy windows | DevComponents | DLL | Installed on Development PC, sent with installation |
| Eject Drive | Ejects a given drive from the computer |  | C# Source | ejectDrive.cs |
| OS Info | Gets information about the current computer |  | C# Source | OSInfo.cs |
| 7-Zip Binaries | Compression / Decompression from 7-Zip format | 7-Zip | DLL | 7-Zip SDK |

# Program Screens (See designs for more)

