SEAL

SquishMail Relay Manager

& File Processor

Version 0.50

21 JUL 96

Table of Contents

NO WARRANTY

THIS SOFTWARE AND MANUAL ARE PROVIDED "AS IS" AND WITHOUT WARRANTIES AS TO PERFORMANCE OF MERCHANTABILITY OR ANY OTHER WARRANTIES WHETHER EXPRESSED OR IMPLIED. BECAUSE OF THE VARIOUS HARDWARE AND SOFTWARE ENVIRONMENTS INTO WHICH THIS PROGRAM MAY BE PUT, NO WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE IS OFFERED. GOOD DATA PROCESSING PROCEDURE DICTATES THAT ANY PROGRAM BE THOROUGHLY TESTED WITH NON-CRITICAL DATA BEFORE RELYING ON IT. THE USER MUST ASSUME THE ENTIRE RISK OF USING THE PROGRAM. ANY LIABILITY OF THE PROVIDER WILL BE LIMITED EXCLUSIVELY TO PRODUCT REPLACEMENT.

LICENCE

Seal is protected by Canadian copyright laws and international treaty provisions.

Seal and SealCfg are trademarked by Robert Presland.

Seal is provided `as is' and without warranty of any kind.

Seal is provided at no cost. No cost shall be imposed for the distribution of Seal beyond the cost of transfer media or phone service as a method of transfer.

Seal cannot be bundled with any other software package, for any purpose, without prior written consent from the author.

You are licenced to use Seal at no cost in a non-commercial environment. A non-commercial environment is a location that does not meet the requirements of a commercial environment.

You may not operate Seal in a commercial environment without explicit written permission from Robert Presland. A commercial environment is one which stands to make a profit of any kind. Any registered business or government office (Canadian or otherwise) is deemed a commercial environment for the purposes of this licence.

Permission to operate Seal in a commercial environment can be obtained by writing to the author.

INTRODUCTION

What is Seal

Seal is an Areafix and Raid clone that works with Squish's configuration file as well as Tick's configuration file.

For those of you who are not familar with Areafix, it is a program that automates the linking and unlinking of your downlinks to the echomail areas you carry and optionally requests areas from your uplink for your downlinks, all without you having to lift a finger.

Raid is a similar program that handles the processing of files into your system in the same way Areafix handles echomail.

Why use Seal

No reason. Use it if you want, unless you want to do all the linking and unlinking yourself. Seal does not have many advantages over other software that reports to do the same thing, nor does it have many disadvantages (that I can see) from other software so it's just a matter of choice.

Why I wrote Seal

For a long time I had used Areafix to do my echo area management but it lacked some of the new features that Squish afforded me. This in combination with not being able to find another reliable and satisfactory piece of software finally drove me to say "I can do better than that!". So here it is. I use it, so why not you?

The Archive

Hopefully you will have received Seal from a reputable place such that it does not fail CRC checks or the like. Inside the archive you should have the following files.

SEAL.EXE	Real mode executable.
SEAL386.EXE	Protected mode executable.
SEALCFG.EXE	Configuration utility.
SEAL.PRN	Documentation file.
LANGUAGE.EXE	Archive of language files.
IMPAREAS.EXE	Import AREAS.BBS into SQUISH.CFG.
INSTALL.EXE	Installation/Upgrade Utility.
SEALINI.DAT	Used by INSTALL.EXE.
SETUP.INI	INI file for SealCfg Utility.
DPMI.ARJ	Protected mode drivers for SEAL386.EXE.
WHATSNEW.050	List of new features since version 0.40.

If you are missing any of these files, throw the rest away too, then f'req SEAL from 1:243/27.

INSTALLATION

Conventions

No, this is not the section on international meetings for sysops. There are several concepts used throughout Seal which you should be aware of. They are as follows.

The first address listed in the EchoArea line of the SQUISH.CFG file (ignoring any -p switch addresses) is considered to be the uplink for that area. If that address is your address, then you originate the area. Seal will never change the feed for an area even though it does sort the rest of the node addresses on the line. For example:

```
EchoArea POINTS \MSG\POINTS -0 -$ 1:23/67 1:23/12 23.1 34
```

The feed for POINTS is 1:23/67.0 and the other nodes (1:23/12.0, 1:23/23.1, and 1:23/34.0) are sorted, which brings me to my next point.

Anywhere where you specify a node address, you need not specify a zone or a point or a net. The zone is assumed to be the same as the zone of your primary address -- the first address listed in the SQUISH.CFG file, unless explicitly specified otherwise. Points are assumed to be zero unless otherwise specified. For example, check the EchoArea line above.

Although Seal supports areas listed in either the SQUISH.CFG or the AREAS.BBS configuration files, Seal does not support split area definitions (i.e. areas listed in both files). You should list your areas in *either* the SQUISH.CFG file *or* the AREAS.BBS file. If you must list areas in both files due to the operation of another utility which only supports AREAS.BBS let me know as I may be able to add to Seal.

Translation Characters

In several places as outlined elsewhere in this document, Seal utilizes translation characters. This allows you to customize the look of Seal on your system.

Translation characters not only allow you to substitute information, but allow you to format your configuration files.

The syntax for a translation character is as follows.

```
%[[-]len][:mlen]<char>
```

A standard translation character could be %p or %a, while a formatted translation character could be %-20p or %20:20a.

[len] is the length to make the translation character after substitution. So if you specify %20p, then the character will be substituted and then right padded to 20 characters. If the substituted value is more than 20 characters already, then no

padding is added and no truncating take place.

If you specify the optional dash preceding the [len] then the value if padded on the left after substitution.

[:mlen] specifies how much of the original value to use before padding (if specified). So if you use %:40p then only the first 40 characters of the value will be substituted.

Both the len and :mlen are optional as well as the -. If you omit the len then no padding takes place. If you omit the :mlen then no truncating is done before substitution. The following table illustrates the concept. The original value is MUFFIN and the basic translation character is %t, with spaces represented by a period.

Translation sequence	Replaced with
%t	MUFFIN
%10t	MUFFIN....
%-10t	....MUFFIN
%:4t	MUFF
%-5:4t	.MUFF
%10:4t	MUFF......

Installation

1. If you are upgrading from version 0.40, you should backup your Seal directory. You cannot upgrade from any version prior to 0.40 with this archive.

2. Extract the Seal archive to a temporary directory.

3. Run the INSTALL program contained in the archive.

4. INSTALL will prompt you for the directory in which you want to install Seal. If you are upgrading, enter the directory where your Seal insstallation currently resides. Otherwise, you should enter a directory that is not shared with any other application.

5. INSTALL will install all the files that Seal needs to operate. If upgrading, INSTALL will backup your SEAL.INI to SEAL.SAV.

6. If you are upgrading, you are done. If this is a first time installation, you should configure your nodelinks and feeds using SealCfg..

7. Before using Seal 0.50, you should backup your SQUISH.CFG and TIC.CFG files.

8. If you wish to run the protected mode of Seal (SEAL386.EXE) you should extract the contents of the DPMI.ARJ archive and place the files in your path.

Memory Requirments

Seal really doesn't have any special memory requirements, per say. Seal doesn't need and won't use any extended or expanded memory, although it will try to load the overlay file into expanded memory if it can.

The memory requirements really depend on your setup and how many areas you have. Seal will not run unless it can successfully load all your areas into memory.

Seal is also smart with regards to what it loads into memory. For example, it won't load the file areas if you only specify SEAL BAD, since it doesn't need them in this case. Seal still requires the presence of the files, but it may not read everything in the files. This should allow for better memory management as well as speed performance while reading your config files.

There is one option of which you may take advantage with regards to Seal's memory management. If the ANNOUNCE command line parameter is the only reason Seal reads your echomail areas, then you can use the -ne to tell Seal not to read the echomail areas. This results in default announcement messages, but otherwise will work the same as if Seal had read the echomail areas. Refer to the section on command line switches for more details regarding the -ne switch.

THE COMMAND LINE

Command Line Parameters

Add

This option tells Seal to create an echomail or a file area. The syntax is as follows:

SEAL ADD <type> <tag> <node>

<type> is either `FILE'' or ``ECHO''.

<tag> is the tag of the area to be created.

<node> is the node address of the feed for this area.

An example might look like this:

SEAL ADD ECHO MUFFIN 1:234/5

The feed must already be configured within Seal as a valid feed. If the area to be created is an echomail area, Seal will check the feed's configuration to determine what switches to place on the EchoArea line. If the area is to be a file area, then Seal will check the feed's configuration to determine what flags to place on the feed's entry in the newly created area.

Announce

This option tells Seal to check the inbound directory for incoming TIC files, and if found, announce the arrival of these files. You must have a TIC.CFG file configured for this option to function. Please refer to the section called ``Announcing Files'.

Areas

This option tells Seal to create a formatted file listing the available echomail or file area. The syntax for this command is as follows:

SEAL AREAS <type> <lvl> <key(s)> <file>

<type> is either `ECHO'' or ``FILE''.

<lvl> is the access level of the areas to include in the list.

<key(s)> is a single word containing the keys that a nodelink must possess to gain access to any of the areas included in this list. You can specify all keys by using an asterisk.

<file> is the file to create with the areas list.

If you are creating a list of echomail areas, the resulting file will be in AREAS.BBS format. If the list is a list of file areas, then the resulting file will be in FILEBONE.NA format.

Bad

This options tells Seal to scan your bad message area. Seal will check each message found in your bad message area to detrmine if it comes from a valid echomail feed. If a message meets this criteria, and the feed is configured to create new areas, then Seal will create an area for the message.

Seal will not create an area if the area is listed in the delete file.

If the created area is also listed in the queue file, then Seal will add the downlinks found in the queue file to the created area. Seal will also send a message to each downlink added to the new area announcing the new area's creation.

If the feed that the message is from is configured to announce in an echomail area, Seal will then create a packet announcing the creation of the new area.

Clean

This option tells Seal check all areas to see if they are pass-through areas and are not being fed to any downlinks. If the feed for such an area is configured to create new areas, then Seal will unlink the area from the feed and remove the area from

your system.

If the feed for the area is configured to announce in echomail, Seal will create a packet announcing the removal of the area in question.

Note than an area that has downlinks all of whom are configured to not allow a free ride, is considered a dead area, and will be treated as if it had no downlinks.

Describe

This option tells to add descriptions to your SQUISH.CFG or TIC.CFG files. The syntax is as follows:

SEAL DESCRIBE <type> <description_file> [-forced]

<type> is either `FILE'' or ``ECHO''.

<description_file> is a file provided by your feed.

-forced is an optional switch which tells Seal to override existing descriptions.

An example might look like this:

SEAL DESCRIBE ECHO FIDONET.NA

SEAL DESCRIBE FILE FILEBONE.NA -forced

Seal will only add descriptions to areas that don't already have a description, unless the -forced switch is used. Seal expects to find the description file in the standard FIDONET.NA format for echomail areas, and FILEBONE.NA format for file areas.

Drop

This option tells Seal to remove an area from your system. It is the oposite of the Add option. The syntax is as follows:

SEAL DROP <type> <tag>

<type> is either `ECHO'' or ``FILE''.

An example might look like this:

SEAL DROP ECHO MUFFIN

SEAL DROP FILE SDSMAX

Seal will remove the area from your system as well as sending a message to each of the downlinks that were receiving the area.

If the feed for the area is configured to announce in echomail, Seal will create a packet announcing the removal of the area in question.

For echomail areas in *.MSG format, Seal will remove all *.MSG

files and the directory. For echomail areas in Squish format, Seal will delete all message base files for the area in question.

For pass-through file areas, Seal will delete all files found in the area's directory and then remove the directory. If the file area is not a pass-through area, Seal will NOT delete any files -- you must delete them manually.

FileCtl

This option tells Seal to create a Maximus-style FILEAREA.CTL file which lists all non-pass-through file areas. The syntax is as follows:

SEAL FILECTL <level> <lock(s)> <priv> <file>

<level> is the access level of the areas to include. You can specify one access level, or a range in the form of xxx-yyy, +xxx, or -xxx.

<locks(s)> is the lock or locks that an area must have to be included in the list of areas. You can specify all locks by using the asterisk (*).

<priv> is the privilege level (and keys) to use in the FileAccess line of the Maximus area definition.

<file> is the name of the file to create.

An example might be as follows.

SEAL FILECTL 0-50 F Normal/F FIDOFILE.CTL

SEAL FILECTL +100 AB Favoured FIDOAREA.CTL

If ``Local ListName' appears in an area defination that is included in the resulting file, Seal will place a ``FileList'' line in the resulting file.

FileStat

This option tells Seal to create a netmail message addressed to you that details the traffic by number of files and average size for each file area you have configured.

Find

This option tells Seal to find the areas a downlink is linekd to and display them on the screen. The syntax is as follows:

SEAL FIND <addr>

<addr> is the node address of a downlink.

An example might look like this:

SEAL FIND 1:234/56

You must have the downlink already configured.

Format

This option tells Seal to rewrite your SQUISH.CFG and TIC.CFG files. Seal normally does this when it makes changes to your configuration files. You can force a rewite by using this option.

Hatch

This options tells Seal to place a file into distribuution in a file area. The syntax is as follows.

SEAL HATCH <area> <file> [-x<file>] [desc]

<area> is the file area tag in which you want to hatch the file.

<file> is the name of the file you want to hatch. It must be located in the directory specified for the area in TIC.CFG.

[-x<file>] is the name of an existing file to be replaced with the file you are hatching.

[desc] is a description for the file you are hatching. The description muust be the last parametr on the command line.

You can tell Seal to read the file's description from a file instead of typing it at the command line, by replacing [desc] with !<file> where <file> is a text file that contains the description of the file being hatched.

Help

This option tells Seal to send a help message to each of the downlinks you have configured.

Kill

This option tells Seal to delete any message found in your netmail area that originate from a configured uplink.

Link

This option tells Seal to link a downlink to one or more echomail or file areas. The syntax is as follows:

SEAL LINK <addr> <type> <area(s)> [-forced]

<addr> is the node address or a nodelink already configured.

<type> is either `ECHO'' or ``FILE''.

<area(s)> is one or more area tags to which the downlink should be linked.

[-forced] will link the nodelink regardless of security access.

An example might look like this:

SEAL LINK 1:234/56 ECHO MUFFIN -forced

SEAL LINK 1:234/67 FILE SDSMAX PDNPASCL

To specify all areas the downlink has access to, you can replace the areas with an asterisk (*).

This option simulates a request from the downlink, abiding by the security access of that downlink. To override the security access, use the -forced switch.

List

This option tells Seal to create a list of echomail or file areas. The syntax is as follows:

SEAL LIST <type> <level> <lock(s)> <file>

<type> is either `ECHO'' or ``FILE''.

<level> is the access level that a nodelink must have to access the areas listed on the resulting file.

<lock(s)> is the lock or locks that a nodelink must have to access the areas listed in the resulting file. To specify all possible locks, replace the locsk with an asterisk (*).

<file> is the file that is created.

An example might look like this:

SEAL LIST ECHO 5 . AREAS.LST

SEAL LIST FILE 50 F FILEAREA.LST

If you specify `MSG'' as your file to create, Seal will create a netmail message addressed to you instead of creating a file.

MsgCtl

This option tells Seal to create a Maximus-style MSGAREA.CTL file which lists all non-pass-through echomail areas. The syntax is as follows:

SEAL MSGCTL <level> <lock(s)> <priv> <file>

<level> is the access level of the areas to include. You can specify one access level, or a range in the form of xxx-yyy, +xxx, or -xxx.

<locks(s)> is the lock or locks that an area must have to be included in the list of areas. You can specify all locks by using the asterisk (*).

<priv> is the privilege level (and keys) to use in the MsgAccess line of the Maximus area definition.

<file> is the name of the file to create.

An example might be as follows.

```
SEAL MSGCTL 0-50 F Normal/F FIDOMSG.CTL

SEAL MSGCTL +100 AB Favoured FIDOAREA.CTL
```

Seal will add the following to the resulting file if the appropriate swith is present on the EchoArea line in SQUISH.CFG for each area.

```
-s          Public Only
-h          Private Only
-$          Type Squish
-$dxx       Renum Days xx
-$mxx       Renum Max xx
-px:xx/xx   Origin y
```

When Seal finds the -p switch, it will set y to the number of the listed akas that matches the -p directive. So your addresses in MAX.CTL should be the same as in SQUISH.CFG and in the same order for this to function correctly.

NewFeed

This tells Seal to replace one feed with another feed. This option is useful if you change from one uplink to another uplink. When used, Seal will send a message to the old uplink dropping all areas that you receive from that uplink. Seal will then request all those areas from the new uplink. Seal will also go through your configuration file and change all all the associated areas. The syntax is as follows.

```
SEAL NEWFEED <E|F> <old_addr> <new_addr>
```

Use E for echomail feeds, and F for file area feeds.

Note: You must first define the new uplink in SealCfg. If you don't, NewFeed won't work.

Notify

This option will send a notify message to all the nodes you have configured. The notify message can optionally contain a list of areas that the node is linked to, or can contain a list of all the areas available to that node. For example, to send a notify message to every configured node, you would do the following:

```
SEAL NOTIFY
```

Or to send a notify message to one node only:

```
SEAL NOTIFY 1:234/567
```

Or to send a notify message to more than one node:

```
SEAL NOTIFY 1:234/567 2:345/678
```

You may want to put this sort of thing in a monthly batch file to remind your downlinks what areas you have them linked to.

Pause

This option tells Seal to unlink all or some of your downlinks from all their areas but save a list of those areas so that they can be relinked later. You can pause all areas for all your downlinks like so.

SEAL PAUSE

You may also place all areas for one node (say 1:234/56) on pause like so.

SEAL PAUSE 1:234/56

Or you may also place all areas for more than one node on pause like so.

SEAL PAUSE 1:234/56 78

The areas (both file and echo) for each node placed on pause are placed in the pause file for use later by the RESUME command. Unlinking during a pause command is identical to unlinking initiated by the downlink.

Pass

This option tells Seal to toggle the passthru status of either an echomail or file area. The syntax is as follows.

SEAL PASS <ECHO|FILE> <tag>

If you specify an echomail area, Seal will add or delete the `-0' flag. If you specify a file area, Seal will add or delete the `Local Passthru' line from that area's definition block.

Relink

This option tells Seal to produce a message for each of your uplinks that has create new areas enabled, requesting all the areas that you get from that uplink. This is used if your uplink has lost their control files and has asked you to request your areas again. You may use this in either file or echo mode as follows:

SEAL RELINK <type>

For example:

SEAL RELINK ECHO
SEAL RELINK FILE

Resume

This option tells Seal to re-establish all previously paused areas for all or some of your downlinks. You can resume all areas for all your downlinks like so.

```
SEAL RESUME
```

You may also resume all areas for one node (say 1:234/56) like so.

```
SEAL RESUME 1:234/56
```

Or you may also resume all areas for more than one node like so.

```
SEAL RESUME 1:234/56 78
```

The areas (both file and echo) for each node previously placed on pause are read from the pause file and are re-established identical to a link command initiated by the downlink.

Scan

This option tells Seal to scan your netmail and respond to any messages to it. This is commonly called auto-response mode.

SetNode

Use this option to set the access level and keys for a downlink. The syntax is as follows.

```
SEAL SETNODE <node> <level> <keys(s)>
```

An example might look like this.

```
SEAL SETNODE 1:234/56 15 FAm
```

This example would set the access level for 1:234/56 to 15 and their keys to `FAm'. This is handy if you wish to upgrade or downgrade a downlink's access in a batch file.

Status

This option tells Seal to create a status message to you as the sysop, which includes stats for your echomail and file areas, as well as your feeds and nodelinks. It will also include any areas in your queue file.

Tic

This option tells Seal to process incoming TIC files like Tick does. Refer to the TIC processing section of this document for more details.

Unlink

This option tells Seal to unlink a node from one, more than one, or all areas. For example, you wanted to manually remove 1:234/567 from the POINTS area, you would do the following:

SEAL UNLINK 1:234/567 ECHO POINTS

And if you wanted to remove 1:234/567 from POINTS and NODES:

SEAL UNLINK 1:234/567 ECHO POINTS NODES

In addition, if you wanted to remove 1:234/567 from all areas that you carry (whether that node is currently linked or not) you would do the following:

SEAL UNLINK 1:234/567 ECHO *

Note that use of this parameter without the -forced switch simulates a downlink's request exactly including security checks and request forwarding.

Update

This option tells Seal to rewrite your configuration files updating anything that needs to be updated due to new information in the SealCfg data files. You should do this if you change a downlinks' or file feeds' Tic flags using SealCfg, to update their entries in your TIC.CFG file.

Command Line Switches

The following switches may be used anywhere on the command line in any order.

-a<file>

Use this switch to use <file> instead of the configured AREAS.BBS file.

-c<file>

Use this switch to use <file> instead of SQUISH.CFG. This is handy if you use more than one SQUISH.CFG file or you wish to run Seal from a different directory than Squish.

-d

Use this switch to tell Seal to use DOS screen writes instead of direct screen writes while in SETUP mode. This is useful if you are using SEALCFG over a modem line and need to redirect the output over the modem.

-forced

Use this switch in combination with either the DESCRIBE, LINK or

UNLINK command line parameters.

When used with DESCRIBE, Seal will overwrite any description that it finds. When used with LINK and UNLINK, Seal will link/unlink the nodelink in question to an existing area without checking to see if the nodelink actually has access to that area. No forward requesting is done when using this switch.

-l<file>

Use this switch to use <file> instead of the configured log file. If Seal cannot open <file> for logging because it is in use by another program, Seal will echo the log to the screen only.

-ne

Use this switch to prevent Seal from reading the your echomail areas when using the ANNOUNCE cammand line parameter. Normally, ANNOUNCE will ensure Seal loads the echomail areas into memory. If this is the only reason it loads the echomail areas, you can supress this by using this switch. This will cause Seal to create default packets using your primary address and default header and footer files, but otherwise, the announcement messages will be fine. If you also specify another command line parameter that requires that Seal load the echomail areas, then they will be loaded regardless of this switch. Refer to Memory Requirments for a more detailed explanation of Seal's memory management.

-nl

This switch tells Seal to reverse the configured setting for notify with list. If you have Seal setup to notify with a list or available areas, then using this switch will cause Seal to notify with a list of linked areas instead. If Seal is configured to notify with a list of linked areas, this switch will cause Seal to notify with a list of available areas instead.

-nm

Use this switch to suppress the creation of messages. If used with ADD or DROP, no message will be created to the uplink. If used with LINK, PAUSE, RESUME, or UNLINK, no message will be created to the downlink.

-q

Use this switch to suppress all screen output. Normally Seal will echo the log file to the screen. If this switch is used, no log entries will appear on the screen.

-t<file>

Use this switch to override the configured TIC.CFG file name. <file> will be used as the Tick configuration file instead of what you have specified in SealCfg.

-x<file>

Use this switch when you use the HATCH command line parameter and you want to replace an existing file. If your downlinks have

replacements enabled, <file> will be deleted from the area before the new file is processed.

-#<n>

Use this switch to specify where to start numbering when using the %# template in the area tag for generating MSGAREA.CTL and FILEAREA.CTL. When the %# is used in the area tag template, Seal will start with <n> and increase for each subsequent area.

SEAL.INI

Seal gets its configuration from the SEAL.INI file. This file is similar in structure to a Windows INI file. The file is comprised of many sections, each with a section header and a series of keywords, each with an associated value, or a list of values, as follows.

```
[section]
key=value
key=value,value,value
```

There must be at least one blank line between each section. A blank line is defined as a line with no characters, only a carriage return; ie. no spaces, tabs, white spaces, etc.

For keys that have a boolean (or logical) value , you must use one of either: yes or no, true or false, on or off, or 1 or 0. Neither Seal nor SealCfg will recognize any other logical values.

Some keys have a list of values, separated immediately by one comma; no spaces. If a value contains a space, then it should be contained within double quotes, otherwise, simply the value will suffice; as follows.

```
key=part1,"part2a part2b"
```

For some of the keys that have multiple values, simply listing as many values needed will suffice. However, in such areas as the feeds sections, it is important to include all values, whether they be blank or not, by including the correct number of commas. For example, some thing like the following would not be uncommon.

```
1:234/5="Seal",PWD,0,F,LKI,,cne,"-0 -s",,,,FIDONET.NA
```

All file names must be specified in upper case only; same with area tags, passwords, path specifications, management flags, msg attribute flags, tic flags, and aliases.

It is VERY important that you maintain this structure, or else Seal will not work properly, and that is probably the least of what could happen.

When using SealCfg to edit SEAL.INI, the length of the input fields will be enforced. However, when editting SEAL.INI manually, there is no indication what the maximum length of the

fields are. You must keep this in mind, and if in doubt use SealCfg. When SealCfg or Seal load SEAL.INI and subsequently

write it when changes are made, any fields that originally were longer than the field's limit may be truncated.

If you are at all hesitant to edit the SEAL.INI file directly, you should side with caution and use the included SealCfg utility. SealCfg will ensure that the formatting is preserved within SEAL.INI which will in turn eliminate many possible unforseeable errors.

What follows is a brief description of each section and key in the SEAL.INI file.

[Global]

Sysop

This is your name. This name will be used as the originator of all messages to your uplinks.

Attr

This are used for messages to you as the sysop.

Pointnet

Use this keyword if your Tick configuration file requires two or three dimensional addresses. Seal will convert all your point address to a pointnet address before processing the Tick configuration file.

Backup

This keyword tells Seal to create backup files of your configuration files prior to writing any changes to disk. The backup files will have a name of *.SL.

MatchZone

This forces Seal to use the first zone matching aka in your aka list. By default, Seal will try to match the zone:net and if not found, then would try and match the zone, and if not found would use your primary address. If you would rather have Seal use the first matching zone address regardless of whether there is a matching zone:net address, use this.

GroupLines

Use this to group the configuration lines in your configuration files into groups when written by Seal. When this keyword is used, Seal will write all your EchoArea lines together, your EchoSpec lines together, your EchoAnn lines together, etc, instead of breaking them up into logical area definitions.

Optimize

Use this to tell Seal to optimize the writing of your

configuration files. If enable, Seal will omit unnecessary text like the redundant -p switch on each EchoArea line.

DeleteBad

When you use the BAD command line parameter and Seal encounters a message from a valid feed and the area tag is found in the delete file, Seal will delete the message from the bad area if this is set to `Yes'. If set to `No', Seal will leave the message untouched.

Registration

This is wehere you enter your registration number if you have one. A registration number is a six digit numeric string, such as 123456.

Language

These are the various languages you have configured as available to your nodelinks. Each language is a directory under the current directory in which is located the language files for that language.

LongNames

If you enable this option, then Seal will abandon the 8.3 directory/file name standard when creating paths for you areas. If an echomail area is created on your system whose echomail tag is SOUND_BLASTER, then the EchoArea line might look like this:

```
EchoArea SOUND_BLASTER C:\MSG\SOUND_BLASTER -$ 1:234/56
```

This option would only be of use to OS/2 users, as Squish might object to such a long name on a DOS based system.

SwapSize

Seal/386 will use as much extended memory as is available to it. However, if that is till not enough to load all your echomail and file areas, then you can tell Seal to use a swap file. Use this option to include the size of the swap file (in Kilobytes) you wish to establish.

LongPaths

This option is used mostly when converting Internet newsgroups to Fido echomail. When Seal creates an area, it would normally strip out the periods from the echo tag before costruction a path for the area. If this option is enabled, Seal will instead substitute a back-slash (\) for each period in the tag when creating a path. Consequently, a tag of ALT.SEX.ANIMALS would result in a relative path of ALT\SEX\ANIMALS.

Max3

This option governs what format of file is created when using the MsgCtl or FileCtl options. If enabled, Seal will create files compatible with Maximus 3.00, otherwise Maximus 2.02 is

supported.

[Areas.Bbs]

Level

This is the access level associated with any area listed in your AREAS.BBS file.

Locks

This is the lock characters needed to access any area listed in your AREAS.BBS file.

Description

This is the description that will appear with any area listed in your AREAS.BBS file.

Dimension

This is the minimum number of dimensions to use when writing node addresses in your AREAS.BBS file. If Seal needs to write more than this number, then it will.

[Filenames]

Squish

This is your Squish configuration file.

Tic

Use this if you want to enable the Tick processing mode of Seal. In this mode, Seal will manage your file areas in the same way it manages your echomail areas. To disable this function, leave this field blank.

Queue

If this is used, the default name of the queue file will be overridden by the one you specify. The queue is where Seal stores information about areas that have been requested by your downlinks that have been requested from you uplinks but have not arrived yet.

Delete

If this option is used, the default name of the delete file will be overridden by the one you specify. The delete file is where Seal stores the tags of areas that have been dropped from your system either by using the Drop command line parameter or by response mode. This ensures that Seal does not create an area based on a message in your bad area in an echo if that unlink message has not gone out yet. You may want to periodically delete this file as it may grow quite large in a very active system. Seal will remove tags from this file if it requests an area from your uplink.

Pause

This is the name of Seal's pause file in which it will store areas (both echo and file) that are paused by use of the %PAUSE token or PAUSE command line parameter.

Language

This is the default language to use. If not specified, Seal will use the hard-coded defaults.

EchoRescan

This is where you specify the location of the batch file that Seal will produce to enable rescans. This file is deleted if found before response mode is enabled.

FdRescan

Use this to specify a FrontDoor rescan file to create if Seal deletes a message while in KILL mode.

UEFTS

This is the path and file name of your UEFTS volume tracking file.

FileStat

This is the path and file name where Seal should save traffic statistics for your file echo processing. Leaving this field blank will disable the keeping of file stats.

[Template]

EchoArea

Use this to specify the format of EchoArea lines in your SQUISH.CFG file when Seal rewrites it. Complicated translation characters are valid here. The line specified here will be copied verbatim into SQUISH.CFG after any translation characters are replaced. If you leave this line blank, Seal will use the hard-coded defaults. An example might look like this.

```
EchoArea %15t %30p %10s %n
```

WARNING: Using this option with care as if you omit one of the translation characters, your SQUISH.CFG file will be rewritten with no hope of Seal being able to read it again. If in doubt, leave this option blank.

EchoSpec

This is functionally similar to the template for EchoArea lines. The same warning applies. An example of this option might look like this.

FileSpec

This is functionally similar to the template for EchoSpec lines. The same warning applies. An example of this option might look like this.

;FileSpec %15t %-3a %5l %d

EchoRescan

This is functionally similar to both the EchoArea and EchoSpec templates. This is the template for writing to the Squish rescan batch file. An example might look like this.

SQUISH RESCAN %t %a

FileList

This is again similar to the EchoArea line with the same warning applied. This line will be used as the template for updating the file lists for file areas when a file is received or hatched from an area you carry. This can be overridden by a similar line in the area definition in TIC.CFG. An example might look like this.

%13f %:65d

[Log]

File

This is the file that Seal will log to. What gets in the log file depends on the log level. If no log file is specified, then no log file is created, although the log lines will still echo to the screen.

Level

This is the specification for how much information is to be contained in the log file. Seal will use the same type of logging style as Squish. The default log level for Seal is 6.

FD

Use this to enable FD style logging instead of Squish (Opus) style logging.

[EchoAreas]

Alias

This is any and all aliases you want Seal to recognize messages to for echomail area requests.

NewPath

This is where Seal should create all new areas. You must specify

a directory that exists. Seal will create Squish areas in this directory, and will create directories under this one for *.MSG

areas. This can be overridden by specifying a new area path for an echomail feed.

[FileAreas]

Alias

Use this to define names which Seal will recognize as messages regarding file areas.

NewPath

This is where Seal should create all new areas. You must specify a directory that exists. This path can be overridden by specifying a new area path for a file feed.

[Message]

Copies

If this is used, a copy of all messages generated by Seal will be sent to you except query and help messages. Note that the copy messages will be exact copies of the message sent to your downlinks (ie. same language).

INTL

Use this if you want Seal to include an INTL line in every message. By default, Seal will only use such a line if it is needed.

PID

Use this is you want Seal to use the PID kludge instead of the tear line for it's indentification.

FakeName

Use this to specify who you want Seal to pretend to be. The name specified here will be used as the From: in all message to your downlinks.

KillProcessed

If this is used, Seal will delete messages that have been received by Seal.

LeaveUnRcvd

If this is used, messages that are received by Seal will not be marked as such. Be careful with this one, as Seal will only process messages that have not been received and if you use this, then Seal will process the message over and over until you change this or you delete the message.

ShortList

Use this to list the areas in a query or linked reply across the page with no description rather than the usual down the page with descriptions.

ShowFeeds

This tells Seal to mark echo and file areas that the downlink is the feed for when replying to a request with a list of linked area. The mark character is a @ unless overridden by a language file specification.

NotifyList

If this is used, all notify messages will use a list of available areas as opposed to the default list of linked areas.

RespondLinked

Use this to always include a list of linked areas with response messages to your downlinks.

RespondList

Use this to always return a list of available areas with response messages to your downlinks.

DefaultDescription

Use this to specify a description for all echo and file areas that do not already have one.

Split

You can use this option to specify how you want Seal to split long messages. If left blank or set to zero (0), Seal will never split messages. To split messages into 12K messages, then set Split=12.

[Files]

CRC

If enabled, Seal will calculate the CRC for each file that is associated with a TIC file and verify that the CRC of the file matches that listed in the TIC file. TIC files that do not contain a CRC line will be passed as valid.

Fido

Use this option to have Seal create file attachments as a method of sending files. Otherwise (set to `No') Seal will create FLO type files.

Rad

If enabled, Seal will create RAD files in the hold directory when a file is hatched from your system. This allows Seal to announce the hatching of the files.

Replace

If enabled, and a Replace line is present in a TIC file, Seal will attempt to delete the old file before moving the new file into the area.

FileList

If enabled, and you hatch or receive a file that replaces an existing one, Seal will delete the existing file from the file list in that area. Otherwise, Seal will leave it in the file list, but still delete the file (assuming Replace is enabled) and it show as offline.

OldFiles

If you specify a path here, Seal will move a replaced file to this directory instead of deleting it. If left blank, Seal will attempt to delete the file.

MultipleDesc

If enabled, Seal will place a file's description in a TIC file so that each line doesn't exceed 80 characters, using as many Desc lines as necessary. Properly written TIC file processors should be able to read such a configuration (Seal does), but if you have problems, turn this off, and Seal will only use one Desc line, no matter how long it is.

In

This is the directory you want Seal to look in for incoming files and their associated TIC files.

Hold

This is where Seal will store the TIC and RAD files it creates. Seal will maintain this directory on it's own, deleting TIC files as they are no longer needed. You should not store anything in this directory, and it should not be shared with any other utility.

SendAs

Seal will use this name as the `From' name in file attach messages to your downlinks. If left blank, Seal will use `Seal'. This can be used in combination with other utilities that look for file attach messages from specific names. Note that this option is not used for non-Fido mode mailers (like Binkley) that create FLO type files.

[Announce]

To

Use this keyword to specify who all file announcement messages will be addressed to.

From

Use this keyword to specify who all file announcement messages are from.

SubjectFile

Use this keyword to specify the subject of all file announcement messages.

SubjectArea

Use this keyword to specify the subject of all new area announcement messages.

HeaderFile

Use this keyword to specify a header file for file announcement messages. This file will be put at the top of all file announcement messages.

FooterFile

Use this keyword to specify a footer file to be placed before the origin line in all file announcement messages.

NewAnnounce

Use this keyword to add an ;Announce line to all new file areas created.

[MsgArea.Ctl]

Area

Use this to specify a template for the area name when using the MSGCTL command line parameter.

Extra

Use this to specify extra lines you want to appear at the end of the Maximus MSGAREA.CTL area definitions.

TagDesc

If enabled, Seal will use the areas tag as it's description when creating a MSGAREA.CTL type file.

[FileArea.Ctl]

Area

Use this to specify a template for the area name when using the FILECTL command line parameter.

Extra

Use this to specify extra lines you want to appear at the end of the Maximus FILEAREA.CTL area definitions.

Upload

Use this to specify a common upload directory for all areas written when using the FILECTL command line parameter. If you do not use this, then Seal will use the download directory as the upload directory as well.

TagDesc

If enabled, Seal will use the areas tag as it's description when creating a FILEREA.CTL type file.

[Security]

ValidateName

If this option is used, Seal will check the name in the from field of a downlink's request against that in your setup and if they are different will treat the request as unauthorized. Without this option, Seal will assume any message from the downlink's node address with the right password is authorized. Seal will however, send any messages back addressed to the name configured in your setup

CheckFreeRide

If this option is used, then Seal will ensure that any unlink request will not result in an areas being sent only to nodelinks that are not allowed a free ride. If this option is not used, then Seal will treat all nodelinks equally.

ShowProtected

If this is used, Seal will list all the areas available on your system whether a node can link to them or not. Protected areas under this scheme will be flagged with a set of <> (or language file equivalent).

Secret

Use this to ignore messages from addresses that you do not have configured on your system. Seal will ignore all messages addressed to it at any of your akas if they are from a node address other than any of your downlinks. If you enable an open

system, then the open system will take precedence over this setting.

IgnoreFrom

Use this to configure Seal to ignore messages from certain names. Seal will ignore any message that contains any of the names specified here in the From field. This is useful if you want Seal to ignore messages from programs like MsgTrack.

NoQuery

Use this if you want to disable the -Q (or language file equivalent) switch by the downlink on the subject line. If used, Seal will ignore the switch and will not return a list of linked areas. This also affects the %QUERY token.

NoHelp

Use this is you want to disable the -H (or language file equivalent) switch by the downlink on the subject line. This also affects the %HELP token.

NoPause

Using this will disable a downlink's ability to use the %PAUSE and %RESUME tokens in their request messages.

NoTic

Using this will disable a downlink's ability to use the %TIC token in their request message.

NoNotLinked

Using this will disable a downlink's ability to use the %NOTLINKED token in their request message, as well as the -N subject line switch.

NoRescan

This disables the use of the -R (or language file equivalent) switch by your downlinks on the subject line. Notice is given to the downlink when a rescan is requested but disabled by the use of this switch.

[OpenSystem]

OpenSystem

Use this to tell Seal to accept open system requests. When operating in open system mode, Seal will respond to messages to it from anyone and respond as if the originator of the message was listed in a nodelink line.

Seal will link/unlink the originating system following normal rules and send a response message. Under an open system, all query and linked request messages will be honoured although you cannot notify systems that are not defined as nodelinks.

Level

This is the access level to assign to open system requests.

Keys

Use this to specify the keys associated with an open system request.

Attr

This is the message attributes to place on the reply message to an open system rtequest.

[NodeLinks]

addr

The node address of this downlink.

name

The name of the sysop of the downlink. Only request from this name at the node address with a correct password will be honoured.

pw

A password that should appear in the downlink's message subject line for the request to be valid.

level

The security level to assign to this downlink. It should be a number from 0 to 255.

keys

A series of letters that refer to the locks that this downlink can access. Each lock is a single letter from "A" to "Z" and is case-sensitive. Specify more than one lock by stringing them together without spaces. To specify `always has access to everything' use an asterisk (*).

dlevel

This is the security level that Seal will use to show the downlink areas in a query list. If this is set to something `above' the access level, then Seal will show the areas they would have access to if they had this level as protected. This is handy if you want to show areas to a downlink but don't want them to have access to them, but yet, don't want them to see every area they don't have access to.

dkeys

This is the keys associated with the display level.

attr

The message attributes for messages sent to this nodelink.

flags

The flags to be added to this nodelink's entry in your Tick configuration file.

mgmt

Allow to manage your areas

Allowed to manage your system from their end. This includes the ability to use the %DELETE token for any of your areas.

Treat as an auto system

Will be sent an auto delete message when an area is dropped instead of a user-type notice of the dropped area. When the downlink's system receives the auto delete message (and they're running Seal as well) then their system will automatically drop the area in question instead of the downlink have to do it manually.

Do not allow a free ride

Will not receive a free ride at other downlinks' expense. If an unlink request from any downlink results in an area being carried only by nodes with this enabled, then the area will be dropped and the affected nodes will be sent a standard `dropped area' message. The same will happen when you use the CLEAN command line parameter.

Allow forward requests

Will enable forward requests from this system to be forwarded to your uplinks. If you set this to `No', then this nodelink will only be allowed to request areas that you already carry.

This works well in combination with the free ride option to limit a nodelink's ability to abuse your system.

Exclude from notify

Will not receive a notify message when the NOTIFY command line parameter is used unless they are specifically listed on the command line.

language

The language file to be used when sending messages to this downlink. If no file is specified, or the file cannot be loaded, then the default language file will be used.

akas

If your downlinks has more than one address and ca not seem to quite remember what address to use when talking to Seal on your system, you can add their akas here so that Seal will respond to your downlink at their primary address no matter what address they send their request from.

If you specify another nodelink as an aka, Seal will allow a downlink to link and unlink areas accessible by both node addresses, while still using the correct address when adding the downlink to an area.

[EchoFeeds]

addr

Node address of the uplink.

name

The name of your uplink's Seal equivalent. Seal will address all requests from your downlinks to this user at this uplink's address.

pw

Your password for this uplinks system. It will be specified on the subject line of all requests sent to this uplink. It should be one word with no spaces.

level

The security level that is assigned to any area requestable from this uplink and is added to any areas created by this uplink. This can be a number from 0 to 255.

locks

The locks that are assigned to any area requestable from this uplink and is also assigned to any area created from this uplink. It can be any number of letters from `A' to `Z'.

attr

These are the message attributes to place on messages sent to this uplink.

aka

The address to use as the origin address sending a forward message to this uplink. It will also be used as the -p address when creating an areas if it's different than your primary address.

cne

Refers to whether your downlinks may forward area request to this node. If you say `No' then Seal will ignore all but the address field of this uplink. If however you say "Yes" you must complete the rest of the definition.

switches

The switches to be added to the switches field in the EchoArea line of any area created from this uplink. The switches recognized by Seal are as follows.

-s -x<node> -+<node> -$ -f -h -0 -a -u<node>

You need not include the -p switch here as Seal will automatically work that one out.

group

This is the area group in which you want Seal to place newly created areas. The group specified here must already exist in your configuration files.

announce

You may specify any number of either echomail tags or node addresses. For each echomail tag, an echomail message will be created listing any areas that were created from this echomail feed when using the BAD command line parameter. For each node address specified, a netmail message will be created with the same information addressed to that node address. If the node address can be found as a nodelink, then the nodelink's name will be used in the `To' field.

dir

This is the directory where you want areas created by this echomail feed to be created. If this echomail feeds created Squish-style areas, then they will be created in this directory. If this echomail feeds created *.MSG areas, then a directory will be created under this directory. You must specify a directory that exists. This specification will override that of the default echomail new area path.

nodes

You may specify one or more node addresses of nodelinks that you want to be auto-linked to any areas created by this echomail feed. Separate each address with a space.

lists

The forward list for this node. It is a file that lists the areas that your downlinks may request from this uplink through you. This file must be standard ASCII text. If the forward list is in

AREAS.BBS format, precede the file specification with a `@' like so.

Otherwise Seal will expect the file to simply be a list of tags with optional descriptions.

[FileFeeds]

addr

Node address of the uplink.

name

The name of your uplink's Seal equivalent. Seal will address all requests from your downlinks to this user at this uplink's address.

pw

Your password for this uplinks system. It will be specified on the subject line of all requests sent to this uplink. It should be one word with no spaces.

level

The security level that is assigned to any area requestable from this uplink and is added to any areas created by this uplink. This can be a number from 0 to 255.

locks

The locks that are assigned to any area requestable from this uplink and is also assigned to any area created from this uplink. It can be any number of letters from `A' to `Z'.

attr

These are the message attributes to place on messages sent to this uplink.

aka

The address to use as the origin address sending a forward message to this uplink. It will also be used as the -p address when creating an areas if it's different than your primary address.

cne

Refers to whether your downlinks may forward area request to this node. If you say `No' then Seal will ignore all but the address field of this uplink. If however you say "Yes" you must complete the rest of the definition.

flags

These are the flags supported by Tick to place on this uplink's entry line in your Tick configuration file.

group

This is the area group in which you want Seal to place newly created areas. The group specified here must already exist in your configuration files.

announce

You may specify any number of either echomail tags or node addresses. For each echomail tag, an echomail message will be created listing any areas that were created from this file feed when using the ANNOUNCE command line parameter. For each node address specified, a netmail message will be created with the same information addressed to that node address. If the node address can be found as a nodelink, then the nodelink's name will be used in the `To' field.

path

This is the directory where new areas will be created from this file feed. You must specify an existing directory, and will override the default new area path for files.

nodes

You may specify one or more node addresses of nodelinks that you want to be auto-linked to any areas created by this file feed. Separate each address with a space.

lists

The forward list for this node. It is a file that lists the areas that your downlinks may request from this uplink through you. This file must be standard ASCII text. If the file is in standard FILEBONE.NA format, the precede the file specification with a `@' like so.

@H:\FILE\FIDO\FILEBONE.NA

Otherwise Seal will expect the file to be simply a list of tags with optional descriptions.

[ProtectedAreas]

tag

The tag of the area you wish to protect.

type

`No' if the area you are protecting is an echomail area, `Yes' if the area is a file area.

level

The access level (0-255) which you want assigned to this area.

locks

The locks to assign to this protected area.

SEALCFG

SealCfg is the suggested way to edit SEAL.INI. However, there is limited help within the utility to attempt to keep the size relatively small. You should refer to the previous section (SEAL.INI) for details about what each option means.

IMPAREAS

ImpAreas is a quick and dirty little utility to convert an AREAS.BBS style file to a SQUISH.CFG style file. The syntax for running this utility is as follows.

IMPAREAS <areas.bbs> <squish.cfg> <access_level> <lock(s)>

<areas.bbs> is your AREAS.BBS file that presently contains your area definitions. <squish.cfg> is your SQUISH.CFG file to which you want ImpAreas to append the areas from AREAS.BBS. Each area read from AREAS.BBS will be added to the end of your SQUISH.CFG file with access level <access_level> and locks <lock(s)>. For example.

IMPAREAS AREAS.BBS \SQUISH\SQUISH.CFG 10 FS

Note that you should only run this utility once since it does append the areas to your SQUISH.CFG file. Note also that it does not read SQUISH.CFG first so it will not properly handle split area definitions -- you will have to edit any split area definitions you have.

CONFIGURING SQUISH.CFG

Squish Keywords

Address

This is the Squish specification for your address like so:

Address 1:234/567
Address 2:345/678.9

The first address is considered your primary address. All of these addresses should be at least 3d (zone:net/node).

AreasBBS

This is the Squish specification for your AREAS.BBS file. Seal will read from this file as if echos where included in the Squish configuration file. When changes are made, areas from this file will go back to this file. You may also configure Seal to create new areas in this file.

BadArea

This is the Squish specification for your bad message area. Seal uses this area to scan for bad message from which to create new areas if you set it up that way. This area may be *.MSG or *.SQD, if the latter, add a -$ after the path. Your line should look something like this:

BadArea BAD_MSGS \MSG\BAD

BinkPoint

If this in enabled, when creating FLO files, Seal will enable full 4d Binkley-style point directories.

DefaultPacker

This is where you specify the default archiver for your downlinks. Seal will use this archiver if a downlink is not explicitly listed in a Pack line.

EchoArea

This is the Squish specification for an echomail area. The format is as follows:

EchoArea <tag> <path> <switches> <feed_node> [node(s)]

Consult the Squish documentation for more details.

NetArea

This is the Squish defined netmail area. Seal will try to use as many netmail areas as you have defined, although some messages will always be created in your primary (first listed) netmail area. Seal does not read the -p switch on a NetArea line. All requests in a netmail area will be replied to in that netmail area. Uplink messages as well as notify and status messages will be created in the primary netmail area. A netmail area might look like this.

NetArea NETMAIL \MSG\NET

Your netmail may be either *.MSG or *.SQD, if the latter, add a -$ after the path.

NetFile

This is the path to your mailer's inbound directory. Seal will put announcement packets in this directory for Squish to find and toss.

Outbound

This is where Seal will put the FLO files if running in non-Fido mode when processing TIC files, and hatching files. Consult the

Squish documentation for a full explanation of this keyword.
Seal will also support the domain kludge when specifying a zone

number after the outbound path when multiple outbound lines are used.

Pack

This line (or lines) specify the archiver used for each of your downlinks. Refer to the SquishMail documentation for a complete description of how to use this keyword. Seal will read these lines to determine the archiver used for each of your downlinks.

Password

This line (or lines) enable packet level passwords. Seal will read and use these passwords if present when creating packets.

Seal Keywords

;EchoAnn

Use this keyword to associate either a header file and/or a footer file for announcement messages that appear in this echo area. The format of this line is as follows.

```
;EchoAnn <tag> <header_file> <footer_file>
```

To simply use both default header *and* footer files, do not use this keyword. To use the default header *or* footer file use a period (.) as the file name. To over-ride the default files with no file, use NONE as the file name. For example.

```
;EchoAnn MUFFIN NONE MUFFIN.FTR
```

This would specify no header file and MUFFIN.FTR as the footer file.

;EchoSpec

This is the Seal specification for a particular echo. This applies to the last echo seen, so you would normally do it like this:

```
EchoArea  TUB \MSG\PASSTHRU\TUB -0 -$ 1:234/567
;EchoSpec TUB 2 F (Fido) SquishMail Support Echo
```

The syntax is as follows:

```
;EchoSpec <tag> <level> <locks> <desc>
```

<level> is a number from 0 to 255 needed by a downlink to gain access to this area.

<locks> is any number of lock characters that is needed by a downlink to gain access to this area. Locks can be "A" to "Z".

<desc> is anything left on the line and is displayed in linked and list messages by Seal.

;SealGroup

This is to create logical groups out of your echo or file areas. The syntax is as follows:

```
;SealGroup <tag> <desc>
```

<tag> is the group tag that relates to the group of areas.

<desc> is the description of this group of areas.

An example might look like this:

```
;SealGroup FIDO FidoNet Echo Areas
EchoArea BACKBONE \MSG\BACKBONE -$ 1:234/56
;EchoSpec BACKBONE 10 F BackBone Conference
EchoArea SYSOP234 \MSG\NET234 -$ 1:234/56
;EchoSpec SYSOP234 50 F Net 234 Sysop Echo
```

You may use any number of groups, either in your SQUISH.CFG and/or your TIC.CFG file. You may also use the same group tag for both your echo and file areas.

CONFIGURING TIC.CFG

Tick Keywords

Some keywords have associated menu options in SealCfg, such as Crc, Hold, Raid, Replace, In, and Fido. These need not be present in TIC.CFG if you are using Seal as your file echoing software -- you can enable them using SealCfg. However, enabling them in TIC.CFG will override the setting through SealCfg.

Area

This defines a file area. The syntax for an area definition is as follows.

```
Area <path> <tag> [tag]
 [Local ListName <file>]
 [Local ListFormat <fmt>]
 [Local Passthru]
 [Local BigDesc <mask> <tag(s)>]
 [<node> <password> [flags]]
 [{more nodes}]
```

A blank line must separate area definition blocks from other area block or other keywords. The Local directives are as follows.

Local ListName

This tells Seal to update <file> instead of FILES.BBS in the area directory. You must include a full path and filename here. Note that this is ignored if the area is a pass-through area.

Local ListFormat

This tells Seal to override the default format for writing to the file list for this area. Refer to List format in the menus for complete details on specifying a list format.

Local Passthru

This tells Seal to treat this areas as a pass-through area. All files arriving and hatched in this area will be deleted when there are no associated TIC files for them. You should not store files in this area.

Local BigDesc

This tells Seal that if a file arrives or is hatched in this area that matches <mask> to create a packet with the file as the contents of a message bound for echomail area <tag>. You can specify any number of tags you wish. If you would prefer a netmail message, then use * as the tag.

Crc

This keyword tells Seal to perform a CRC check on incoming files. If a file fails a CRC check, the TIC file will be renamed to BAD. Incoming TIC files that do not contain a CRC will not have their associated file checked regardless of this keyword.

Fido

If this keyword is enabled, then Seal will create file attachment message to send TIC files and their associated files. If commented out, Seal will create FLO files in the various outbound directories.

Hold

This directory is where Seal will store its TIC and RAD files. This directory must not be shared by anything else. The syntax is as follows.

Hold <dir>

In

Seal uses this directory to search for incoming TIC files. The syntax is as follows.

In <directory>

Raid

This keyword enables the creation of RAD files when you hatch a file from your system. Seal can read these when using the Announce command line parameter and announce them.

Replace

This keyword enables the replace option of Seal when Seal tosses a file to an area. If this keyword is enabled, and a Replaces keyword is in the TIC file, then Seal will attempt to delete the old file before it tosses the new file to that area.

Seal Keywords

;Announce

Use this keyword so that files processed in this file echo are to be announced in the specified echomail areas. For example, to announce the BACKBONE file echo files in an echo called BACKFILE, your TIC.CFG entry would look like this:

```
Area \FILE\BACKBONE\ BACKBONE
     1:234/56      SECRET
     ;FileSpec BACKBONE 0 F FidoNet Backbone Files
     ;Announce BACKBONE BACKFILE
```

You may specify more than one tag by separating them with spaces like this:

```
;Announce BACKBONE BACKFILE NEWFILE
```

The absence of an ;Announce line in the area definition tells Seal that you do not want files in this area to be announced.

If you wish to announce files that have arrived in a file area to a particular node address, then instead of specifying an echomail tag, specify a node address. A message similar to the echomail announcement message will be created in your netmail area addressed the address specified in the ;Announce line.

;FileSpec

This is the Seal specification for a particular file area as follows.

```
Area \FILE\FIDONET\ BACKBONE
   1:234/56  SECRET     *
   ;FileSpec BACKBONE 2 F (Fido) Backbone files
```

The syntax is as follows:

```
;FileSpec <tag> <level> <locks> <desc>
```

<level> is a number from 0 to 255 needed by a downlink to gain access to this area.

<locks> is any number of lock characters that is needed by a downlink to gain access to this area. Locks can be "A" to "Z".

<desc> is anything left on the line and is displayed in linked and list messages by Seal.

;SealGroup

(See ;SealGroup under Configuring SQUISH.CFG.)

NODELINK INSTRUCTIONS

Talking to Seal

To talk to Seal, you must create a netmail message. You should address the netmail message to `Areafix' or `Seal Echo', or simply `Seal' if you want to request echomail areas. To request file areas, you should address your message to `Raid' or `Seal File'.

You must enter your password on the subject line as the first word on that line. Case does not matter.

Then in the message body, you list the echos you wish to request, one per line. If you wish to drop an echo, then precede the tag with a dash.

For example, to request the MUFFIN echo and drop the TUB echo, the message would look like this (if your password was `FUN').

```
--------------------------------------------------
By:  Robert Presland (1:234/56)
To:  Seal (1:234/0)
Re:  FUN
--------------------------------------------------
MUFFIN
-TUB
```

Seal will also understand a request if you place a plus sign in front of the tag.

Subject Line Switches

You may also specify a number of switches after your password on the subject line. Separate the switches from your password as well as other switches with at least one space. The switches are as follows.

-Q Send a list of all the areas that are available.

-L Send a list of areas that you are linked to.

-N Send a list of areas that you are not linked to.

-R Rescan all areas listed in the message.

-H Send a help message.

Message Text Tokens

Within the message body, among the tags, you may also use the following tokens.

%ECHO

This signals Seal that all the tags on succeeding lines in the request message are echomail tags, regardless of the original status of the request message.

%FILE

This signals Seal that all the tags on succeeding lines in the request message are file area tags, regardless of the original status of the request message.

%HELP

Return a message that includes help on how to use Seal.

%LANGUAGE

Change current language. To change to the French language file, include this in your message:

%LANGUAGE FRENCH

If you omit the language name or specify a language that is not available then a list of available languages will be included in the reply message

%LINKED

Include a list of areas that you are linked to in the reply message. This is the same as using the -L switch.

%LIST

See %LINKED.

%NOTLINKED

Return a message with a list of areas that are available but that you are not linked to. This is the same as using the -N switch.

%PASSWORD

Change your password to the word following the token. For example, to change the password to `nofun' you would do the following.

%PASSWORD NOFUN

%PAUSE

This tells Seal to unlink you from all the areas you are currently linked to, but save a list of the areas, and wait for you to resume your areas by using the %RESUME token.

%QUERY

Return a message with a list of all the areas that are available to you. This is the same as using the -Q switch.

%RESCAN

This signals Seal to turn on the rescan toggle. All area requests in succeeding lines that result in a `Linked' or `Already linked' message will be rescanned. This is the same as using the -R switch.

%RESUME

This tells Seal to relink you to all the areas that you were previously linked to before using %PAUSE.

%TIC

This tells Seal to change your profile so that you will no longer receive a TIC file if were receiving one, or to start receiving one if you have not been receiving one.

Extended Tokens

Nodelinks that are allowed to remotely manage your system may use these tokens.

~<tag>

This is functionally identical to the %DELETE token, but is only applicable to echomail areas.

%CREATE

This tells Seal to create an area on your system as if you had used the ADD command line parameter. For example, to add the MUFFIN echo from your feed (1:234/56), a nodelink would do this.

%CREATE ECHO MUFFIN 1:234/56

To add a file area, replace ECHO with FILE. Note that you must have allowed the echomail feed to create areas just like you would if you were using the ADD command line parameter.

%DELETE

This tells Seal to drop an area from your system as if you had used the DROP command line parameter. For example, to drop the MUFFIN echo from your system, a nodelink would do this.

%DELETE ECHO MUFFIN

To delete a file area, replace ECHO with FILE.

%FILESTAT

Return a file statistics message to the nodelink identical to the message created by using the FILESTAT command line parameter.

%FROM

This is used to simulate a request from another downlink. For example, say 1:234/1 is allowed to manage your areas, and s/he wants to link another downlink, say 1:234/4 to MUFFIN, the message text would look like this.

```
%FROM 1:234/4
MUFFIN
```

Note that the header of the message must be in accordance to standard practice for a request from 1:234/1 (password, `From:' name, etc). Further, using this token changes the current node address for future tokens within the same message. So if you also with to use the %RECEIPT of %FILESTAT or any other extended token, they must be specified before any %FROM tokens in the same message.

%RECEIPT

This is used to send a copy of the reply message to another person. Say for example, 1:234/1 is allowed to manage your areas, and s/he links 1:234/4 to MUFFIN by using the %FROM token, then 1:234/1 might include a %RECEIPT line in the message to send a copy of the reply message to 1:234/4. That line would look like this.

```
%RECEIPT 1:234/4
```

%SETNODE

This is functionally identical to the SETNODE command line parameter. For example, if a downlink who was allowed to manage your areas, say 1:234/1 wanted to set the access of another downlink, say 1:234/4 to level 10 and keys FS then s/he would do this.

```
%SETNODE 1:234/4 10 FS
```

Note that the header of the message must be in accordance to a request from 1:234/1 (password, `From:' name, etc).

%STATUS

Return a status message to the nodelink identical to the message created by using the STATUS command line parameter.

HOW SEAL WORKS

Request Forwarding

If a downlink requests an area that you are not currently carrying, it looks in all forward lists that the downlink has

access to until it finds the tag as specified by the downlink in their message. Assuming it finds it somewhere, it will create a request message to that feed and then place the area in the queue, as well as remove it from the del file if present.

The above applies to both echo and file areas.

When Seal scans the bad message area and finds a message from one of your feeds, it will create the area (if that feed is allowed to create areas). Seal will remove it from the queue file if it exists there, and link all nodes in the queue file to the newly created area. It will also create a message to each of the linked nodes announcing the arrival of that area.

Seal will do the same thing for file areas when it finds a TIC in your file inbound when performing an ANNOUNCE from one of your file feeds.

One more thing, Seal will delete pass-through areas if all downlinks unlink themselves from that area. In the case of echo areas, the unlink message goes to the feed. In the case of file areas, unlink messages will go to all linked nodes with a * in the switches for that node. A downlink in a file area is a node without a * in their switches.

Seal will also keep track of the echo areas that have been dropped so that they do not get created by a bad message arriving before the unlink message is received by your uplink.

File Statistics

Seal can keep it's own file traffic statistics in addition to supporting UEFTS. To enable the keeping of file traffic statistics simply specify a path and file name under `File stats'.

File statistics can be viewed by using the FILESTAT command line parameter, or by a downlink who is allowed to manage your areas, using the %FILESTAT token.

A message will be created that lists all your file area tags, the number of files, the total number of bytes of traffic, and the average size of each file for each area you have configured in TIC.CFG.

To reset the stats, simply delete the stats file.

File statistics include both files coming into your system and files that are hatched by your system.

Language Support

You can configure Seal to display any language you desire through the use of external language files. Despite what language you impose on your downlinks, Seal will continue to log in English, as well as create messages to you in English.

All message texts have defaults so Seal will work with no problems with no language file. However, if you want to change one line of a message, or rewrite the entire message texts, you may do so by using language files.

Each set of language files is stored in a sub-directory of the Seal directory. The name of this directory is the name of the language as it appears to your downlinks. Each set is comprised of four (4) files.

LANGUAGE.INI	Most of the text that is displayed ot the downlink.
NOTIFY	The header for the notify messages.
NOACCESS	The message text for the message that states that a downlink has no access to Seal on your system.
HELP	The message text for the help message.

The LANGUAGE.INI file is made up of any number of lines. Some keyword lines allow the use of translation characters such as %c and %n which you may place in the message text and Seal will replace them with the appropriate information. You may also use extended translation characters to enable formatted replacements in both the notify header and the help text. Refer to the distribution language files for more details on what they should look like.

Again, you need not include a full language file. Only include those texts that you wish to change, the remaining texts will remain the defaults.

If you have a language file specified for a downlink specifically, then that file will be used for messages to that downlink. If no file is specified or the file specified cannot be loaded, then the default language file will be used. If there is no default specified or it cannot be loaded, then the hard coded defaults will take effect.

If you would like to contribute a valid language to Seal's languages, that is, a real language as opposed to pig latin or valley talk, please send it to the author, as listed in the Bugs, Suggestions & Comments section. Your language file will be included in the next release of Seal and you will receive credit in these docs.

UEFTS Support

Seal will create a UEFTS compatible volume tracking file for use with volume tracking software such as FidoBill. To enable volume tracking, specify a path and file name under `UEFTS'. Seal will add records to this file when a file is hatched from your system, and when files are received for processing by your system.

Compatibility Notes

Seal was originally designed to be able to interface with as many different AreaFix clones as are available, as well as the original Tick.

When compatibility mode is turned on, Seal will create messages that are similar in structure to those created by AreaFix so that any AreaFix clone that reads AreaFix replies should still function properly.

Seal also creates TIC files that should be readily processed by any Tick compatible TIC processor. Despite this fact, there is a discrepancy between the time stamp used by Tick and Hatch, and that of Seal. For this reason, Seal does not support pre-release.

Seal also does not support some of the more exotic features of Tick, such as ListFmt, and Dupe checking.

RULE OF THUMB: If it's not documented in this document, Seal probably does not support it.

TIC SUPPORT

Managing Your TIC.CFG File

Seal will also manage your TIC.CFG file if you wish it to. Since most of the functions are the same between Squish and Tick, this is a relatively easy thing to do. If you specify a filename for Tick, Seal will process your TIC.CFG file every time you run Seal just as it would your SQUISH.CFG. Note that you cannot disable the Squish processing and just use the Tick processing.

Seal will assume that all access levels and passwords between Squish and Tick are the same. That is a node's password for Squish areas is the same as that for the Tick areas.

When writing the TIC.CFG file, Seal will automatically write your primary address as well as all your Akas as listed in SQUISH.CFG. This is so Seal can automatically determine the Ax flag for adding a downlink to an area. (Consult the documentation with Tick 2.10 or later for complete details on this switch.) To be consistent, Seal will also add the Ax switch to each nodes entry in TIC.CFG when needed to ensure they match the akas listed. This allows you to change your akas in SQUISH.CFG and they will be changed accordingly in your TIC.CFG next time it is written. To force the update, you must do a SEAL FORMAT.

If you have used a pointnet number when configuring Seal, it will convert your points' addresses to 3D pointnet address for processing.

Announcing Files

Seal will look for *.TIC and *.RAD files if you use the Announce

command line parameter, and create an echomail message including all new file arrivals.

Seal will read your TIC.CFG file (or equivalent) to locate the *.TIC and *.RAD files (*.RAD files are created when you hatch a file into a file echo). Seal will create packets from any TIC and RAD files and leave them in your Squish NetFile directory for Squish to toss. Each announce message will contain all files to be announced in that area regardless of file area tag.

For example, say you have the following.

Tic file	File tag	File name	Echo tag
TK000001.TIC	BACKBONE	FIDONET.NA	NEWFILE,FIDOFILE
TK000002.TIC	NODEDIFF	NODEDIFF.A71	NEWFILE
TK000003.TIC	ALT_BONE	ALT_NET.NA	ALT_NET,NEWFILE

ALT_NET is an alternate FTN network that uses one of your akas, say 123:456/7 instead of your primary address, say 1:234/56. The EchoArea entry in SQUISH.CFG for ALT_NET would look like this.

```
EchoArea ALT_NET \MSG\ALT_NET -$ -P123:456/7 123:456/7 12 34
```

Note that your node number must exist on the EchoArea line if you want Squish to toss announce messages as each message is from you to you. NEWFILE and FIDOFILE use your primary address in this example.

Assuming all the files exist in your inbound, the following will happen. Seal will create one packet from 1:234/56 (your primary) to 1:234/56 with a message in the NEWFILE echo area. This message will contain an announcement for both FIDONET.NA, NODEDIFF.A71, and ALT_NET.NA. Another packet from 1:234/56 to 1:234/56 will be created with a message in the FIDOFILE echo area announcing FIDONET.NA. A third packet will be created from 123:456/7 to 123:456/7 with a message in the ALT_NET echo area announcing ALT_NET.NA.

You can configure the look of the echomail messages. In addition, you can override the header and footer files by including the ;EchoAnn in each echomail area in which announcements are placed.

When announcing files, Seal will check for alternate origin lines for each echomail area in the files <areaname>.SQO for Squish-style areas and ORIGIN. for *.MSG-style areas.

Processing TIC Files

If you use the Tic command line parameter, Seal will process your incoming TIC files, meaning that it will toss the associated files to the appropriate directory and send TIC files to any nodes that are linked to the file areas for which TIC files were processed.

Seal will scan the TIC files in your inbound directory, verify that the associated file is also present, verify the CRC, and if all is ok, will then toss the associated file to the appropriate

directory. Seal will also update the FILES.BBS (or ListName) for non-pass-through areas.

Then Seal will create TIC files and send these files as well as the associated file to any nodes that are set to receive this file in the area that was processed.

NOTE: Seal does not support secondary areas (beyond the reading and writing of a secondary tag) or pre release or FLE file flags.

Seal will create either a file attach message or a FLO (or CLO or HLO) file according to your `Fido' setting in TIC.CFG. If Seal can't create a FLO file, then Seal will store the information needed in the hold directory so that it can try again later when SEAL TIC is run again.

As a final step, Seal will either scan your file attach messages or FLO files (according to `Fido') and compile a list of all files going out. It will then double check all pass-through files and TIC files in the hold directory and delete those that are orphans as they have already been sent out. Seal does not send TIC files as delete-when-sent or truncate-when-sent when using FLO files.

OTHER UTILITIES USING SQUISH.CFG

Seal allows you to code into SQUISH.CFG other switches that may be read by other third party utilities like SqLink.

Other utilities may read switches on the EchoArea line like SqLink's `-NL' which are ignored by Squish. However, when Seal writes your SQUISH.CFG it must be able to store these unknown switches to write them back. Seal will do just that providing you place the switches in the correct place, between the path to the area and the node address like so.

```
EchoArea MUFFIN \MSG\MUFFIN -$ -s -nl 1:234/56
```

RUNNING SEAL

Command Line Parameters

You may combine some of the command line parameters in one run of Seal. These parameters are Announce, Bad, Clean, Kill, Scan, and Tic. Any combination of these parameters are valid. Their order on the command line is irrelevant, as Seal has a hard coded order. All other parameters must be used by themselves.

Error Levels

Seal will exit with any combination of the following error levels.

1 One or more of your control files were re-written.

2 A message was created (either echomail or netmail)

4 An area was created (either file or echo)

8 A TIC file was processed successfully (including copying the new file to its destination)

Seal will combine the error levels if more than one action was done during the running or Seal. For example, if Seal created both a file area and a message, then Seal would exit with error level 6.

Environment Variable

If you want to locate the SEAL.INI file in a directory other than the directory in which SEAL.EXE is located, then you should use the SEAL environment variable.

Seal will look for its environment variable, and assume that the SEAL.INI file will be located in the directory specified. If no environment variable is set, it is assumed that the data files are found in the directory which contains the executable file (SEAL.EXE or SEAL386.EXE).

If you use the environment variable you should be sure to use full path and filenames when specifying a file name in SEAL.INI.

Sample Batch File

Here is a sample batch file using Seal, Squish is used here as the echomail processor.

```
REM Process Seal requests from your downlinks
REM Look for TIC files and create new file areas
SEAL SCAN ANNOUNCE TIC
REM Run any rescans asked for by your downlinks
CALL SQRESCAN.BAT
REM Toss new echomail
SQUISH IN OUT
REM Scan bad msg area and create new echo areas
SEAL BAD
REM Toss msgs from bad area, and pack outgoing echomail
SQUISH IN OUT SQUASH
```

IN CLOSING

Thanks

I have never taken this much time in thinking of who to thank, and for those listed here, I offer my apologies for not recognizing them sooner. My apologies also to those whom I might have inadvertently forgotten. This is not only for Seal, but anything I have written and anything I may write in the future.

First off, thanks to Scott Dudley for his most excellent BBS package Maximus and associated SquishMail. Having access to the author as Scott provides seems to be a rare thing, and something that is much appreciated.

Thanks to George Peace for AreaFix and Raid, two excellent and reliable programs that have served me well in the past years. Both programs are a good examples of what programs should excel to be. They have provided great role models for Seal in what it should do.

Thanks to Roy Pereira (author of Sqaem), and the author of SqaFix for initially tackling the problem of a Squish-compatible AreaFix, and scouting out all the functions needed for such a program so I would not have to.

Thanks to Barry Geller for creating Tick and Hatch, both of which have served me very well for years.

Thanks to Robin Rehberg for being a sounding board and much more for this and many utilities. Sometimes you just need someone to talk to.

Thanks to Edward Kuca for being the first to have faith in Seal and put it to the test, and for providing much needed feedback as to it's development.

Thanks to Jose Avelar for running Seal in its infancy and for providing a text driver which made this documentation possible.

Thanks to Raymond Beriau for translating the language file and support files to french as well as proposing new support files, and also having faith in Seal.

Thanks to Giancarlo "Vertigo" Cairella for creating an Italian language file, no easy task.

Thanks as well to Thomas Krause and Peter Beeftink who provided the German language file.

Thanks to Steven Crandall for lending a helping hand with Seal's DPMI ability.

Many thanks to all my beta testers who provided much needed and varied input as well as tolerating my constantly changing structures.

And finally, thanks to Tom Jennings for creating Fido, without which I would no doubt be struggling as a mediocre architecture student with nothing better to do than stare at buildings all day, instead of a begin computer nut.

Tech Support

Support for Seal is free and is available TUB, the SquishMail support echo.

Money & Registration

None is required. However, if you feel this utility is useful and you find it more reliable than others in its field, I urge you to

help me out and send a donation in the form of a money order or

certified cheque (Canadian funds). I can offer nothing in return except my gratitude.

Contacting the Author

You can contact the author at any of these addresses:

Fidonet:	1:243/27
Internet:	73617.740@compuserve.com
WWW:	http://ourworld.compuserve.com/homepages/presland
CompuServe:	73617,740
BBS:	(613) 236-9013, V.Everything
Snail mail:	235 Somerset St West, Apt 507 Ottawa, Ontario CAN K2P 0J3