SEAL
Squish Echo Area Link/Unlink Manager

Version 0.21

24 Apr 1993

TABLE OF CONTENTS

INTRODUCTION

What is SEAL

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

For those of you who don't know what Areafix does, it's 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.

And 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 un-linking yourself. Seal doesn't 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	The executable
SEAL.PRN	The documentation
SQSEAL.INC	Sample SQUISH.CFG section
SEAL_ENG.LNG	Sample language file (English)
SEAL_FRA.LNG	Sample language file (French)

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

INSTALLATION

Installation of Seal is as easy as blueberry muffins (if you don't get it, don't ask).

1. Make a directory on your hard drive, say C:\SEAL.

2. Copy the contents of the Seal archive to this directory.

3. Make a backup of your SQUISH.CFG file and TIC.CFG.

4. Splice the SQSEAL.INC file into your SQUISH.CFG file.

5. Edit the Seal portion of your SQUISH.CFG file to your liking. Consult the Conventions section for details on your existing file format.

6. Run SEAL FORMAT and ensure that Seal has not deleted any areas in either your SQUISH.CFG or TIC.CFG files, and has the right placement of nodes. If everything looks alright, then Seal should perform without incident from this point on.

7. Run Seal preferably upon receipt of any netmail.

That's it. If you managed to do these simple steps (I hate complicated installation procedures) Seal should work flawlessly for you for years to come.

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.

COMMAND LINE PARAMETERS

Add

This option tells Seal to create either an echomail area or a file area. The syntax for this command 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. All the normal rules for creating areas will be followed when Seal creates and area via this command. If you specify a

node address that is not listed as a feed then Seal will not create the area.

Announce

This option tells Seal to look for *.TIC and *.RAD files and create announcement echomail message for each one. This function is only operational if you have configured your TIC.CFG file in SQUISH.CFG. For more information on announcing, refer to the Announcing Files section.

Areas

This option is similar to the List parameter. When this parameter is used, Seal will create a formatted areas list file. The syntax for this command is as follows.

```
SEAL AREAS <type> <lvl> <grp(s)> <file>
```

<type> is either ECHO or FILE.

<lvl> is the security level of the areas to list.

<grp(s)> is the group letters of the areas to list in the areas list.

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

If <type> is ECHO then an AREAS.BBS type file will be produced. If <type> is FILE then a RAID type file will be produced.

Bad

Use this keyword to scan the bad message area and create echos if messages exist from valid uplink's. If an area is created that is listed in the queue file, then the downlinks listed in the queue file will be added to the new area, and a message announcing the new area will be sent to each of the downlinks. The area will also be removed from the queue file.

Describe

This option tells Seal to read a file that contains a list of echomail tags and descriptions and to try to match them with any areas you have in your SQUISH.CFG or TIC.CFG file that don't already have a description. To run with this parameter, do the following:

```
SEAL DESCRIBE <type> <description_file> [FORCED]
```

Where <type> is either FILE or ECHO. For example:

```
SEAL DESCRIBE ECHO FIDONET.NA
SEAL DESCRIBE FILE FILEBONE.NA FORCED
```

Note that Seal expects that the files you use here are in standard FIDONET.NA and FILEBONE.NA format. If you use the optional FORCED parameter as in the second example, Seal will overwrite any existing descriptions if a description is found. This is useful if your feed keeps changing the descriptions of the areas in the forward request list.

Drop

This option tell Seal to discontinue the passing of a specified area. For example, if you currently carried the POINTS echo area but wanted to stop carrying it regardless of who was linked to it, you would do the following:

```
SEAL DROP ECHO POINTS
```

Seal would then send a message to all the downlinks telling them they are no longer linked to the area, and also send an unlink message to your uplink if you are not the originator of the area (as well as delete it from your configuration file). Seal will also delete the messages and directory (in the case of *.MSG) or the message base files (in the case of .SQD).

The same thing can be done with file areas. To stop carrying the BACKBONE file area, use the following.

```
SEAL DROP FILE BACKBONE
```

Seal would then send an unlink message to all linked feeds, and a message to all other nodes that the area has been dropped. SEAL will also delete all files in the area's directory and then remove the directory only if the area was a pass-through area. You must delete the files and directory yourself if the area was not a pass-through area.

Find

This option will help you determine what areas a certain node is linked to without having to search your configuration file or wait until that node requests a status of their links. For example, to find all the areas that 1:234/567 are linked to, you would do the following:

```
SEAL FIND 1:234/567
```

Seal would then list on the screen the echo and file areas that that node is linked to.

Format

This option tells Seal to read and rewrite your SQUISH.CFG and TIC.CFG file such that all the addresses are sorted, compressed and neatly lined up. You may want to do this after you make some manual changes to your configuration file. To use this option you would do the following:

```
SEAL FORMAT
```

Help

This option works identically to the Notify option except that it sends a help message to all nodes or just the specified nodes. Refer to Notify for complete details.

Hold

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 hold all areas for all your downlinks like so.

```
SEAL HOLD
```

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

```
SEAL HOLD 1:234/56
```

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

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

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

Kill

This option tells Seal to search your netmail for messages from a valid uplink to your name at one of your address and delete them. This is handy for deleting reply messages from your uplink for areas that your system has requested in auto-forward mode.

Link

This option tells Seal to link the specified node to the specified area without checking to see if they have the proper access to do so. For example, if you wanted to add node 1:234/567 to the POINTS area then you would do the following:

```
SEAL LINK 1:234/567 ECHO POINTS
```

ECHO means POINTS is a message area, use FILE for file areas. If you wanted to add 1:234/567 to POINTS and NODES:

```
SEAL LINK 1:234/567 ECHO POINTS NODES
```

And in addition, if you wanted to add 1:234/567 to all the areas that you carry, then you could do the following:

```
SEAL LINK 1:234/567 ECHO *
```

List

This option tells Seal to create a standard ASCII text file that contains a list of areas and their associated descriptions. For example, to create a list of areas and description for all areas that nodes with access level equal to or higher than 5 and in the F group you would do the following:

```
SEAL LIST ECHO 5 F AREAS.LST
```

You may want to do this for each group that you have configured and then send this list to the nodes in the group every month. You can substitute ECHO for FILE to list the file areas. The file created only lists the tag and the description in both ECHO and FILE lists.

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

MaxCtl

Use this option to create a Maximus-style MSGAREA.CTL file based on your SQUISH.CFG file. The format for this option is as follows.

```
          SEAL MAXCTL <level> <grp> <access> <file>
```

<level> is the access level (0-255) of the areas to use.

<grp> is the one letter group character of areas to use.

<access> is the access 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 MAXCTL 0 F Normal/E FIDOMSG.CTL
```

Seal will add the following if they are specified by the appropriate SQUISH.CFG EchoArea switch.

Switch	Area definition
-s	Public 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 work correctly.

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.

Scan

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

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
```

Unhold

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

```
SEAL UNHOLD
```

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

```
SEAL UNHOLD 1:234/56
```

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

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

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

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 SQUISH.CFG UNLINK 1:234/567 ECHO *
```

COMMAND LINE SWITCHES

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

-Q

Use this switch to supress 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.

-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.

CONFIGURATION KEYWORDS (SQUISH.CFG)

All of the following should be located in your Squish configuration file. If you want to put them some place else, you don't need Seal. These can be placed anyway in the file and the order does not matter, but it is preferable if you placed all the Seal ones after the Squish ones, but before the area definitions.

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
```

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.

LogFile

This is the Squish's log file and the file that Seal will log to. What gets in the log file depends on LogLevel (see below). If no LogFile is specified, then no log file is created, although the log lines will still echo to the screen.

LogLevel

This is the Squish specification for how much information is to be contained in the log file. Seal will use the same type of logging style as Squish (unless you use ;FDLog). The default log level for Seal is 6. (This parameter was introduced with Squish v1.01 so check the appropriate documentation).

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 mesages 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.

Seal Keywords

;Announce From

Use this keyword to specify who all file announcement messages are from. This defaults to Seal. For example:

```
;Announce From Robert Presland
```

;Announce Footer

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

```
;Announce Footer FILEFTR.TXT
```

;Announce Header

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. For example:

```
;Announce Header FILEHDR.TXT
```

;Announce New

Use this keyword to add an ;Announce line to all new file areas created. To have all new file areas announced in the NEWFILES echo, use this line.

```
;Announce New NEWFILES
```

You may include more than one tag on this line as follows.

```
;Announce New NEWFILES ADMIN
```

;Announce Subject

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

```
         ;Announce Subject Look what just came in...
```

;Announce To

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

```
                 ;Announce To All File Hogs
```

;Areas EchoSpec

Use this keyword to specify an access level, a group and a description for areas listed in AREAS.BBS. The format for this keyword is as follows.

```
            ;Areas EchoSpec <level> <grp> <desc>
```

<level> is a number from 0 to 255.

<grp> is a one letter group.

<desc> is a description to appear in all linked and query lists.

An example might look like this.

```
                ;Areas EchoSpec 0 F (Fidonet)
```

;Areas NodeDim

Use this keyword to specify how many dimensions are used when writing node links in AREAS.BBS. The format for this keyword is as follows.

```
                  ;Areas NodeDim <0|2|3|4>
```

Use 0 for intelligent entries (zones and nets written when needed), 2 for two dimensional (net/node), 3 for three dimesnional (zone:net/node) and 4 for four dimensional (zone:net/node.point). Note that in all cases Seal will write the zone or net if needed.

;Backup

This keyword tells Seal to create backup files of your configuratin files prior to writing any changes to disk. The backup files will have a name of xxxxSEAL.SAV where xxxx is TIC for TIC.CFG, SQ for SQUISH.CFG, and AREA for AREAS.BBS.

;CheckName

If this keyword is used, Seal will check the name in the from field of a downlink's request against that in your SQUISH.CFG and if they are different will treat the request as unathorized. Without this keyword, 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 SQUISH.CFG

;CopyToSysop

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).

;Del

If this keyword 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 hasn't 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. And example of this would look like:

```
;Del DROPPED.DEL
```

;EchoAlias

This is any and all aliases you want Seal to recognize messages to for echomail area requests. The hard-coded aliases are AreaFix, Sqaem, EchoMgr, AreaMgr, and of course Seal. For example:

```
                    ;EchoAlias AreaManager
```

;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, don't 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.

;EchoFeed

This is a the Seal specification for one of your uplinks. Note that this keyword is preceded by a semi-colon. The format is as follows:

```
 ;EchoFeed <a> <nw> <ls> <t> <l> <g> <n> <pd> <fg> <ak> <sw>
```

<a> is the address of the uplink.

<nw> is either "Yes" or "No" and refers to whether your downlinks may forward area request to this node. If you say "No" then you can leave out the rest of the line. If however you say "Yes" you must continue with the rest of the line.

<ls> is 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 and in the format according to <t>.

<t> specifies the format of the forward list for this uplink. It can either be "A" for AREAS.BBS format or "T" for standard text. The formats are as follows:

AREAS.BBS

```
[#][$]<path> <tag> <uplink_node> [node(s)]
```

Standard Text

```
<tag> [desc]
```

<l> is 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 form 0 to 255, default being "0".

<g> is the group that is assigned to any area requestable from this uplink and is also assigned to any area created from this uplink. It can be one letter from "A" to "Z".

<n> is the name of your uplink's Seal equivalent. Seal will address all requests from your downlinks to this user at this uplink's address.

<pd> is 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.

<fg> is the series of attributes associated with this uplink. The flags supported are:

Flag	Meaning
C	Crash (msg)
K	Kill/Sent (msg)
L	Local (msg)
H	Hold (msg)

<ak> is the address to use as the origin address sending a forward message to this uplink. It will also be used as the -p address if it's different than your primary address.

<sw> are the switches to be added to the switches field in the EchoArea line of any area created from this uplink. Any remaining text after the <flag> specification will be treated as switches. The switches recognized by Seal are as follows.

Switch	Action
-s	Strip private bit
-x<node>	Read-only for <node>
-+<node>	Add <node> to SEEN-BYs

-$	Squish style area
-f	*.MSG style area
-h	Add private bit
-0	Pass-through area
-a	Add new areas to AREAS.BBS

You need not include the -p switch here as Seal will automatically work that one out. Some examples of EchoFeed are as follows:

```
 ;EchoFeed 24/5 Y AREA.LST A 0 F Afix AWAY KLC 243/47 -S -0
 ;EchoFeed 24/1 Y FIDO.NA T 0 . SQAEM AHA KLH 243/103 -S -a
 ;EchoFeed 11:230/57 N
```

;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 Intl Support Echo
```

The syntax is as follows:

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

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

<group> is the one letter group that is needed by a downlink to gain access to this area. Groups can be "A" to "Z".

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

;FakeName

Use this keyword to specify who you want Seal to pretent to be. The name specified here will be used as the From: in all message to your downlinks. You must however add it to your ;EchoAlias and/or ;FileAlias lines for Seal to respond to messages to this name.

;FDLog

Use this keyword to enable FD style logging instead of OPUS (Maximus) style logging.

;FDRescan

Use this keyword to specify a FrontDoor rescan file to create if Seal deletes a message while in KILL mode. An example might look like this.

```
;FDRescan C:\FD\FDRESCAN.NOW
```

;FileAlias

Use this keyword to define names which Seal will recognize as messages regarding file areas. The default is SEAL FILES and RAID.

;FileFeed

This is a the Seal specification for one of your file uplinks. Note that this keyword is preceded by a semi-colon. The format is as follows:

```
FileFeed <a> <nw> <ls> <t> <l> <g> <n> <pd> <fg> <aka>
```

This operates in the exact same way as ;EchoFeed with the following exceptions.

<t> specifies the format of the forward list for this uplink. It can either be "R" for RAID format (FILEBONE.NA) or "T" for standard text. The formats are as follows:

RAID

```
Area <tag> <level> <flags> <desc>
```

Standard Text

```
<tag> [desc]
```

<fg> Specifies the flags for this file feed. The flags supported are:

Flag	Meaning
C	Crash (msg/tic)

H	Hold (msg/tic)
L	Local (msg)
K	Kill/Sent (msg)
P	Pre-release (tic)
T	No TIC file (tic)
F	FLE file (tic)
*	Accept (tic)
&	Don't send (tic)
0	Create pass-through areas (tic)

All flags lised as (msg) are used when creating messages, those with (tic) are used when adding this node to TIC.CFG. Seal will add the * to a feed's entry in TIC.CFG so you need not include it here.

There are no switches for a file feed. An example would look like this.

```
 ;FileFeed 24/5 Y FILE.NA R 0 F Raid AWAY KLC& 243/47
 ;FileFeed 24/1 Y FIDO.NA T 0 . Raid AHA  KLH0 243/103
 ;FileFeed 11:230/57 N
```

;ForceINTL

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.

;GroupLines

Use this keyword 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.

;Hold

This is the name of Seal's hold file in which it will store areas (both echo and file) that are placed on hold by use of the %HOLD token or HOLD command line parameter. An example might look like this.

```
                   ;Hold G:\SEAL\SEAL.HLD
```

;KillProcessedMsgs

If this is used, Seal will delete messages that have been received by Seal, otherwise, Seal will just mark them as received.

;LanguageFile

Use this keyword to define a language file with which to override some of the message texts or rewrite the entire outlook of Seal. This file wil be used for the creation of all messages unless overriden by a language file specification on the ;NodeLink line.

;MarkNotRcvd

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.

;MatchZone

This keyword 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 keyword.

;MaxCtl Area

Use this keyword to specify a template for the area name when using the MAXCTL command line parameter. The default is as follows.

```
;MaxCtl Area %#
```

Valid tokens are as follows.

Token	Replaced with:
%T	Full tag

%t	Tag without vowels, underscore, period or hyphen
%#	Incrementing number
%g	Group letter

Note the the tokens are case sensitive. For example, to sepcify area names F1, F2, F3, F4, etc. when the group letter is F.

```
;MaxCtl Area %g%#
```

;MaxCtl Extra

Use htis keyword to specify extra lines you want to appear at the end of the Maximus MSGAREA.CTL area definitons. For example, if you wanted to add `Disgrace ReadOnly' to all your areas, you would specify:

```
;MaxCtl Extra Disgrace ReadOnly
```

You may have as many of these such lines as you wish. You should not include any lines that are covered in the EchoArea switches here.

;NewEchosPath

This is where Seal should create all new areas. For example:

```
;NewEchosPath \MSG\PASSTHRU
```

You may also specify different paths for each group you create. For example, if one of your uplinks has been specified as group F (for Fido) and you want all areas created by this uplink to be placed in \MSG\PASS\FIDO then use the following:

```
;NewEchosPath Group F \MSG\PASS\FIDO
```

If an area is created and a group path is not defined, the default will be used, and if there is no default, it will be created in the current directory.

;NewFilesPath

Same as ;NewEchosPath except for file areas. See ;NewEchosPath for complete details.

;NodeAka

Use this keyword in combination with ;NodeLink to fully qualify a downlink. If one of your downlinks has more than one address and can't seem to quite remember what address to use when talking to Seal on your systsm, 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. The format is as follows.

```
                 ;NodeAka <primary> <aka(s)>
```

A sample line might look like this.

```
          ;NodeAka 1:234/56 57 45:23/8 12 127:12/34
```

This means that seal will treat requests coming from 1:234/57, 45:23/8, 45:23/12, and 127:12/34 as being from 1:234/56. Reply messages will always be sent to the primary address (in this case 1:234/56).

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

;NodeLink

This is the Seal specification for one of your downlinks. Note that this keyword is preceded by a semi-colon. The format is as follows:

```
 ;NodeLink <addr> <lvl> <grps> <pass> <name> <flags> <lang>
```

<addr> is the node address of this downlink (in at least 2D format.

<lvl> is the security level to assign to this downlink. It should be a number from 0 to 255. The default level (no security) is "0".

<grps> is a series of letters that refer to the group that this downlink belongs to. Each group is a single letter from "A" to "Z" and is case-sensitive. Specify more than one group by stringing them together without spaces. To specify no security, use a period (.).

<pass> is a one word password that should appear in the downlink's message subject line for the request to be valid.

<name> is the name of the sysop of the downlink. Only request from this name at the node address with a correct password will be honoured. Use _ instead of a space if the name is more than one word.

<flags> is the message attributes you want Seal's response messages to have. Flags supported are:

Flag	Meaning
C	Crash (msg/tic)
H	Hold (msg/tic)
L	Local (msg)
K	Kill/Sent (msg)
P	Pre-release (tic)
T	No TIC file (tic)
F	FLE file (tic)
*	Accept (tic)
&	Don't send (tic)

Flags with a (msg) are used when creating messages, those with (tic) are used when adding the downlink to a file area.

<lang> is 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.

Some example NodeLink lines would look like this:

```
 ;NodeLink 123/456   2 FA MAGIC  John_Q_Public KLP   ENG.LNG
 ;NodeLink 234/23.2  0 B  WONDER John_Smith    KLCT*
 ;NodeLink 12:345/6 34 GA GAME   Hell_Raiser   LHKF  FRA.LNG
```

;No Help

Use this is you want to disable the -H (or language file equivalent) switch by the downlink on the subject line.

;No Query

Use this switch 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.

;No Rescan

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.

;NoDesc

Use this keyword to specify a description for all echo and file areas that don't already have one. An example might look like this.

```
;NoDesc (no description available)
```

;Notify Exclude

Use this keyword to specify downlinks you do not want notified when the NOTIFY command line parameter is used. For example, you can exclude 123/456 from a SEAL ... NOTIFY by including the following in SQUISH.CFG.

```
;Notify Exclude 123/456
```

;Notify List

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

;Open

Use this keyword 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 using ;Open Groups and ;Open Level and send a reponse message. Under an open system, all query and linked request messages will be honoured although you cannot notify systems not in a NodeLink line.

;Open Groups

Use this keyword to specify an access level for open system requests. Consult ;Open for a description of an open system. An example is as follows.

```
;Open Groups FA
```

;Open Level

Use this keyword to specify the group letters for open system requests. Consult ;Open for a description of running an open system. An example is as follows.

```
;Open Level 15
```

;PointNet

Use this keyword if your TIC.CFG file requires two or three dimensional addresses. Seal will convert all your point address to a pointnet address before processing the TIC.CFG file. Note that this is already in your SQUISH.CFG file, either uncommented or commented. To hide it from Seal you must use two semi-colons like so.

```
;;PointNet 30933
```

;ProtectArea

Use this to protect either a file or echo area that is a legal forward request but is protected. For example, if REG12 did not allow points, and you did not carry REG12, but it was in your uplink's forward request list, you would want to potect it so that your points cannot link into it. The format of this keyword is as follows.

```
;ProtectArea <type> <tag> <level> <grp>
```

<type> is either ECHO or FILE.

<tag> is the tag of the area you wish to protect.

<level> is the access level (0-255) which you want assigned to this area.

<grp> is a one letter group character to assign to this area.

An example would look like this.

```
;ProtectArea Echo REG12 50 F
```

;Que

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. An example would look like this:

```
;Que FORWARD.QUE
```

;QueryFile

This is where you specify a text file to be used instead of the generated list that Seal creates from your configuration file. This is used in response to a -Q (or language file equivalent) request from one of your downlinks. An example would look like this:

```
;QueryFile AREAS.LST
```

This is handy if you want a more verbose description of your areas, or you just want to put pretty pictures among your areas.

;Rescan

This is where you specify the location of the batch file that Seal will produce to enable rescans. Seal will write a line for each area that needs rescanning in this file like this:

```
SQUISH RESCAN <tag> <node>
```

This file is deleted if found before response mode is enabled. An example would look like this:

```
;Rescan \SQUISH\SQRESCAN.BAT
```

;Respond Linked

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

;Respond List

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

;ShortAreaList

Use this keyword to 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.

;Show Feeds

This keyword 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.

;Show Protected

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).

;Sysop Name

This is your name, like this:

```
                 ;Sysop Name Robert Presland
```

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

;Sysop Flags

This is similar to the NodeLink's flags, but are used for messages to you as the sysop. Flags supported are Crash (C), Hold (H), Local (L), and Kill/Sent (K). All message are always private.

;Tick

Use this keyword if you want to enable the Tick processing mode of Seal. In this mode, Seal will respond to messages addressed to SEAL FILES or RAID.

CONFIGURATION KEYWORDS (TIC.CFG)

;Announce

Use this keyword that files processed in this file echo are to be announced in the specified echos. 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 don't want files in this area to be announced.

;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> <group> <desc>
```

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

<group> is the one letter group that is needed by a downlink to gain access to this area. Groups can be "A" to "Z".

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

REQUEST FORWARDING

I'm not going to go into detail about how this works cause I really don't want to [grin]. What I will say is how Seal operates (briefly).

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. 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 don't get created by a bad message arriving before the unlink message is received by your uplink.

USER INSTRUCTIONS

No sense in duplicating the user instructions so please refer to the SEAL_ENG.LNG file as distributed with this utility. Briefly, here they are:

Subject switch	Message token	Description
-Q	%LIST	List of areas available
	%QUERY	Same as -Q
-R	%RESCAN	Rescan of areas in msg
-L	%LINKED	List of linked areas
-H	%HELP	Help file

Your downlinks may also change their password by using %PASSWORD and specifying a new password, however their old password must still appear in the subject field when changing to the new password.

Your downlinks may also request both file and echo areas in the same message by using %ECHO and %FILE tokens in the message. For example.

```
%ECHO
MUFFIN
TUB
%FILE
NODEDIFF
FIDONEWS
```

This list of areas will result in the processing of echo areas MUFFIN and TUB and file areas NODEDIFF and FIDONEWS.

A downlink may also place all their areas on hold by using the %HOLD token in a message. If used, this token will cause Seal to unlink the downlink form all their areas and place the names of the areas in the hold file. When a %UNHOLD token is used later, Seal will relink the downlink to the areas previously held by reading the hold file. The same can be accomplished by using the HOLD and UNHOLD command line parameters.

All tokens (%<word>) can be overridden by use of a language file.

To address a message to Seal regarding file areas, address the message to "SEAL FILES" (sans quotes) and regarding echos as "SEAL ECHOS". Seal will also recognize "RAID" as the former and "AREAFIX" as the latter. Messages addressed to "SEAL" are the treated as "SEAL ECHOS".

Refer to SEAL_ENG.LNG for more complete instructions.

TICK SUPPORT

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 enable the ;Tick keyword in your SQUISH.CFG file, 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 enable the ;Pointnet keyword in your SQUISH.CFG (note that this keyword is already in SQUISH.CFG so if you want to comment it out for Seal you must use two colons, i.e. ;;POINTNET), Seal 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 for each file processed.

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 one series of packets from any TIC files, and one seriesof packets from RAD files and leave them in your Squish NetFile directory for Squish to toss. Each series will contain as many packets as there are echomail tags to announce files in. 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 addres 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 NODEDIFF.A71. 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 by using the ;Announce* keywords. In addition, you can override the header and footer files by including the ;EchoAnn in each echomail area in which announcements are placed.

MULTI-PURPOSE OPERATION

When Seal is run in both modes (echo and file) all linked messages as well as query messages will include the appropriate list of both echo areas and file areas. Notify message will also include both lists. However, if you use a ;QueryFile specification, then only that file will be used as the message content -- *no* lists are generated.

LANGUAGE SUPPORT

You can configure Seal to display any language you desire through the use of an external language file. This file may contain all text that is displayed in messages to your downlinks. 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 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 a language file.

You specify a language file like so in SQUISH.CFG.

```
                    ;LanguageFile SEAL_ENG.LNG
```

The language file itself is made up of any number of lines, each with a keyword as the first word, and the rest of the line being the message text corresponding to the keyword. In addition, 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. Refer to the distribution SEAL_ENG.LNG 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, the the default language file will be used. If there is no default specified or it cannot be loaded, then the hard coded default wil take effect.

Also contained in the language file are a number of sections of message texts. They are as follows.

The notify header text appears at the top of notify messages. The help message text is copied to the help message when your downlinks uses -H (or the language file equivalent). The no access message text will be used in a message to a downlink (or any node) that is not authorized to use Seal.

Each text section is delimited by a header line and a footer line. Every line inbetween these lines is considered text and will be copied verbatim. The structure will look like this.

```
     NotifyHeader
     ...text...
     End

     HelpMsg
     ...text...
     End

     NoAccess
     ...text...
     End
```

Consult the distribution language files for a more complete example.

ENVIRONMENT

To simplify the command line of Seal, Seal will look for the Squish environment variable to point to your SQUISH.CFG. Following is the logic applied when trying to find your SQUISH.CFG.

1. Will read the SQUISH environment variable to find your SQUISH.CFG file, if no SQUISH environment variable...

2. Seal will look for SQUISH.CFG in the current directory.

3. Seal will use the file specified with the -c switch.

To set the SQUISH environment variable, do something like this.

```
SET SQUISH=G:\SQ\SQUISH.CFG
```

RUNNING SEAL

You may combine some of the command line parameters in one run of SEAL. These parameters are Scan, Bad, Kill, and Announce. Any combination of these parameters are valid. All other parameters must be used by themselves.

When you run Seal, it will look for SQUISH.CFG just like Squish does. Refer to the previous section on the Squish environment variable for more details.

SAMPLE BATCH FILE

Here is a sample batch file using Seal in all it's modes. Squish is used here as the echomail processor, and Tick as the file echo processor.

```
REM Process Seal requests from your downlinks
REM Look for TIC files and create new file areas
SEAL SCAN ANNOUNCE

REM Run any rescans asked for by your downlinks
CALL SQRESCAN.BAT

REM Process incoming TIC files
TICK TIC.CFG >> TICK.LOG

REM Toss new echomail
SQUISH IN OUT

REM Scan bad msg area and create new echo areas
SEAL BAD

REM Add descriptions to any new areas created
SEAL DESCRIBE ECHO FIDONET.NA
SEAL DESCRIBE FILE FILEBONE.NA

REM Toss msgs from bad area, and pack outgoing echomail
SQUISH IN OUT SQUASH
```

HISTORY

0.21	Added registration keys. Fixed a bug that would create message after message if descriptions were included and an area in the message didn't have a description. Fixed a bug when using open system that would send the reply to 65536:65536/65536.65536. Changed the protected char to a left and right char (default=<>). ;ShortAreaList now applies to linked list. Seal will no longer reply to messages when the To: field is blank. Added %HOLD and %UNHOLD and all that it entails. Fixed a bug that would write A0 instead of A1, A2, A3, etc in TIC.CFG.
0.20	Added another line to the announce text block, and %x for the `replaces' file name. Will show file feeds in linked area lists. Will now only include areas that a file feed feeds (* present in switches) in relink messages instead of just being linked to the area. When creating a file area, all standard flags will be added to the feed's switches as is now with a NodeLink.

FileFeed's flags are the same as a NodeLink's. Will now delete all lines associated with a deleted file area instead of just the AREA line. Changed some keywords as follows. Added `MaxCtl Extra' for extra lines in the MSGAREA.CTL area definition. Deleted AddNewNoPassthrough and AddNewAreasBBS. Removed auto upgarde of previuosly changed keywords. Can now specify any combination of ANNOUNCE, SCAN, KILL, and BAD on one command line. Changed the help screen to reflect this. Added -c for the alternate config file. Rewrote the internal storage of everything. Limits are according to available memory. Announce header and footer files are no longer supported in TIC.CFG. Will kill the current announce text line if %x was specified and no replaces file is present, or if %d was specified and no description is present, or %p was specified and no path is present. Added variable tag length, will adjust to the maximum needed tag length in all list of tag outputs like messages, lists, etc. Added a pause in the FIND output. Changed the format of the config files, ;EchoSpec, ;FileSpec, and ;Announce lines will be placed after all the EchoArea (Squish) and Area (Tick) lines. Deleted ;DefaultEchoSpec and ;DefaultFileSpec keywords. Added `;Areas EchoSpec <level> <grp(s)>' for areas in AREAS.BBS. Echo and file areas with no specs will receive `0 .', and this will be written to the config file next write. Added `;NoDesc' for no echo or file description (won't be written to config files). Added `Areas NodeDim <0|2|3|4>' for dimension of nodes in AREAS.BBS. Added `;FDRescan' for use with KILL. Added `;GroupLines' for choice in how Seal writes your config files. Added `;Respond List' and `;Respond Linked'. Tags, paths, descriptions, and announce tag lines are now limited to 255 characters without loss of memory. Added `;EchoAnn <tag> <hdr_file> <ftr_file>' (formerly ;AnnounceHdr and ;AnnounceFtr). Changed manual change ALL tag to *.

0.19 (Internal release) Will no longer create areas from echo feeds that have No new areas create. Added ;AddNewNoSquish. Changed Squish switch processing and internal storage. Will now interpret all switches instead of copying them verbatim. Added ;FDLog for FD log styling. Fixed a bug that sometimes would not create new areas (echo or file) as non-passthrough. Added `;MaxCtl Area' as an area name template for use with MAXCTL. Will now search through a node's akas for matching NodeLinks and if found, will use that address for link/unlink request for each tag in question. When using UNLINK, wil no longer log every area, but rather only the areas that the node is linked to. Added routine to ensure that new echo and file paths are unique up to 99 `what would normally be the same' paths. Added ;OpenSystem, ;OpenGroups, and ;OpenLevel for operating an open Areafix/Raid system. Added support for multiple netmail areas. Added KILL command line parameter for killing messages from your file and echo uplinks.

0.18 Fixed a bug that would not match a 2D feed in a bad *.MSG area.

0.17 Will now log to Squish's LogFile Spec if present. LogLevel will dictate what gets into the log. Will now create 2+ packets instead of 2 (stoneage). Changed some of the log lines. Added ;Backup. Added ;ProtectArea. Added MSG for LIST command line parameter. Will now only rescan ECHO areas in the current message if they result in a "Linked" or "Already linked" status. Added NotifySubject language spec line. Added %ECHO and %FILE tokens for message processing. Combined OVERDESCRIBE and DESCRIBE. Added support for +tag in request msg. Will now only delete files in a file area if that file area was passthrough. If using DROP command line parameter on a non-passthrough area, you must delete the files and directory yourself. Fixed a bug that would cause Seal to open stdin as a forward list. Changed "EchoAreaFeedChar" to "AreaFeedChar" in language file. Will no longer scan the bad messages in auto-response mode -- use SEAL BAD. Changed ;NodeFeed to ;EchoFeed, and ;NewAreasPath to ;NewEchosPath, and removed ;ScanBad. Same thing with ;Alias, changed it to ;EchoAlias, and ;TickAlias to ;FileAlias. Added ;FakeName to pretend that it is something other than SEAL in reply messages.

0.16 Added AREAS command line parameter. Changed formatting of AREAS.BBS lines when written. Added an origin line back to all messages. Announce messages will now contain all files in that area instead of one message per file, also one packet will be created for each echomail tag that announcement messages are created in. SEAL will also produce type 2 packets instead of a stoneage one. Added HELP command line parameter. LIST now creates a simple list file for files. AREAS creates a Raid-style file list. Split the EXE into overlays, should use less memory. Will also try and load the overlays into EMS if possible. Will no longer "loose" secondary file area tags when using FORMAT. Added support for Tick's pre-release (P) flag. Fixed a mite when sorting nodes on the EchoArea line. Changed the command line processing logic. Added support for the SQUISH environment variable. The banner will use BIOS and not direct screen writes. Added ;NodeAka.

0.15 Corrected a spelling error when reading YouAreTheFeed and AreaAdded in the language file. Added language support on a node basis. Moved notify header file, help message file and no access message file to the language file. Copies of notify messages and uplink request messages will not be sent to the sysop when ;CopyToSysop is used. Added ;AddNewNoPassthrough File, and ;AnnounceNew. Will now create the new directory when creating a new file area. Will unlink you from a file area if it's passthrough and all your downlinks have unlinked as well as deleting the directory and all files in it. Corrected reading of the flag characters (now ignores case). Will exit with erorr level 1 if a message was processed, 0 otherwise. Added & and * to node flags for adding to Tick areas. Added ADD command line parameter.

0.14 Added split netmail message capability, added ContinuedNextMsg and ContinuedPreviousMsg to language file. Added langauge file entry for day and month names. Added ;CheckName, by default won't check the name on downlink's request, but still sends to the name configured in SQUISH.CFG. ;CopyToSysop will now copy all messages created except query and help messages. Will no longer add the default description to the ;EchoSpec line but will display it in messages. Added ;DefaultFileSpec. Removed MSGID: and PID: from file announcement message (FD thought it was netmail). Will no longer default to zone 0 but rather your zone in the case where there is no MSGID: line in request messages. Fixed processing of incoming INTL lines.

0.13 Changed internal storage of links, should enable more link storage in the future. Changed DESCRIBE and OVERDESCRIBE to include file areas. Groups are now case-sensitive ("a" is not the same as "A") so that gives you 52 different groups. Added request forwarding and creation of Tick areas, ;FileFeed. Changed RELINK to include file areas. Added ;NewFilesPath (and group option).

0.12 Added ;ShortAreaList. Added tokens to the language file. Finally added deletion of a Squish-style netmail message -- now is fully Squish compatible. Fixed a small error where SEAL forgot to replace the %a in the queue seciton of a area list. Changed the queue file to a plain ASCII file that can be edited. Fixed an error that would produce a strange reply message if any of a uplinks' forward lists could not be read.

0.11 18 Feb 93; Changed the TIC reading routine so it works with AllFix's TIC files too. Will no longer add an origin line (only a tear line) to netmail msgs. Added the Group option to NewAreasPath. Added ;NotifyExclude. Fixed a bug that would write level 0 to all file areas on a FORMAT command. Added full language support via an external language file. Made the downlink's name check case-insensitive. Fixed a bug that would add areas more than once, and not add the requesting downlink to a newly created forward requested area upon arrival.

0.10 Added ;SysopFlags, same as NodeLink flags, Private is always assumed, default is Local and Private. Added confirmation of a password change in the response message. Added confirmation of a rescan, or notice of rescans disabled. Will no longer send empty messages if nothing was done. Added ANNOUNCE, add an ";Announce <area_tag>" at the end of each TIC area. Added ;AnnounceFtr, ;AnnounceHdr, ;AnnounceTo, ;AnnounceFrom, and ;AnnounceSubject. Will now announce .TIC and .RAD files. Added ;AddNewNoPassthrough. Added ;AnnounceHdr and ;AnnounceFtr to TIC area definition. Added ;NoAccessFile. Added ;NoSortAreas.

0.09 8 Feb 93; Uses dynamically allocated message buffers to decrease stack overflow errors (Runtime error 202). Sorts area tags in response messages. Incompatible with the old queue file, so make sure your queue file is size 0 before installing 0.09, or delete the queue file.

0.08 Now creates one second Unix style MsgIDs and real date/time stamps. Hopefully this won't cause some tossers to declare dupe. Added the ability for the downlink to change their password. Now uses even better text file buffers -- much faster. Fixed an unknown (until now) bug that would write the DOS date time stamp as time-date instead of date-time. Added ;MatchZone. Fixed bug in rescan file, now writes multiple lines instead of just one. Added OVERDESCRIBE command line option. Added the Del file. Fixed an error in the help screen. Added Squish-style message base support. Will now add the Ax switch to your TIC.CFG whenever the file is written for all nodes if needed to ensure that the list of Akas match the Ax switches. Will only store the needed Akas in your TIC.CFG instead of every aka listed in SQUISH.CFG. Corrected an error when displaying the "Msg for existing ...area..." message.

0.07 18 Nov 92; Corrected a few documentation omissions. Now deletes messages when a drop is issued. Will now report "not enough arguments" in all applicable cases. Added ;ShowFeeds. Will now create messages in memory and then write them to disk instead of writing them on the fly. Seems faster this way. Better code resulting in a smaller executable. Increased memory usage to accommodate internal messages, shouldn't cause problems unless you have 1000s of areas. Better dealing with extra areas (areas that SEAL couldn't load into memory. Now unallocates memory used by a dropped area. Fixed a bug that would introduce an extra line in your config files. Removed environment variables. Added Tick's T and F flags in ;NodeLink. Won't search for nul tags when various end of message methods are used.

0.06 10 Nov 92; Changed error display when not enough arguments on the command line. Revamped the way SEAL does linking/unlinking to/from all echo/file areas. Corrected a spelling error in "Message for existing area ...". Will now ignore messages to SEAL FILES if not in Tick processing mode. Revamped LIST to support file areas and "as documented" level.

0.05 8 Nov 92; Fixed a bug that would insert a LF in occasionally. Added Tick support using TIC.CFG and associated ;Tick, ;TickAlias, and ;Pointnet keywords. SEAL will now automatically add an ;EchoSpec line in both SQUISH.CFG and TIC.CFG whenever writing the files instead of only if one already exists. Change command line syntax to enable use of echo and file areas.

0.04c Transferred file I/O code to a unit.

0.04b Improved file reading and writing for SQUISH.CFG and AREAS.BBS. Added ;QueryFile to SQUISH.CFG. Added deletion of .SQB (or .MSG) when deleting a passthrough area.

0.04a Fixed a bug when writing new areas to config file.

0.03 31 Oct 92; Fixed a bug when writing the rescan batch file. Changed the way SEAL reads the forward lists. Finally found the bug that would cause SEAL to hang when responding to a request message. Included the <aka> in the docs for the ;NodeFeed line (forgot to put it in there, although the example shows it, it goes between the flags and the switches).

0.02	29 Oct 92; Fixed a mite when replying to an area that does not exist. Added Squish style logging. Changed status message structure. Changed name spec for log file. Changed algorithm for calculating MsgID. Added coded date in message. Added support for Squish's log level spec. Added file sharing capability. Added better file reading capability. Added % type things. Added support for addresses on one line and aliases. Added RELINK.
0.01	23 Oct 92; First release to beta testers for evaluation.

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. 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 wouldn't have to.

Thanks to Barry Geller for creating Tick and including all the wanted options so that I didn't have to add them to Seal.

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.

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.

SUPPORT

Support for Seal is free and is available through TUB, the Squish support echo. Netmail is always welcome as well.

MONEY & REGISTRATION

None is required. But this starts a new era for me. This is the first utility that I will begin accepting donations for. 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 small donation in the form of a money order or certified cheque (Canadian funds). I can offer nothing in return except my gratitude.

BUGS, SUGGESTIONS & COMMENTS

All of the above should be sent to:

Robert Presland (1:243/27)
60 Daly Avenue, Suite 604
Ottawa, Ontario CANADA
K1N 6E5