Secure Source Backup

Users Manual

Version 2.0

September 3, 2014

MarshallSoft Computing, Inc.
Post Office Box 4543
Huntsville AL 35815 USA

Email: info@marshallsoft.com
web: www.marshallsoft.com

TABLE OF CONTENTS

1 Introduction

Cloud ready source code backup - secure, private, and dependable!

The MarshallSoft "**Secure Source Backup**" (**SSB**) is software designed to create encrypted backups of source code and other critical files. In particular, **SSB** is optimized for uploading to cloud services such as Dropbox and OneDrive (SkyDrive).

The primary design criteria for **SSB** are:

1. **Strong Encryption**: **SSB** uses 256-bit Advanced Encryption Standard (AES).

AES is the standard encryption algorithm used by financial institutions around the world to secure customer data.

2. **File Privacy**: **SSB** can disguise the names of backed up files and directories.

Unlike most backup programs, **SSB** can disguise filenames and directory names. This is an essential feature for files backed up to cloud storage.

3. **File Organization**: **SSB** maintains the same directory organization as your source code.

This is particularly important for files backed up to cloud storage since individual files can easily be recovered without having to download a file archive.

In addition to the above, **SSB** comes with a "Script Assembler" that can be used to make more complex backups, as described sa.pdf .

4. **Independent Encryption:** The user has total control of the encryption of files and the pass phrase.

This is very important for files backed up to cloud storage since you do not rely on the cloud company's encryption policies. The **SSB** software that encrypts your files does not have access to the cloud space where your files are stored, and the software responsible for uploading your files to the cloud only has access to the encrypted version of your files.

See http://www.SecureSourceBackup.com for downloads and more information.

1.1 Features

Some of the many features of the **Secure Source Backup** application are as follows:

- Copies only files that have been changed since the last backup.
- Can specify files to copy or to skip using wildcards.
- Can ignore backing up entire directories such as OBJ and BIN directories.
- Maintains file attributes and file time stamps for all files.
- Can back up and restore hidden and read-only files.
- Can back up and restore files in multi-level sub-directories.
- Can back up and restore files in local storage devices such as internal & external drives.
- Can back up and restore files across local area networks.
- Can compress files to reduce storage space required.
- Can encrypt files using the 256-bit "Advanced Encryption Standard" (AES).
- User maintains complete control of the pass phrase.
- Can disguise file & directory names independently of file content encryption.
- Can be run manually or automatically on a fixed time schedule.
- Maintains the same directory organization so that individual files can easily be restored.
- Uses an authorization file for decryption in addition to the password phrase file.
- The password phrase need be entered just once when creating the password phrase file.
- Encryption is unique for each customer even using the same pass phrase.
- Can backup to multiple destinations during a single backup run.
- Can verify backup volume names and volume serial numbers.
- Programmable using the Script Assembler.

1.2 Script Assembler

Secure Source Backup also includes a "script assembler" for those wishing to perform more sophisticated backups.

The Script Assembler allows the user to check for the existence of specified files and/or directories, check the volume label & serial numbers of drives, check the month and/or day of the week, and then perform conditional branching.

Refer to the "Script Assembler Manual" **ScriptAssembler.pdf** in the \ssb\doc directory.

Example script programs can be found in the \SSB\PGM\ sub-directory.

1.3 Downloading SSB

The latest version of **SSB** can always be downloaded from the **SSB** download page http://www.marshallsoft.com/_ssb.htm or the **SSB** product page http://www.marshallsoft.com/ssb.htm

1.4 SSB API

Using SSB32API.DLL, **SSB** can be started, monitored, and terminated from your application program written in C/C++, Delphi, Visual Basic, Power Basic, Visual FoxPro, dBase, Xbase, COBOL, and Fortran.

The API is currently (September 2014) in beta test. Email us at info@marshallsoft.com with subject "SSB API" if you would like more information or would like to test the DLL. Be sure to mention what particular computer language you use.

2 Installation

After downloading & unzipping **SSB**, and running SETUP.EXE, the following directory structure is created:

\SSB	- Contains SSB.EXE, RunSSB.EXE, & command file Verify.t.
\SSB\DOC	- Contains all documentation files.
\SSB\BIN	- Contains User.bin, Pass.bin, and Auth.bin.
\SSB\EXE	- Contains all executables except ssb.exe & RunSSB.exe.
\SSB\CMD	- Contains several backup & restore command files (*.t).
\SSB\PGM	- Contains Script Assembler and several script assembler programs (*.a).

2.1 Distributed Files

2.1.1 Distributed Binary Data Files

The following files are created by **SSB**. The User.bin and Auth.bin files are unique to each customer and cannot be changed.

User.bin:	Contains user's ID & registration string, and authorized version.
Pass.bin:	The default pass phrase file. Replace this file by running MakePass.exe.
Auth.bin:	Contains user's ID. This is required for decryption, but not encryption.

The user’s ID 100 is for the evaluation version. A unique ID will be assigned to each purchased copy.

2.1.2 Distributed Text Data Files

*.t:	Backup and restore command file examples (found in directory \SSB\CMD).
*.a:	Backup and restore command script examples. (found in directory \SSB\PGM).

2.1.3 Distributed Executables

ssb.exe:	Console mode program that performs the actual backup and/or restore.
MakePass.exe:	Windows program that creates the password phrase file Pass.bin.
RunSSB.exe:	Windows task scheduler used to run ssb.exe once per day.
AskMe.exe:	Windows GUI program that communicates with ssb.exe and allows the user to enter their pass phrase manually (rather than storing in the Pass.bin file).
sa.exe:	Console mode program that assembles script files (*.a) into an executable scripts (*.b).

2.1.4 User Created Files

Pass.bin:	Password phrase file (created by MakePass.exe).
*.b:	Binary command files created from *.a files by the Script Assembler. (Refer to the "Script Assembler Manual" ScriptAssembler.pdf).

2.2 Purchase

SSB may be evaluated for free for 30 days.

The introductory price for **SSB** is $37 USD, which includes 30 days of technical support. One copy of SSB may be used on up to 3 computers. Email us (with subject "SSB HELP") for a multi-copy discount.

This price is good through the end of 2014.

SSB can be ordered online from our web site order page http://www.marshallsoft.com/order.htm

We accept American Express, VISA, MasterCard, Discover, PayPal and International Postal Money Orders (such as Western Union).

Print the file INVOICE.TXT if a "Pro Forma" invoice is needed.

Academic Discount

We offer an "academic price" of 40% to faculty and full time students currently enrolled in any accredited high school, college, or university.

To qualify for the discount, your school must have a web site and you must have a current email address (not forwarded) at your school. Email us (with subject "Academic Discount") to get the academic discount coupon code.

2.3 Updates

SSB is identified by its version number X.Y.Z where:

X = Major version number.
Y = Minor version number.
Z = Maintenance version number.

Maintenance updates are always free. Major updates will be released every 3 to 12 months, and will be offered to existing customers at a reduced price.

However, **SSB** never expires. Purchasing an update is never required.

2.4 Technical Support

Technical support is provided by email and is available (without charge) for registered users for 30 days after purchase. To request support, email us at info@marshallsoft.com with subject "SSB support xxxxx" where xxxxx is your customer ID.

Installation support is also available for the evaluation version. To request support, email us at info@marshallsoft.com with subject "SSB support".

Additional technical support can be purchased for $33 per incident at http://www.marshallsoft.com/order.htm, which also includes an update to the current version.

3 Running SSB

SSB is a Windows 32-bit console mode program that can be run in one of four ways.

3.1 Starting SSB from the Command Line.

To start **SSB** from the Windows command line, type

```
ssb backup.t
```

The text file backup.t is created by the user using any text editor and contains the commands that direct the backup or restore operation. See Section 4.2 “Example Command Files”.

3.2 Indirect Command Files

Several command files can be listed in an "indirect" file. For example,

```
ssb @list.t
```

where list.t contains the list of command files (one command file name per line).

```
command1.t
command2.t
command3.t
```

That is, the above example (ssb @list.t) is equivalent to running **SSB** three times.

3.3 Starting SSB from the Keyboard

SSB can be started from the keyboard by holding down the Windows logo key then pressing the R key, then entering "\ssb\ssb.exe backup.t", where backup.t contains the backup text commands.

3.4 Starting SSB from the Task Scheduler

SSB can be started by the **RunSSB** task scheduler.

Start RunSSB then click "Configure" to enter the **SSB** run command and time of day to run (each day), then click the "OK" button. Use the "Tray" button to move the RunSSB icon to the program tray.

3.5 Automatically Starting RunSSB

RunSSB can be started automatically by placing it in the Windows Startup directory.

Create a shortcut to RunSSB by right-clicking on your desktop and clicking "New" then "Shortcut". For example, enter

c:\ssb\RunSSB.exe

in the box displayed. Name it "RunSSB".

<u>For Windows XP</u>

1. Double-click "My Computer".
2. Double-click the Local Drive (C:) icon.
   (if you see "these files and folders are hidden", click "view these now")
3. Scroll down to the "Documents and Settings" folder.
4. Double-click it.
5. Double-click "All Users"
6. Double-click "Start Menu"
8. Double-lick "Programs"
8. Double-lick "Startup"
9. Drag the RunSSB shortcut into "Startup" window.

<u>For Windows Vista</u>

1. Right-click "Start" button
2. Left-click "Explore All Users"
3. Double-click "Programs"
4. Scroll down until you see the folder labeled "Startup"
5. Double-click it to open its contents window
6. Drag the RunSSB shortcut into "Startup" window.

<u>For Windows 7 & 8</u>

1. Left-click "Start" button
2. Right-click "All Programs"
3. Double-click “All Users”
4. Double-click "Programs"
5. Scroll down until you see the folder labeled "Startup"
6. Double-click it to open its contents window
7. Drag the RunSSB shortcut into "Startup" window.

RunSSB will now be started each time your computer is booted.

4 SSB Run Modes

There are four modes that SSB can be run in: BACKUP mode, RESTORE mode, LIST mode, and VERIFY mode.

4.1 BACKUP Mode

Backup can be performed (1) without encryption, and (2) with encryption.

4.1.1 BACKUP without Encryption

This is a straight forward backup of selected files that have changed since the previous backup. Compression and filename disguising can not be enabled in this backup mode.

The command to backup without encryption is "RUN BACKUP", without a preceding "ENCRYPT Y" command.

4.1.2 BACKUP with Encryption

All selected files that have changed since the previous   copied after encrypting their contents with 256-bit "Advanced Encryption Standard" (AES). Optionally, unlike doing a backup without encryption, files may also be compressed (using the COMPRESS command), and file names may be disguised (using the DISGUISE command).

When SSB encrypts a file, it prepends "0." to the filename of the encrypted file if it has not been disguised, and "1." if the filename has been disguised.

Optionally, SSB can also disguise directory names if the "directory disguise character" has been specified. The directory disguise character (DDC) is set using

```
SET_DDC $
```

where the $ character is prepended to each disguised directory name. Rather than using $ as the DDC, # or @ may be used instead, as for example "SET_DDC @".

The command to backup with encryption is "RUN BACKUP", with a preceding "ENCRYPT Y" command. If compression is also desired, there should also be a preceding "COMPRESS Y" command. If file name disguising is desired, there should be a preceding "DISGUISE Y" command. For example:

```
ENCRYPT Y
COMPRESS Y
DISGUISE Y
RUN BACKUP
```

4.2 RESTORE Mode

Restoring means decrypting previously encrypted files and undisguising previously disguised filenames. Only filenames beginning with "0." and "1." can be decrypted. If the SET_DDC command was previously issued, directories beginning with the specified DDC (directory disguise character) will be undisguised.

The command to restore encrypted files is "RUN RESTORE".

4.3 LIST Mode

The purpose of LIST mode is to generate a map of undisguised files names and their undisguised equivalent. Decryption and file copying is not performed.

The command to list disguised filenames is "RUN LIST".

4.4 VERIFY Mode

The purpose of VERIFY mode is to verify the integrity of the USER file. This command should be used after receiving a new USER file when a registered version of SSB is purchased. See the Verify.t command file.

5 Example Command Files

5.1 Verify Example

Once installed, the first command file to run is "Verify.t".

The verify.t command file verifies that **SSB** can find the files (User.bin, Pass.bin, and Auth.bin) necessary to perform a backup or restore.

This example can be found at \SSB\CMD\Verify.t

5.2 Backup Example

The Backup_SSB.t command file directs **SSB** to perform two backups:

(1) Back up files, without encrypting, from the source directory c:\SSB to the target directory c:\SSB(CopiedBackup).

(2) Back up files, with compression, encryption, and filename disguising, from the source directory c:\SSB to the target directory c:\SSB(EncryptedBackup).

The BACKUP_SSB.t command file copies only newer files if the same files exists in both the source directory and the equivalent target directory.

This example can be found at \SSB\CMD\Backup_SSB.t

5.3 List Example

The List_SSB.t command file directs **SSB** to list the disguised filenames in c:\SSB(EncryptedBackup) along with the equivalent undisguised filenames.

Note that no files are copied.

This example file can be found at \SSB\CMD\List_SSB.t

5.4 Restore Example

The Restore_SSB.t command file directs **SSB** to restore the backed-up files from the source directory c:\SSB(EncryptedBackup) to the target directory c:\SSB(Restored).

Backed-up filenames will always begin with either with "0." (indicating an encrypted file) or "1." (indicating an encrypted file whose filename is disguised). The Restore_SSB.t command does <u>not</u> copy files that are not encrypted.

This example file can be found at \SSB\CMD\Restore_SSB.t

6 Testing SSB

The best way to test **SSB** is to edit one of the example command files.

After installation, the recommended course of action is:

1. Read the documentation file SecureSourceBackup.pdf (this file).
2. Look through the example command files in the \SSB\CMD directory.
3. Choose one of the example command files to edit.
4. Edit the SOURCE command with the location of files to be backed up.
5. Edit the TARGET command with the location of where files are to be backed up to.
6. Create a Pass.bin file using MakePass.exe or use the default Pass.bin file.
7. Run a backup: ssb backup.t
8. Edit Restore.t to correspond with the information in Backup.t, specifying the SOURCE & TARGET directories,
9. Run a restore: ssb Restore.t
10. Verify that the files restored are identical to the original files.

When restoring, be careful not to restore more than you actually want. It is recommended that you always restore to a different directory than the source.

It is very important to verify that you know how to restore files before the need actually occurs.

7 Files Used by SSB

7.1 USER.BIN File

The USER.BIN file contains the purchaser's user name and ID number, and must be read by **SSB** before any backup or restore operation can be performed.

The USER.BIN file is specified in all of the example command files (*.t).

7.2 PASS.BIN File

The PASS.BIN file contains the current pass phrase used for encryption and decryption, and must be read by **SSB** before any backup or restore operation can be performed.

The PASS.BIN file is specified in all of the example command files (*.t).

Rather than specifying a pass phrase in the PASS.BIN file, the pass phrase can be entered manually at runtime. For example, replace the PASS command in the command file with

```
  PASS c:\ssb\exe\$
```

where the AskMe.exe executable is in the directory c:\ssb\exe\

7.3 AUTH.BIN File

The AUTH.BIN file contains the purchaser's ID number that must match the ID number in the USER.BIN file. The AUTH.BIN file is required when decrypting previously encrypted files (RESTORE operation) but not for encrypting files (BACKUP operation).

The purpose of the authorization file AUTH.BIN is to prevent someone else, who even has a copy of your Pass.bin file, from decrypting files.

8 Important SSB Capabilities

In addition to being able to copy files, **SSB** can also compress & encrypt the contents of files, disguise file names, and disguise directory names.

7.1 File Compression

SSB can compress files using a variation of the FLZ compression algorithm in order to reduce the space needed to store backed-up files. Text files, such as source code, are typically compressed to between 30% and 40% of their original size.

7.2 File Encryption

SSB can encrypt files using the (256-bit) "Advanced Encryption Standard" (AES).

AES was developed under the auspices of the U.S. National Institute of Standards and Technology (NIST). See fips-198.pdf (http://csrc.nist.gov/publications/fips/fips197/fips-198.pdf)

AES is considered "strong encryption" and replaces the previous US encryption standard "Data Encryption Standard" (DES). AES is commonly used by many financial entities such as banks to protect their customer's sensitive information.

Our implementation of the Advanced Encryption Standard has been verified by running the "Advanced Encryption Standard Algorithm Validation Suite" (AESAVS), which can be found at Advanced Encryption Standard Algorithm Validation Suite (http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf).

7.3 Filename Disguising

SSB can disguise file names and directory names, which is particularly important when backing up to cloud storage. Name disguising is designed to make it difficult for someone else to determine the actual name of a file or directory.

During backup the '$' character is appended to directory names to indicate that they are "disguised". For example, directory "ABC" might be disguised as "$JLN".

Only encrypted file names can be disguised. An encrypted filename will begin with extension "0." whereas an encrypted filename whose name is disguised will end with extension "1.".

7.4 Encryption Key

When the user enters a password or password phrase, it is processed by the SHA-256 hash algorithm to produce a 256-bit encryption key. SHA-256 was designed by the U. S. National Security Agency (NSA) and published in 2001 by the NIST as a U.S. Federal Information Processing Standard (FIPS).

The greatest security hazard is safeguarding the password phrase. **SSB** supports this by using a decryption authorization file (Auth.bin) that is unique to each customer and which can be kept offline for added security until needed for restoring encrypted files.

9.0 Usage Notes

9.1 Multiple File Sources

Files from directories without a common root can be processed by specifying multiple SOURCE and TARGET directories, as for example:

```
 # backup MySourceFiles to MySourceBackup
SOURCE MySourceFiles
TARGET MySourceBackup
RUN BACKUP
 # backup YourSourceFiles to YourSourceBackup
SOURCE YourSourceFiles
TARGET YourSourceBackup
RUN BACKUP
```

9.2 Directory Processing

Windows also marks certain directories are system, junction, hidden, or read-only. These directories are used by the Windows operating system to help organize different users, among other system uses.

When following source or target directory trees, **SSB** ignores system, junction, hidden, and read-only directories. Only normal sub-directories are processed.

9.3 Time Stamps

SSB uses the Windows file "last write" time stamp to determine when a file was last modified. However, the write time resolution is different for different Windows file systems. For example, on the FAT file system, the write time has a resolution of only 2 seconds.

The "TIMESTAMP" command is used to specify the maximum time difference between two files for them to be considered as identical files. The argument for the TIMESTAMP command is the number of seconds (0 to 120) or the letter 'N' which disables timestamp testing.

The default value of the TIMESTAMP is 1 second.

10 Caveats

10.1 Forgetting the Pass Phrase

Files can only be decrypted using the same pass phrase that was used to encrypt them. That is, don't forget your pass phrase!

10.2 Changing Pass Phrases

If the pass phrase is changed, then all files that were previously encrypted cannot be decrypted using the new pass phrase. The best solutions are to (1) delete all backed up files before the next backup run, or to (2) add "TIMESTAMP N" to the backup command file in order to ignore timestamps when the backup is run, thus forcing SSB to backup all selected files. After the backup run, remove the TIMESTAMP command so that subsequent backups can use file timestamps.

10.3 Using "older" Files

If you copy (or unzip, etc) an older version of a file into the source file set (set of files that will be backed up) and there is already a backed version of this file in the target file set, then the backed up version will have a more recent time stamp and therefore will not be backed up during the next backup run. The best solutions are to (1) "touch" (or edit) the file so that it has a current timestamp, (2) delete the existing backed up version of this file (in the target file set) before the next back-up run, or (3) add "TIMESTAMP N" to the backup command file in order to ignore timestamps when the backup is run. After the backup run, remove the TIMESTAMP command so that subsequent backups use the timestamp.

11 Cloud Storage

11.1 Why Use Cloud Storage

It's a "no brainer" that source code and other critical files should be backed up on a regular basis. Storing source code locally isn't much help if the local storage devices are destroyed in a fire or similar disaster. Storing your source code in the cloud provides an obvious solution.

The only reason not to upload your source code or other critical files to cloud storage (such as Dropbox and OneDrive) is the concern about security and privacy. The only sure way to make sure that your files cannot be copied is to encrypt them independently of the cloud storage software. This ensures that your files are encrypted before being uploaded but more importantly that you alone control access to your pass phrase.

This security requirement is achieved in **SSB** by encrypting files with 256-bit "Advanced Encryption Standard". The privacy requirement is achieved by optionally also disguising both filenames and directory names.

11.2 Sync Folders

"Cloud Storage" simply refers to offsite file storage accessed via the Internet. This could be storage at the Dropbox or OneDrive (SkyDrive) storage facility or a friend's computer that is remote to you.

There are several interface methods used by cloud storage companies. The simplest method is to use "sync folders". This is the method used by Dropbox, OneDrive (SkyDrive), ZipCloud, and iDriveSync among others.

A "sync folder" is a directory created on one of your drives such that when anything is written to the sync folder, it is automatically uploaded to the cloud server by an application program that actively monitors the sync folder. It makes no difference whether you drag a file to the sync folder or if a program copies a file there.

If a new computer is setup to use an existing account, then all existing files will be downloaded to the new computer. Several computers can have sync folders. If a file is deleted from one, it is deleted from the others. That is, the cloud monitoring program (such as Dropbox) actively enforces file synchronization across all computers.

11.3 Using Dropbox

Dropbox provides a nice example of how sync folders work.

1. Locate your DROPBOX directory on your computer. On some machines (such as XP) it will be in the "C:\Documents and Settings\Mike\My Documents" directory, where "Mike" is replaced by your computer's name. On other machines (such as Visa, Windows 7, & Windows 8) it will be in the "C:\Users\Mike\Dropbox" directory, where "Mike" is replaced by your computer's name.

2. Create the directory MyBackup (or whatever name you like) in your Dropbox directory.

3. Edit the file Backup.t. Replace the line

```
TARGET c:\Backup
```

With the location of your DropBox "MyBackup" directory. For example,

```
TARGET c:\Users\Mike\Dropbox\MyBackup
```

4. The remainder of the command file is the same whether using Dropbox or not. See Section 4 “Example Command Files”.

12 Best Security Practice

1. Use long pass phrases that can be easily remembered. For example,

"George Washington was the first President"
"My first girlfriend was Brigitte Bardot"

2. If you "must" keep a text copy of your pass phrase, put it on a USB flash drive stick then remove it from your computer. Alternatively, write it down in a favorite book on an easily remembered page.

3. Move the restore authorization file Auth.bin to an external device such as a USB flash drive stick then remove both Auth.bin and the stick drive from your computer. Without Auth.bin encrypted files cannot be decrypted even if the pass phrase is known.

4. Change your pass phrase on a regular basis. Of course, a new pass phrase cannot decrypt files that were encrypted using the previous pass phrase.

5. Backup daily, preferably to a local USB drive and also to an offsite cloud service. Note that multiple backups can be performed in a single command file.

6. Consider doing distinct backups for each day of the week. Thus you will always have a backup for each day of the last week. This is most easily accomplished using a SSB script.

7. Archive backups on a regular schedule. For example, archive a backup each month.

8. Be sure to test restoring files so that you will be confident that files can be restored when needed. Remember that you need the Auth.bin file when restoring files.

13 Text Commands

The "Text Command" file is a plain ASCII text file that contains all commands that guide **SSB** during backup or restore. Text files used by **SSB** have extension ".t".

Each line in a command file may have no more than 256 characters. Lines beginning with the '#' character are comments. All other lines are formatted as

LeftString RightString

as can be seen in the example command files.

<u>Command Instructions</u> (* = script assembler <u>only</u>).

	AUTH	: Specifies the path name of the restore authorization file.
	BEEP	: Makes a beep sound.
*	CALL	: Calls a block of code (subroutine).
*	CANCEL?	: Prompts the user to cancel execution or not.
	COMPRESS	: Enables compression.
	COPY	: Specifies a file to copy.
	DEBUG	: Sets the SSB debug level.
	DIR?	: Tests for the existence of the specified directory.
	DISGUISE	: Enables file name disguising.
	DISPLAY	: Displays a user specified message.
*	DOW?	: Checks the day of the week.
*	DRIVE?	: Tests for the existence of the specified drive.
	ENCRYPT	: Enables encryption.
*	FILE?	: Tests for the existence of the specified file.
*	GOTO	: Transfers control to the specified program address (label).
	HIDDEN	: Enables the processing of hidden files.
*	IF_FALSE	: Transfers control if the condition code is "true".
*	IF_TRUE	: Transfers control if the condition code is "false".
	IGNORE	: Ignores (does not process) specified directory.
	LOG_FILE	: Specifies the name of the SSB log file.
	MAP_FILE	: Specifies the name of the SSB map file.
*	MONTH?	: Checks the month of the year.
	MOUNT	: Request mounting the specified drive.
*	NOP	: Performs no function.
	PASS	: Specifies the name of the SSB password phrase file.
	READONLY	: Enables processing of "read only" files.
	RESET	: Resets the script program (other than USER command).
*	RETURN	: Jumps to the instruction immediately following the last CALL.
	RUN	: Starts a BACKUP, RESTORE, or LIST operation.
	SKIP	: Specifies a file to skip (not copy).
	SLEEP_MIN	: Sleeps the specified number of minutes.
	SLEEP_SEC	: Sleeps specified number of seconds.
	SET_DDC	: Sets the "directory disguise character".
	SOURCE	: Specifies the name of the SSB source directory.
*	STOP	: Terminates execution.
	SUB_DIRS	: Enables the processing of sub-directories.
	TARGET	: Specifies the name of the SSB target directory.
	TIMESTAMP	: Sets the max time between files to be considered the same.
	USER	: Specifies the name of the SSB user ID file.
	VERBOSE	: Enables verbose output in the log file.
*	VOL_NAME?	: Tests for the specified volume name.
*	VOL_SERIAL?	: Tests for the specified volume serial number.
	WRITE	: Writes text to the SSB log file.

See *Appendix A* of the <u>*Script Assembler Manual*</u> for a detailed description of each command.

14.0 Frequently Asked Questions

Q: I upload to the cloud where my files are encrypted, so what's the advantage of using SSB to encrypt my files?

The cloud storage company would have a copy of your encrypted files and also know how files are encrypted and how the password or pass phrase is managed. Furthermore, there is always the possibility of a "back door" or error in their code which would allow them to decrypt your files. The best solution is to encrypt your files independently of where they are stored.

Q: I already backup to an eternal hard drive, so why back up to the cloud?

Backing up to an external USB drive is almost always a good idea, but if your computer hard drive crashes while updating the external USB drive, you could lose files on the crashed computer drive as well as the USB backup drive.

The other consideration is that if there is a fire or similar disaster, all of your files could be lost. This is the primary reason to back up offsite.

Q: Banks have had security breaches although they use AES to encrypt files, so can't AES be broken?

Security breaches at banks, credit card processing companies, and similar institutions do not involve breaking AES. AES is used world wide to safeguard important files by a wide variety of institutions, including the entire banking community. If a way is found to break AES, there would no doubt be a world wide scramble by many thousands of institutions to replace AES with something else, which would be impossible to keep secret.

Q: I use software that backs up every file that I have to the cloud, so why use SSB?

This can be a good solution, although it is very difficult to verify that all of your important files have been properly uploaded because of the very large volume of files. It also takes a long time to upload (which is much slower than downloading) so many files to the cloud.

SSB uploads only source code and other important files that you have chosen which is a small portion of the files on your computer. This means that there is sufficient free space provided at places such as Dropbox and OneDrive.

15 Legal Issues

15.1 License

SSB is licensed per user, not per computer.

This license agreement (LICENSE) is a legal agreement between you (either an individual or a single entity) and MarshallSoft Computing, Inc. for this software product (SOFTWARE). This agreement also governs any later releases or updates of the SOFTWARE. By installing and using the SOFTWARE, you agree to be bound by the terms of this LICENSE. If you do not agree to the terms of this LICENSE, do not install or use the SOFTWARE.

15.2 Warranty

MARSHALLSOFT COMPUTING, INC. DISCLAIMS ALL WARRANTIES RELATING TO THIS SOFTWARE, WHETHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND ALL SUCH WARRANTIES ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. NEITHER MARSHALLSOFT COMPUTING, INC. NOR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DELIVERY OF THIS SOFTWARE SHALL BE LIABLE FOR ANY INDIRECT, CONSEQUENTIAL, OR INCIDENTAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH SOFTWARE EVEN IF MARSHALLSOFT COMPUTING, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR CLAIMS. IN NO EVENT SHALL MARSHALLSOFT COMPUTING, INC.'S LIABILITY FOR ANY SUCH DAMAGES EVER EXCEED THE PRICE PAID FOR THE LICENSE TO USE THE SOFTWARE, REGARDLESS OF THE FORM OF THE CLAIM. THE PERSON USING THE SOFTWARE BEARS ALL RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE.

Some states do not allow the exclusion of the limit of liability for consequential or incidental damages, so the above limitation may not apply to you.

This agreement shall be governed by the laws of the State of Alabama and shall inure to the benefit of MarshallSoft Computing, Inc. and any successors, administrators, heirs and assigns. Any action or proceeding brought by either party against the other arising out of or related to this agreement shall be brought only in a STATE or FEDERAL COURT of competent jurisdiction.

16 Version History

Version 1.0: Jan 28, 2014

Alpha testing

Version 1.1: May 1, 2014:

Beta testing

Version 1.2: July 28, 2014

First public release.

Version 2.0: September 3, 2014

Additional customer information written to User.Bin file.
SSB can be aborted from RunSSB program.