SNC 1.0: The Stickies! Note Compiler

Introduction

What's a note compiler, and why in the world would you want one? The Stickies! Note Compiler, SNC, is a freeware Windows program from Looking Glass Technologies that you can use to create Stickies! export files from plain text files, while still retaining full control over each note's configuration, including such settings as color, initial screen position, size, alarm and expiration settings, linked documents, etc. The files you create with SNC can then be imported into your Stickies! active set or file cabinet, or sent to other users.

SNC can be particularly useful if you regularly need to translate non-Stickies! data into notes, and you want to determine exactly how the notes are created. For example, you can create a stock template for translating the notes, and when the text you want to compile into notes becomes available, you can "pour" it into the template and compile it to make a set of notes for your own use or for distribution within your company.

SNC is also a handy tool for those who want to create their own Stickies! Stacks, export files of reference material on some general topic, such as foreign countries, the chemical elements, famous people you think you've spotted in Central Park, or any other information you'd like to convert into Stickies! notes.

SNC is normally distributed as a DOS-mode, self-extracting archive that was created with LHA 2.13. LHA is copyright 1988-91 by Haruyasu Yoshizaki.

Overview

The process of compiling a text file into a Stickies! export file is very simple. You create the input file with any editor that will handle plain ASCII files, and then you run SNC and give it the name of the input file and the name of the Stickies! export file you want to create, and away it goes. There are no command line options, no twisty little menus all alike for you to get lost in, etc. (In the best tradition of computer designers everywhere, we wanted to make SNC big, complex, and confusing, but we just couldn't bring ourselves to do it. We'll do better in the future, honest. In all seriousness, we did try to keep this first release of the compiler as simple as possible without compromising its usefulness. Still, we're eager to hear about changes you'd like to see in SNC, and we'll certainly take them into account as we continue to develop the program.) SNC will issue warnings and error messages as appropriate, and create the export file, assuming it can interpret your input file. Once the Stickies! SNC has translated your input file, you can delete it if you wish, since it's not needed by Stickies!.

SML Language Reference

The language that you use to write your input file is called SML: Stickies! Markup Language. It's a very simple tag-oriented language that we're sure you'll pick up in no time. The basics of SML are as follows:

* Each statement in your input file must be the first non-blank item on that line, although the statements need not begin in column 1.

* All statements begin with a colon (:).

There are two kinds of statements: directives, which give the compiler some instruction on how to perform the translation (e.g. the :force_ver20 directive makes the compiler generate an export file in version 2.0 format, even if the notes you've defined don't use version 2.0 features), and commands, which give the compiler information about the notes you're creating. Most commands also require you to provide some sort of data, such as the text to use as a note title or the expiration date you're setting.

* Statements and their data can be in any combination of mixed case. The only time case matters is when you're providing some data that Stickies! will ultimately see, such as the title of a note or the text in the note itself. Then, SNC will pass along your data exactly as you provided it.

* Any line that is not part of the note's body of text (the part you would normally type or paste in if you were creating the note in Stickies!) and does not begin with a statement is assumed to be a comment and is ignored. *There is no comment delimiter in SML.* This is unusual for computer languages, but it allows your source files to be quite free-form, and it encourages comments. See SML example 1 for a sample of how the source looks. To help you find problems in your code, such as typos in statements, SNC will issue a warning for any line in which the first non-blank character is a colon.

* SML is very forgiving about things like blanks and blank lines, but you do have to keep a command and its data on the same line. For example, if you want to use the :title command, you can do the following:

```
:title This is a Cool Note!
```

but not the following:

```
:title
This is a comment!
```

In the second example, SNC would complain about the lack of a title on the :title command, and it would assume that the second line of text was a comment.

This release of SNC does not recognize tabs as blanks! If you want to use tabs to separate commands and their data, then you must have your editor convert the tabs to blanks when it writes your file. If your editor doesn't have this feature, then you must manually insert blanks to separate non-blank strings in your file.

* SNC has two levels of diagnostic messages it issues: warnings and errors. Warnings display a message but don't stop the translation process; errors display a message and make SNC stop processing your file. There is a pair of directives that allows you to turn warnings on and off, but errors are always enabled.

* You can set default settings for notes and then let all notes defined after that point in your file "inherit" those settings. This is a nice shortcut, in that it allows you to specify what color you want your notes to be without having to use the color commands on every note, for example. You can change the defaults from place to place in your file, if you want to use different settings for various blocks of notes. Each default section's settings stay in effect until the next default section in the file is encountered. The default sections are optional, though, and you can simply define all settings for each note in isolation, if you want.

* Nearly all statements can be repeated as many times as you like, and only the last one will take effect. For example, if you are defining a note and you have the following lines in your file:

```
:title Hello, World!!!
:title Fred's Notes
```

Then just the second title will be used. SNC does not issue a warning in such cases.

SML: Structure of a Source File

In general, your source file should look like this:

```
First, let's set our defaults for our notes

:begin_defaults
Default settings go here
:end_defaults

Now, let's define the first note

:begin_note
Note settings you want to override go here

:begin_link
Want linked documents in your notes?  This is where you set them up,
with one link block per link.
:end_link

:begin_text
Hey!!! This will wind up in the note!!!
:end_text

:end_note
```

The model here is that almost all statements appear on a single line. Multiple-line commands use a begin_X/end_X pair of lines, with additional commands between the begin/end pair. Throughout this manual, we'll refer to a begin_X/end_X pair of lines and their intervening commands as an X block, e.g. note blocks, defaults blocks, text blocks,, and link blocks.

Note that some commands can only appear in some places-- you can't define a link outside a note block, for example. If you put a statement where it shouldn't appear, SNC will consider it an error. Directives, by contrast, can appear anywhere.

One useful trick is to use a line of underscore characters to separate note blocks from each other, as we did in the sample SML files. This doesn't slow down the compiler, and it improves the readability of the file.

SML: Statement Reference

This section is the one where we get down to brass tacks (whatever that means). Here we provide an alphabetic listing of all the SML statements. We suggest that you consult the example programs that came with this package, since we exercised all the options we could and tried to demonstrate good SML practice in writing them.

See the section following this one, SML Data Formats, for a thrilling discussion of how SNC expects to find various types of data in your input file.

:ALARM

Type: Single-line command
Data: Boolean
Placement: Defaults or note block
Default setting: false/no/nope/0/n
Purpose: Sets the note's alarm.

Notes: If you set a note's alarm flag and don't provide an alarm time or other alarm settings, the note will inherit these other alarm settings, either the ones you set in a defaults block or the Stickies! defaults.

Using any of the other alarm commands will *not* set a note's alarm flag. You must still use the :ALARM command to enable the alarm.

Example:

```
:begin_note
:alarm true
:alarm_time 6/12/1993 9:0
:alarm_repeat no
:alarm_prefix 15
:alarm_daily nope
:alarm_sound false
(Rest of note definition goes here)
:end_note
```

See also :ALARM_DAILY, :ALARM_PREFIX, :ALARM_REPEAT, :ALARM_SOUND, :ALARM_TIME

:ALARM_DAILY

Type: Single-line command
Data: Boolean
Placement: Defaults or note block
Default setting: false/no/nope/0/n

Purpose: Sets the note's alarm's daily flag.

Notes: This is a Stickies! 2.0 feature, and Stickies! 1.5 will ignore this flag's setting.

See also :ALARM
(example), :ALARM_PREFIX, :ALARM_REPEAT, :ALARM_SOUND, :ALARM_TIME

:ALARM_PREFIX

Type: Single-line command
Data: Integer, must be in the range 0..99
Placement: Defaults or note block
Default setting: 15
Purpose: Sets the note's alarm's prefix time (the number of minutes before the alarm time that the alarm actually sounds).

See also :ALARM
(example), :ALARM_DAILY, :ALARM_REPEAT, :ALARM_SOUND, :ALARM_TIME

:ALARM_REPEAT

Type: Single-line command
Data: Integer, must be in the range 0..99
Placement: Defaults or note block
Default setting: 0
Purpose: Sets the note's alarm's repeat time (the number of minutes between soundings once an alarm goes off for the first time).

Notes: This is a Stickies! 2.0 feature, and Stickies! 1.5 will ignore this setting.

See also :ALARM
(example), :ALARM_DAILY, :ALARM_PREFIX, :ALARM_SOUND, :ALARM_TIME

:ALARM_SOUND

Type: Single-line command
Data: Boolean
Placement: Defaults or note block
Default setting: true/yes/yep/1/y
Purpose: Sets the note's alarm sound flag.

See also :ALARM
(example), :ALARM_DAILY, :ALARM_PREFIX, :ALARM_REPEAT, :ALARM_TIME

:ALARM_TIME

Type: Single-line command
Data: Date and time (see SML Data Formats section for details)
Placement: Defaults or note block
Default setting: 12/31/2001 9:00
Purpose: Sets the note's alarm date and time

See also :ALARM
(example), :ALARM_DAILY, :ALARM_PREFIX, :ALARM_REPEAT, :ALARM_SOUND

:ALWAYS_ON_TOP

Type: Single-line command
Data: Boolean
Placement: Defaults or note block
Default setting: false/no/nope/0/n
Purpose: Sets the note's always on top flag

Notes: This flag is only respected by Stickies! under Windows 3.1, since the always on top feature was not available in Windows 3.0.

:BEGIN_DEFAULTS

Type: Multi-line command
Data: None
Placement: Cannot appear in defaults or note block
Purpose: Marks the beginning of a defaults block

Notes: All settings in a defaults block become the new defaults for all notes defined in the input file after that point, until the next defaults block is encountered. You can have as many defaults blocks as you want in a file. See some of the sample SML files for examples of how defaults blocks work.

Example:

```
:begin_defaults
Set our default colors to something dramatic
:bg_color black
:fg_color red
:end_defaults
```

See also :END_DEFAULTS

:BEGIN_LINK

Type: Multi-line command
Data: None
Placement: Can appear only in a note block
Purpose: Marks the beginning of a link block

Notes: You use one link block in a note block for every linked document that you want that note to have. See the following example and the sample code that came with SNC for usage. If one or more of your notes has a link block, it will force SNC to create a version 2.0 format export file, since links were not supported in Stickies! before version 2.0.

Example:

```
:begin_note
(Do other, non-link things here )
Define a link that will run Notepad and load the file fred.txt
:begin_link
:program notepad.exe
:data_and_options fred.txt
:icon_source notepad.exe
:working_dir c:\mystuff\notes
:description Notes on Fred
:startup_option maximized
:icon_number 1
:run_on_alarm yep
:end_link
(Do other, non-link things here )
:end_note
```

See also :END_LINK

:BEGIN_NOTE

Type: Multi-line command
Data: None
Placement: Cannot appear in defaults or note block
Purpose: Marks the beginning of a note block

Notes: You must use a note block for each note you want to define in your export file. See the sample SML files for examples of how to define notes.

See also :END_NOTE

:BEGIN_TEXT

Type: Multi-line command
Data: None
Placement: Can appear only in note block
Purpose: Marks the beginning of the text block

Notes: All non-statement lines in a text block are used to fill the note's text area. If a text block contains no text, SNC will issue a warning and continue processing the file. Each line of note text is

added to the note's buffer with a hard return (actually, a CR/LF pair) at the end, unless that line ends with a '\'. Then, SNC will strip off the '\' and append the following line to the line that had the '\'. *Don't forget to add a blank before the '\' unless you want SNC to run the last word before the '\' and the beginning of the next line into one word!*

If you want to create a note with no text (for example, you want a note to have only links), then you can simply skip the text block for that note.

Example:

```
:begin_note
(Set other note options here)
:begin_text
This is some text that will appear in the note!
Let's add a blank line, just for kicks.

This line and the one after it \
will be in the same paragraph, \
thanks to the '\' characters at the \
end of these lines.
:end_text
:end_note
```

See also :END_TEXT

:BG_COLOR

Type: Single-line command
Data: Color name (see SML Data Formats section for details)
Placement: Defaults or note block
Default setting: dark_blue
Purpose: Sets the note's background color.

See also :FG_COLOR, :BEGIN_DEFAULTS (example)

:DATA_AND_OPTIONS

Type: Single-line command
Data: String
Placement: Link block
Default setting: Blank
Purpose: Provides the data and options line for a link

See also: BEGIN_LINK (example), :PROGRAM (notes)

:DESCRIPTION

Type: Single-line command
Data: String
Placement: Link block
Default setting: Blank
Purpose: Provides the description string for a link. This is the text that is displayed on a link button.

See also: BEGIN_LINK (example)

:DISABLE_WARNINGS

Type: Single-line directive
Data: None
Placement: Anywhere, but see notes below
Purpose: Tells SNC to disable all warning messages

Notes: This directive suppresses all warning messages (but not error message), beginning at the spot where the directive appears, and ending at the next :ENABLE_WARNINGS directive.

Example:

```
:begin_defaults
Normally SNC would issue a warning about the title command without a title,
but since warnings are disabled, it will quietly continue about its business.
:disable_warnings
:title
:enable_warnings
:end_defaults
```

See also :ENABLE_WARNINGS

:ENABLE_WARNINGS

Type: Single-line directive
Data: None
Placement: Anywhere, but see notes below
Purpose: Tells SNC to enable all warning messages

Notes: This directive enables all warning messages, beginning at the spot where the directive appears, and ending at the next :DISABLE_WARNINGS directive. Normally, SNC has warnings enabled.

See also :DISABLE_WARNINGS (example)

:END_DEFAULTS

Type: Multi-line command
Data: None
Placement: Cannot appear in note block

Purpose: Marks the end of a defaults block; required for all defaults blocks

See also :BEGIN_DEFAULTS

:END_LINK

Type: Multi-line command
Data: None
Placement: Cannot appear in defaults block
Purpose: Marks the end of a note block; required for all link blocks

See also :BEGIN_LINK

:END_NOTE

Type: Multi-line command
Data: None
Placement: Cannot appear in defaults block
Purpose: Marks the end of a note block; required for all note blocks

See also :BEGIN_NOTE

:END_TEXT

Type: Multi-line command
Data: None
Placement: Can appear only in note block
Purpose: Marks the end of a text block; required for all text blocks

See also :BEGIN_TEXT (example)

:EXPIRE

Type: Single-line command
Data: Boolean
Placement: Defaults or note block
Default setting: true/yes/yep/1/y
Purpose: Sets the note's expiration flag

Notes: If you set a note's expiration flag and don't provide an expiration date, that note will inherit the default expiration date, either the one you defined or the Stickies! default.

Example:

```
:begin_note
:expiration yep
:expire_date 6/12/1993
```

(Rest of note definition goes here)
:end_note

See also :EXPIRE_DATE

:EXPIRE_DATE

Type: Single-line command
Data: Date (no time component; see the SML Data Formats section for details)
Placement: Defaults or note block
Default setting: 12/31/2001
Purpose: Sets the note's expiration date.

Notes: Using this command does *not* force the note's expiration flag on. You still must use a separate :EXPIRE command.

See also :EXPIRE (example)

:FG_COLOR

Type: Single-line command
Data: Color name (see SML Data Formats section for details)
Placement: Defaults or note block
Default setting: white
Purpose: Sets the note's foreground, or text, color.

See also :BG_COLOR, :BEGIN_DEFAULTS (example)

:FORCE_VER20

Type: Single-line directive
Data: None
Placement: Anywhere
Purpose: Tells SNC to generate a Stickies! version 2.0 export file, even if no v2.0 features were used in the input file.

Notes: Normally, SNC will generate a v1.5 export file to provide maximum compatibility. But if you define any links on any of your notes, then SNC will produce a v2.0 file.

:GROUP_NAME

Type: Single-line command
Data: String
Placement: Defaults or note block
Default setting: "???"
Purpose: Sets the note's work group name field.

Example:

```
:begin_note
:group_name Department 99
(Rest of note definition goes here)
:end_note
```

:H

Type: Single-line command
Data: Integer
Placement: Defaults or note block
Default setting: 200
Purpose: Sets the height of a note in pixels.

Notes: The value you provide will be "clipped" into the range of acceptable note heights. Currently, this is 70 to 450 pixels, but it could change in a future release of Stickies!. If you specify a value that would make a note too "short" to display properly with its links, then Stickies! will change the note's height setting as needed.

See also :X (example), :Y, :W

:HIDDEN

Type: Single-line command
Data: Boolean
Placement: Defaults or note block
Default setting: false/no/nope/0/n
Purpose: Sets the note's hidden attribute

Notes: If a note has this flag set, it will not appear on the user's desktop when imported into the Stickies! active set, and it will not even appear in the Windows Task Manager list of active windows. A note imported directly into a user's file cabinet and then activated will have the hidden attribute turned off.

:ICON_NUMBER

Type: Single-line command
Data: Positive integer
Placement: Link block
Default setting: 0
Purpose: Specifies which icon in the icon source file Stickies! should display on a link button

See also: BEGIN_LINK (example), :ICON_SOURCE

:ICON_SOURCE

Type: Single-line command
Data: file specification
Placement: Link block
Default setting: NA
Purpose: Provides the icon source file name for a link; required for all link blocks

Notes: If a link block does not contain an :ICON_SOURCE command, SNC will report a missing or illegal :ICON_SOURCE command on the line that contains the :END_LINK command.

See also: BEGIN_LINK (example), :ICON_NUMBER

:MINIMIZED

Type: Single-line command
Data: Boolean
Placement: Defaults or note block
Default setting: false/no/nope/0/n
Purpose: Sets the note's hidden minimized attribute

Notes: If a note has this attribute, it will appear on the user's desktop as an icon, not a normal, "open" window.

:PROGRAM

Type: Single-line command
Data: String
Placement: Link block
Default setting: NA
Purpose: Provides the program line for a link; required in all links.

Notes: Unlike the :ICON_SOURCE and :WORKING_DIR commands, the :PROGRAM command takes a simple string argument, which means that the entire line following the command is treated as data. This is to allow you the maximum flexibility in specifying command line options; you can divide such text between the :PROGRAM and :DATA_AND_OPTIONS settings, since Stickies! concatenates these fields (with an intervening blank) just before it runs the link's program. You can set :PROGRAM to "c:\myprogs\zipcalc.exe /flerp" and :DATA_AND_OPTIONS to "c:\myfiles\newsales.zc" and Stickies! will use them to construct the command line: "c:\myprogs\zipcalc.exe /flerp c:\myfiles\newsales.zc".

If a link block does not contain a :PROGRAM command, SNC will report a missing or illegal :PROGRAM command on the line that contains the :END_LINK command.

See also: BEGIN_LINK (example)

:RUN_ON_ALARM

Type: Single-line command
Data: Boolean
Placement: Link block
Default setting: false/no/nope/0/n
Purpose: Specifies whether Stickies! should run a link's program when the alarm sounds for the note that contains the link

See also: BEGIN_LINK (example)

:SCROLL_BAR

Type: Single-line command
Data: Boolean
Placement: Defaults or note block
Default setting: false/no/nope/0/n
Purpose: Turns a scroll bar on or off in a note's text area.

Example:

```
:begin_note
:scroll_bar yes
(Rest of note definition goes here)
:end_note
```

:STARTUP_OPTION

Type: Single-line command
Data: minimized, normal, or maximized
Placement: Link block
Default setting: normal
Purpose: Provides the startup option for a link's program. This setting tells Stickies! whether to try to start the program minimized, normal, or maximized. Note that in Windows this setting is effectively just a suggestion. While most programs honor their startup option, some programs will only run a certain way. Stickies!, for example, will only run minimized (i.e. the main program is always an icon and never a normal window).

See also: BEGIN_LINK (example)

:TITLE

Type: Single-line command
Data: String
Placement: Defaults or note block
Default setting: "Another Sticky!"
Purpose: Sets the note title.

Example:

```
:begin_note
:title Hello, World!!!
(Rest of note definition goes here)
:end_note
```

:USER_NAME

Type: Single-line command
Data: String
Placement: Defaults or note block
Default setting: "???"
Purpose: Sets the note's user name field.

Example:

```
:begin_note
:user_name Fred Smith
(Rest of note definition goes here)
:end_note
```

:W

Type: Single-line command
Data: Integer
Placement: Defaults or note block
Default setting: 250
Purpose: Sets the width of a note in pixels.

Notes: The value you provide will be "clipped" into the range of acceptable note widths. Currently, this is 100 to 600 pixels, but it could change in a future release of Stickies!.

See also :X (example), :Y, :H

:WORKING_DIR

Type: Single-line command
Data: Directory
Placement: Link block
Default setting: Blank
Purpose: Provides the working directory for a link. This is the directory that Stickies! switches to just before it runs a link's program.

See also: BEGIN_LINK (example), :PROGRAM (notes)

:X

Type: Single-line command
Data: Integer
Placement: Defaults or note block
Default setting: 10
Purpose: Sets the x-coordinate for the note's upper-left corner.

Notes: Windows screen coordinate have 0,0 in the upper-left corner of the screen, so increasing x values move to the right, and increasing y moves down.

Example:

```
:begin_note
Let's place our note on the screen and size it...
:x 200
:y 20
:w 300
:h 250
:end_note
```

See also :Y, :W, :H

:Y

Type: Single-line command
Data: Integer
Placement: Defaults or note block
Default setting: 10
Purpose: Sets the y-coordinate for the note's upper-left corner.

Notes: Windows screen coordinate have 0,0 in the upper-left corner of the screen, so increasing x values move to the right, and increasing y moves down.

See also :X (example), :W, :H

SML Data Formats

Boolean

This is just a familiar true or false value, used for flags. Valid representations of each possible value are:

TRUE	FALSE
true	false
1	0
yes	no
yep	nope
y	n
on	off

Date

Dates are always in MDY format. The month and day only need as many digits as required for the number (i.e. you don't have to pad them with leading 0's). If the year is between 90 and 99, then 1900 will be added to it. If the year is between 0 and 50, then 2000 is added. Years in the ranges 1990..1999 and 2000..2050 are taken as presented, and all other values for a year are rejected as an error. Dates that specify more days than a month contains (taking into account leap years) or contain otherwise illegal or non-numeric month or day values will be rejected.

There can be no blanks anywhere in a date.

Legal dates:

1/1/94	
1/1/1994	
06/01/2001	
12/31/0	(Interpreted as December 31st, 2000)

Date and Time

Alarms require both a date and a time value. The date always comes first, and is in the format described above. The time value is in 24-hour format (e.g. 3PM is 15:00), and a seconds field is not accepted. As with dates, no blanks are allowed in the middle of a time, and you don't have to pad out parts of the time with leading zeroes.

Legal times:

12:30	(12:30PM)
1:1	(1:01AM)
23:59	(11:59PM)

Directory

When a command requires a directory, the text following the command, up to the next blank, is used, and stripped of leading and trailing blanks. SNC will read the line:

```
:working_dir      d:\mystuff\data              Point to my secret files
```

and use the string "d:\mystuff\data" (without the quotes) as the working directory for the link that is being defined and ignore the rest of the line.

Note that SNC does not check that the directory exists, or even that the string can be interpreted as a directory.

File Specification

When a command requires a file specification, the text following the command, up to the next blank, is used, and stripped of leading and trailing blanks. SNC will read the line:

```
:icon_source     c:\windows\notepad.exe      Let's use NOTEPAD this time.
```

and use the string "c:\windows\notepad.exe" (without the quotes) as the icon source file and ignore the rest of the line.

Note that SNC does not check that the file exists, or even that the string can be interpreted as a file specification.

Integer and Positive Integer

Integers are limited to the range -32768 to 32767, and positive (technically, non-negative integers) are limited to the range 0 to 32767. Some SML commands further restrict the range of the integers they will accept; the alarm repeat time cannot be greater than 99 minutes, for example, and any value greater than 99 will cause SNC to report an error.

String

Unlike other languages, SML does not recognize string delimiters such as single or double quotes. Any SML command that requires a string argument will assume that its string value begins with the first non-blank after the command, and ends at the end of the line. Trailing blanks are stripped from the string. For example, in the following line, using an underscore to represent a blank (something you would never do in a real SML file, of course):

```
:title_____Hello, World!!!________
```

The note's title would be set to the string "Hello, World!!!", without the quotes.

Default Settings

Notes

Alarm flag	false
Alarm date and time	12/31/2001 9:00AM
Alarm prefix	15 minutes
Alarm sound	true
Alarm repeat	0 minutes/off (Stickies! 2.0 feature)
Alarm daily flag	false (Stickies! 2.0 feature)
Background color	dark_blue
Expiration flag	false
Expiration date	12/31/2001
Group name	Blank
Height	200
Scroll bar	false
Text/foreground color	white
Title	Blank
User name	Blank
Width	250
X-coordinate, upper-left corner	10
Y-coordinate, upper-left corner	10

Links

Data and options line	Blank
Description	Blank
Icon number	0
Icon source file	NA (required field)
Program	NA (required field)
Run on alarm	false
Startup option	normal
Working directory	Blank

Technical Support

Even though this release of SNC is freeware, we urge users to report any problems they have with the program, or send us comments and suggestions for future releases. We can be reached at:

Looking Glass Technologies
P.O. Box 8636
Endwell, NY 13762-8636

CompuServe: 71055,1240

GEnie: L.G.TECH