# Supercite User's Manual

# Supercite Version 3.1

Manual Revision: 3.47
August 1993

Barry A. Warsaw
bwarsaw@cen.com
...!uunet!cen.com!bwarsaw

# 1 Introduction

Supercite version 3.1 is a GNU Emacs package written entirely in Emacs Lisp. It interfaces to most of the commonly used Emacs mail user agents (*MUAs*) and news user agents (*NUAs*), and provides sophisticated facilities for the citing and attributing of message replies. Supercite has a very specific and limited role in the process of composing replies to both USENET network news and electronic mail.

The preferred way to spell Supercite is with a capital '`S`', lowercase '`upercite`'. There are a few alternate spellings out there and I won't be terribly offended if you use them. People often ask though...

Supercite is only useful in conjunction with MUAs and NUAs such as VM, GNUS, RMAIL, etc. (hereafter referred to collectively as MUAs). Supercite is typically called by the MUA after a reply buffer has been setup. Thereafter, Supercite's many commands and formatting styles are available in that reply buffer until the reply is sent. Supercite is re-initialized in each new reply buffer.

Supercite is currently at major revision 3.1, and is known to work in the following environments:

Emacs versions:
: GNU Emacs 18.57 through 18.59, all Emacs 19, all current Lucid Emacs, and Epoch 4.

MUAs:
: VM 4.37 and beyond (including VM version 5), RMAIL, MH-E 3.7 and beyond, PCMAIL.

NUAs:
: RNEWS, GNUS 3.12 and beyond, GNEWS.

For systems with version numbers, all known subsequent versions also work with Supercite. For those systems without version numbers, Supercite probably works with any recently released version. Note that only some of these systems will work with Supercite "out of the box." All others must overload interfacing routines to supply the necessary glue. See Chapter 5 [Getting Connected], page 11, for more details.

## 1.1 Usage Overview

Typical usage is as follows. You want to reply or followup to a message in your MUA. You will probably hit *`r`* (i.e., "reply") or *`f`* (i.e., "forward") to begin composing the reply. In response, the MUA will create a reply buffer and initialize the outgoing mail headers appropriately. The body of the reply will usually be empty at this point. You now decide that you would like to include part of the original message in your reply. To do this, you *yank* the original message into the reply buffer, typically with a key stroke such as *`C-c C-y`*. This sequence will invoke an MUA-specific function which fills the body of the reply with the original message and then *attributes* this text to its author. This is called *citing* and its effect is to prefix every line from the original message with a special text tag. Most MUAs provide some default style of citing; by using Supercite you gain a wider flexibility in the look and style of citations. Supercite's only job is to cite the original message.

## 1.2 What Supercite Doesn't Do

Because of this clear division of labor, there are useful features which are the sole responsibility of the MUA, even though it might seem that Supercite should provide them. For example, many people would like to be able to yank (and cite) only a portion of the original message. Since Supercite only modifies the text it finds in the reply buffer as set up by the MUA, it is the MUA's responsibility to do partial yanking. See Section 6.1 [Reply Buffer Initialization], page 15.

Another potentially useful thing would be for Supercite to set up the outgoing mail headers with information it gleans from the reply buffer. But by previously agreed upon convention, any text above the `mail-header-separator` which separates mail headers from message bodies cannot be modified by Supercite. Supercite, in fact, doesn't know anything about the meaning of these headers, and never ventures outside the designated region. See Chapter 10 [Hints to MUA Authors], page 29, for more details.

## 1.3 What Supercite Does

Supercite is invoked for the first time on a reply buffer via your MUA's reply or forward command. This command will actually perform citations by calling a hook variable to which Supercite's top-level function `sc-cite-original` has been added. When `sc-cite-original` is executed, the original message must be set up in a very specific way, but this is handled automatically by the MUA. See Chapter 10 [Hints to MUA Authors], page 29.

The first thing Supercite does, via `sc-cite-original`, is to parse through the original message's mail headers. It saves this data in an *information association list*, or *info alist*. The information in this list is used in a number of places throughout Supercite. See Chapter 3 [Information Keys and the Info Alist], page 6.

After the mail header info is extracted, the headers are optionally removed (*nuked*) from the reply. Supercite then writes a *reference header* into the buffer. This reference header is a string carrying details about the citation it is about to perform.

Next, Supercite visits each line in the reply, transforming the line according to a customizable "script". Lines which were not previously cited in the original message are given a citation, while already cited lines remain untouched, or are coerced to your preferred style. Finally, Supercite installs a keymap into the reply buffer so that you have access to Supercite's post-yank formatting and reciting commands as you subsequently edit your reply. You can tell that Supercite has been installed into the reply buffer because that buffer's modeline will display the minor mode string '`SC`'.

When the original message is cited by `sc-cite-original`, it will (optionally) be filled by Supercite. However, if you manually edit the cited text and want to re-fill it, you must use an add-on package such as *filladapt* or *gin-mode*. These packages can recognize Supercited text and will fill them appropriately. Emacs' built-in filling routines, e.g. `fill-paragraph`, do not recognize cited text and will not re-fill them properly because it cannot guess the `fill-prefix` being used. See Chapter 9 [Post-yank Formatting Commands], page 25, for details.

As mentioned above, Supercite provides commands to recite or uncite regions of text in the reply buffer, and commands to perform other beautifications on the cited original text, maintaining consistent and informative citations throughout. Supercite tries to be as configurable as possible to allow for a wide range of personalized citation styles, but it is also immediately useful with the default configuration, once it has been properly connected to your MUA. See Chapter 5 [Getting Connected], page 11, for more details.

# 2 Citations

A *citation* is the acknowledgement of the original author of a mail message in the body of the reply. There are two basic citation styles which Supercite supports. The first, called *nested citations* is an anonymous form of citation; in other words, an indication is made that the cited line was written by someone *other* that the current message author (i.e., other than you, the person composing the reply), but no reference is made as to the identity of the original author. This style should look familiar since its use on the net is widespread. Here's an example of what a message buffer would look like using nested citations after multiple replies:

```
>> John originally wrote this
>> and this as well
> Jane said that John didn't know
> what he was talking about
And that's what I think too.
```

Note that multiple inclusions of the original messages result in a nesting of the '`>`' characters. This can sometimes be quite confusing when many levels of citations are included since it may be difficult or impossible to figure out who actually participated in the thread, and multiple nesting of '`>`' characters can sometimes make the message very difficult for the eye to scan.

In *non-nested citations*, each cited line begins with an informative string attributing that line to the original author. Only the first level of attribution will be shown; subsequent citations don't nest the citation strings. The above dialog might look like this when non-nested citations are used:

```
John> John originally wrote this
John> and this as well
Jane> Jane said that John didn't know
Jane> what he was talking about
And that's what I think too.
```

Notice here that my inclusion of Jane's inclusion of John's original message did not result in a line cited with '`Jane>John>`'.

Supercite supports both styles of citation, and the variable `sc-nested-citation-p` controls which style it will use when citing previously uncited text. When this variable is `nil` (the default), non-nested citations are used. When non-`nil`, nested citations are used.

## 2.1 Citation Elements

*Citation strings* are composed of one or more elements. Non-nested citations are composed of four elements, three of which are directly user definable. The elements are concatenated together, in this order:

1. The *citation leader*. The citation leader is contained in the variable `sc-citation-leader`, and has the default value of a string containing four spaces.
2. The *attribution string*. This element is supplied automatically by Supercite, based on your preferences and the original message's mail headers, though you may be asked to confirm Supercite's choice. See Chapter 7 [Selecting an Attribution], page 18, for more details.

3. The *citation delimiter*. This string, contained in the variable `sc-citation-delimiter` visually separates the citation from the text of the line. This variable has a default value of `">"` and for best results, the string should consist of only a single character.
4. The *citation separator*. The citation separator is contained in the variable `sc-citation-separator`, and has the default value of a string containing a single space.

For example, suppose you were using the default values for the above variables, and Supercite provided the attribution string '`Jane`'. In this case, the composed, non-nested citation string used might be something like `" Jane> "`. This citation string will be inserted in front of every line in the original message that is not already cited.

Nested citations, being simpler than non-nested citations, are composed of the same elements, sans the attribution string. Supercite is smart enough to not put additional spaces between citation delimiters for multi-level nested citations.

## 2.2 Recognizing Citations

Supercite also recognizes citations in the original article, and can transform these already cited lines in a number of ways. This is how Supercite suppresses the multiple citing of non-nested citations. Recognition of cited lines is controlled by variables analogous to those that make up the citation string as mentioned previously.

The variable `sc-citation-leader-regexp` describes how citation leaders can look, by default it matches any number of spaces or tabs. Note that since the lisp function `looking-at` is used to do the matching, if you change this variable it need not start with a leading `"^"`.

Similarly, the variables `sc-citation-delimiter-regexp` and `sc-citation-separator-regexp` respectively describe how citation delimiters and separators can look. They follow the same rule as `sc-citation-leader-regexp` above.

When Supercite composes a citation string, it provides the attribution automatically. The analogous variable which handles recognition of the attribution part of citation strings is `sc-citation-root-regexp`. This variable describes the attribution root for both nested and non-nested citations. By default it can match zero-to-many alphanumeric characters (also ".", "-", and "_"). But in some situations, Supercite has to determine whether it is looking at a nested or non-nested citation. Thus the variable `sc-citation-nonnested-root-regexp` is used to describe only non-nested citation roots. It is important to remember that if you change `sc-citation-root-regexp` you should always also change `sc-citation-nonnested-root-regexp`.

Nemacs users: For best results, try setting `sc-citation-root-regexp` to:

```
"\\([-._a-zA-Z0-9]\\|\\cc\\|\\cC\\|\\ch\\|\\cH\\|\\ck\\|\\cK\\|\\ca\\|\\cg\\|\\cr\\|\\cu\\)*"
```

Mule users: For best results, try setting `sc-citation-root-regexp` to:

```
"\\([-._a-zA-Z0-9]\\|\\cj\\)*"
```

# 3 Information Keys and the Info Alist

*Mail header information keys* are nuggets of information that Supercite extracts from the various mail headers of the original message, placed in the reply buffer by the MUA. Information is kept in the *Info Alist* as key-value pairs, and can be retrieved for use in various places within Supercite, such as in header rewrite functions and attribution selection. Other bits of data, composed and created by Supercite, are also kept as key-value pairs in this alist. In the case of mail fields, the key is the name of the field, omitting the trailing colon. Info keys are always case insensitive (as are mail headers), and the value for a corresponding key can be retrieved from the alist with the `sc-mail-field` function. Thus, if the following fields were present in the original article:

```
Date: 08 April 1991, 17:32:09 EST
Subject: Better get out your asbestos suit
```

then, the following lisp constructs return:

```
(sc-mail-field "date")
==> "08 April 1991, 17:32:09 EST"

(sc-mail-field "subject")
==> "Better get out your asbestos suit"
```

Since the argument to `sc-mail-field` can be any string, it is possible that the mail field will not be present on the info alist (possibly because the mail header was not present in the original message). In this case, `sc-mail-field` will return the value of the variable `sc-mumble`.

Supercite always places all mail fields found in the yanked original article into the info alist. If possible, Supercite will also places the following keys into the info alist:

`"sc-attribution"`
: the selected attribution string.

`"sc-citation"`
: the non-nested citation string.

`"sc-from-address"`
: email address extracted from the '`From:`' field.

`"sc-reply-address"`
: email address extracted from the '`Reply-To:`' field.

`"sc-sender-address"`
: email address extracted from the '`Sender:`' field.

`"sc-emailname"`
: email terminus extracted from the '`From:`' field.

`"sc-initials"`
: the author's initials.

`"sc-author"`
: the author's full name.

"sc-firstname"
: the author's first name.

"sc-lastname"
: the author's last name.

"sc-middlename-1"
: the author's first middle name.

If the author's name has more than one middle name, they will appear as info keys with the appropriate index (e.g., "sc-middlename-2", ...). See Chapter 7 [Selecting an Attribution], page 18.

# 4 Reference Headers

Supercite will insert an informative *reference header* at the beginning of the cited body of text, which display more detail about the original article and provides the mapping between the attribution and the original author in non-nested citations. Whereas the citation string usually only contains a portion of the original author's name, the reference header can contain such information as the author's full name, email address, the original article's subject, etc. In fact any information contained in the info alist can be inserted into a reference header.

There are a number of built-in *header rewrite functions* supplied by Supercite, but you can write your own custom header rewrite functions (perhaps using the built-in ones as examples). The variable `sc-rewrite-header-list` contains the list of such header rewrite functions. This list is consulted both when inserting the initial reference header, and when displaying *electric references*. See Section 4.2 [Electric References], page 9.

When Supercite is initially run on a reply buffer (via `sc-cite-original`), it will automatically call one of these functions. The one it uses is defined in the variable `sc-preferred-header-style`. The value of this variable is an integer which is an index into the `sc-rewrite-header-list`, beginning at zero.

## 4.1 The Built-in Header Rewrite Functions

Below are examples of the various built-in header rewrite functions. Please note the following: first, the text which appears in the examples below as *infokey* indicates that the corresponding value of the info key from the info alist will be inserted there. (see Chapter 3 [Information Keys and the Info Alist], page 6). For example, in `sc-header-on-said` below, *date* and *from* correspond to the values of the '`Date:`' and '`From:`' mail headers respectively.

Also, the string `">>>>>"` below is really the value of the variable `sc-reference-tag-string`. This variable is used in all built-in header rewrite functions, and you can customize its value to change the tag string globally.

Finally, the references headers actually written may omit certain parts of the header if the info key associated with *infokey* is not present in the info alist. In fact, for all built-in headers, if the '`From:`' field is not present in the mail headers, the entire reference header will be omitted (but this usually signals a serious problem either in your MUA or in Supercite's installation).

`sc-no-header`
: This function produces no header. It should be used instead of `nil` to produce a blank header. This header can possibly contain a blank line after the `mail-header-separator` line.

`sc-no-blank-line-or-header`
: This function is similar to `sc-no-header` except that any blank line after the `mail-header-separator` line will be removed.

`sc-header-on-said`
: `>>>>> On` *date*, *from* `said:`

`sc-header-inarticle-writes`
    `>>>>> In article` *message-id*`,` *from* `writes:`

`sc-header-regarding-adds`
    `>>>>> Regarding` *subject*`;` *from* `adds:`

`sc-header-attributed-writes`
    `>>>>> "`*sc-attribution*`" ==` *sc-author* `<`*sc-reply-address*`> writes:`

`sc-header-author-writes`
    `>>>>>` *sc-author* `writes:`

`sc-header-verbose`
    `>>>>> On` *date*`,`
    `>>>>>` *sc-author*
    `>>>>> from the organization of` *organization*
    `>>>>> who can be reached at:` *sc-reply-address*
    `>>>>> (whose comments are cited below with: "`*sc-cite*`")`
    `>>>>> had this to say in article` *message-id*
    `>>>>> in newsgroups` *newsgroups*
    `>>>>> concerning the subject of` *subject*
    `>>>>> see` *references* `for more details`

## 4.2 Electric References

By default, when Supercite cites the original message for the first time, it just goes ahead and inserts the reference header indexed by `sc-preferred-header-style`. However, you may want to select different reference headers based on the type of reply or forwarding you are doing. You may also want to preview the reference header before deciding whether to insert it into the reply buffer or not. Supercite provides an optional *electric reference* mode which you can drop into to give you this functionality.

If the variable `sc-electric-references-p` is non-`nil`, Supercite will bring up an electric reference mode buffer and place you into a recursive edit. The electric reference buffer is read-only, so you cannot directly modify the reference text until you exit electric references and insert the text into the reply buffer. But you can cycle through all the reference header rewrite functions in your `sc-rewrite-header-list`.

You can also set a new preferred header style, jump to any header, or jump to the preferred header. The header will be shown in the electric reference buffer and the header index and function name will appear in the echo area.

The following commands are available while in electric reference mode (shown here with their default key bindings):

`sc-eref-next` (*n*)
    Displays the next reference header in the electric reference buffer. If the variable `sc-electric-circular-p` is non-`nil`, invoking `sc-eref-next` while viewing the last reference header in the list will wrap around to the first header.

`sc-eref-prev` (*p*)
: Displays the previous reference header in the electric reference buffer. If the variable `sc-electric-circular-p` is non-`nil`, invoking `sc-eref-prev` will wrap around to the last header.

`sc-eref-goto` (*g*)
: Goes to a specified reference header. The index (into the `sc-rewrite-header-list`) can be specified as a numeric argument to the command. Otherwise, Supercite will query you for the index in the minibuffer.

`sc-eref-jump` (*j*)
: Display the preferred reference header, i.e., the one indexed by the current value of `sc-preferred-header-style`.

`sc-eref-setn` (*s*)
: Set the preferred reference header (i.e., `sc-preferred-header-style`) to the currently displayed header.

`sc-eref-exit` (*C-j*, ⟨RET⟩, and ⟨ESC C-c⟩)
: Exit from electric reference mode and insert the current header into the reply buffer.

`sc-eref-abort` (*q*, *x*)
: Exit from electric reference mode without inserting the current header.

Supercite will execute the hook `sc-electric-mode-hook` before entering electric reference mode.

# 5 Getting Connected

Hitting *C-c C-y* in your MUA's reply buffer yanks and cites the original message into the reply buffer. In reality, the citation of the original message is performed via a call through a configurable hook variable. The name of this variable has been agreed to in advance as part of the *citation interface specification*. By default this hook variable has a `nil` value, which the MUA recognizes to mean, "use your default citation function". When you add Supercite's citation function to the hook, thereby giving the variable a non-`nil` value, it tells the MUA to run the hook via `run-hooks` instead of using the default citation.

Early in Supercite's development, the Supercite author, a few MUA authors, and some early Supercite users got together and agreed upon a standard interface between MUAs and citation packages (of which Supercite is currently the only known add-on `:-)`. With the recent release of the Free Software Foundation's GNU Emacs 19, the interface has undergone some modification and it is possible that not all MUAs support the new interface yet. Some support only the old interface and some do not support the interface at all. Still, it is possible for all known MUAs to use Supercite, and the following sections will outline the procedures you need to follow.

To learn exactly how to connect Supercite to the software systems you are using, read the appropriate following sections. For details on the interface specifications, or if you are writing or maintaining an MUA, see Chapter 10 [Hints to MUA Authors], page 29.

The first thing that everyone should do, regardless of the MUA you are using is to set up Emacs so it will load Supercite at the appropriate time. You can either dump Supercite into your Emacs binary (ask your local Emacs guru how to do this if you don't know), or you can set up an *autoload* for Supercite. To do the latter, put the following in your '`.emacs`' file:

```
(autoload 'sc-cite-original     "supercite" "Supercite 3.1" t)
(autoload 'sc-submit-bug-report "supercite" "Supercite 3.1" t)
```

The function `sc-cite-original` is the top-level Supercite function designed to be run from the citation hook. It expects '`point`' and '`mark`' to be set around the region to cite, and it expects the original article's mail headers to be present within this region. Note that Supercite *never* touches any text outside this region. Note further that for Emacs 19, the region need not be active for `sc-cite-original` to do its job. See Chapter 10 [Hints to MUA Authors], page 29.

The other step in the getting connected process is to make sure your MUA calls `sc-cite-original` at the right time. As mentioned above, some MUAs handle this differently. Read the sections that follow pertaining to the MUAs you are using.

One final note. After Supercite is loaded into your Emacs session, it runs the hook `sc-load-hook`. You can put any customizations into this hook since it is only run once. This will not work, however, if your Emacs maintainer has put Supercite into your dumped Emacs' image. In that case, you can use the `sc-pre-hook` variable, but this will get executed every time `sc-cite-original` is called. See Section 6.1 [Reply Buffer Initialization], page 15.

## 5.1 GNUS, RMAIL, or RNEWS with any Emacs 19

These MUAs, distributed with Emacs and with Lucid Emacs, use Emacs's built-in yanking facility, which provides the citing hook variable `mail-citation-hook`. By default, this hook's value is `nil`, but by adding the following to your '`.emacs`' file, you can tell these MUAs to use Supercite to perform the citing of the original message:

```
(add-hook 'mail-citation-hook 'sc-cite-original)
```

GNUS users may also want to add the following bit of lisp as well. This prevents GNUS from inserting its default attribution header. Otherwise, both GNUS and Supercite will insert an attribution header:

```
(setq news-reply-header-hook nil)
```

## 5.2 GNUS, RMAIL, PCMAIL, RNEWS with Emacs 18 or Epoch 4

These MUAs use Emacs' built-in yanking and citing routines, contained in the '`sendmail.el`' file. '`sendmail.el`' for Emacs 18, and its derivative Epoch 4, do not know anything about the citation interface required by Supercite. To connect Supercite to any of these MUAs under Emacs 18 or Epoch 4, you should first see Section 5.6 [Overloading for Non-conforming MUAs], page 13. Then follow the directions for using these MUAs under Emacs 19. See Section 5.1 [Emacs 19 MUAs], page 12.

Note that those instructions will tell you to use the function `add-hook`. This function is new with Emacs 19 and you will not have it by default if you are running Emacs 18 or Epoch 4. You can either substitute the appropriate call to `setq`, or you can use the `add-hook` function that is provided in the '`sc-unsupp.el`' file of unsupported Supercite hacks and ideas. Or you can upgrade to some Emacs 19 variant! `:-)`

To use `setq` instead of `add-hook`, you would, for example, change this:

```
(add-hook 'mail-citation-hook 'sc-cite-original)
```

to:

```
(setq mail-citation-hook 'sc-cite-original)
```

Note the lack of of a single quote on the first argument to `setq`.

## 5.3 MH-E with any Emacsen

MH-E 4.x conforms to the `mail-citation-hook` interface supported by other MUAs. At the time of this writing, MH-E 4.0 has not been released, but if you have it, put this in your '`.emacs`' file to connect Supercite and MH-E 4.x:

```
(add-hook 'mail-citation-hook 'sc-cite-original)
```

Note that if you are using Emacs 18 or Epoch 4, you will not have the `add-hook` function. See Section 5.2 [Emacs 18 MUAs], page 12, for details on how to proceed without `add-hook`.

MH-E version 3.x uses a slightly different interface than other MUAs. MH-E provides a hook variable `mh-yank-hooks`, but it doesn't act like a hook, and doing an `add-hook` will not work.

To connect Supercite to MH-E 3.x, you should instead add the following to your `.emacs` file:

```
(add-hook 'mh-yank-hooks 'sc-cite-original)
```

You also need to make sure that MH-E includes all the original mail headers in the yanked message. The variable that controls this is `mh-yank-from-start-of-msg`. By default, this variable has the value `t`, which tells MH-E to include all the mail headers when yanking the original message. Before you switched to using Supercite, you may have set this variable to other values so as not to include the mail headers in the yanked message. Since Supercite requires these headers (and cleans them out for you), you need to make sure the value is `t`. This lisp, in your '`.emacs`' file will do the trick:

```
(setq mh-yank-from-start-of-msg t)
```

Note that versions of MH-E before 3.7 did not provide the `mh-yank-hooks` variable. Your only option is to upgrade to MH-E version 3.7 or later.

## 5.4 VM with any Emacsen

Since release 4.40, VM has supported the citation interface required by Supercite. But since the interface has changed recently the details of getting connected differ with the version of VM you are using.

If you are running any release of VM after 4.40, you can add the following to your '`.emacs`' to connect Supercite with VM:

```
(add-hook 'mail-yank-hooks 'sc-cite-original)
```

Note that if you are using Emacs 18 or Epoch 4, you will not have the `add-hook` function. See Section 5.2 [Emacs 18 MUAs], page 12, for details on how to proceed without `add-hook`.

Since version 5.34, VM has supported the newer `mail-citation-hook` interface, but `mail-yank-hooks` is still being supported for backward compatibility. If you are running a newer version of VM and you want to maintain consistency with other MUAs, use this bit of code instead:

```
(add-hook 'mail-citation-hook 'sc-cite-original)
```

## 5.5 GNEWS with any Emacsen

As far as I know, no version of GNEWS supports the citation interface required by Supercite. To connect Supercite with GNEWS, please first see Section 5.6 [Overloading for Non-conforming MUAs], page 13.

After you have followed the directions in that section. You should add the following lisp code to your '`.emacs`' file:

```
(add-hook 'mail-citation-hook 'sc-cite-original)
```

Note that if you are using Emacs 18 or Epoch 4, you will not have the `add-hook` function. See Section 5.2 [Emacs 18 MUAs], page 12, for details on how to proceed without `add-hook`.

## 5.6 Overloading for Non-conforming MUAs

As mentioned elsewhere, some MUAs do not provide the necessary hooks to connect with Supercite. Supercite version 3.1 provides an unsupported mechanism, called *overloading* which redefines certain key functions in the MUA, so that it will call the `mail-citation-hook` variable instead of the MUA's default hard-coded citing routines. Since most newer versions of the known MUAs support the `mail-citation-hook` variable, it is recommended that you upgrade if at all possible. But if you can't upgrade, at least you're not out of luck! Once you set up overloading properly, you should follow the directions for connecting Supercite to the Emacs 19 MUAs. See Section 5.1 [Emacs 19 MUAs], page 12.

Users of Bob Weiner's Hyperbole package take note. Hyperbole provides the necessary overloads (and a whole lot more!) and you can potentially clobber it if you were to load Supercite's overloading after Hyperbole's. For this reason, Supercite will *not* perform any overloading if it finds the variable `hyperb:version` is `boundp` (i.e. it exists because Hyperbole has been loaded into your Emacs session). If this is the case, Supercite will display a warning message in the minibuffer. You should consult the Hyperbole manual for further details.

Overloading involves the re-definition of the citing function with the new, `mail-citation-hook` savvy version. The function in '`sc-oloads.el`' that does this is `sc-perform-overloads`. This function is smart enough to only overload the MUA functions when it is absolutely necessary, based on the version numbers it can figure out. Also, `sc-perform-overloads` will only install the new functions once. It is also smart enough to do nothing if the MUA is not yet loaded.

The tricky part is finding the right time and place to perform the overloading. It must be done after the MUA has been loaded into your Emacs session, but before the first time you try to yank in a message. Fortunately, this has been figured out for you.

If you must overload, you should put the following lisp code in your '`.emacs`' file, to make sure the '`sc-oloads.el`' file gets loaded at the right time:

```
(autoload 'sc-perform-overloads "sc-oloads" "Supercite 3.1" t)
```

Then you must make sure that the function `sc-perform-overloads` gets run at the right time. For GNUS, put this in your '`.emacs`' file:

```
(setq news-reply-mode-hook 'sc-perform-overloads)
(setq mail-setup-hook 'sc-perform-overloads)
```

If you are using RNEWS, put this in your '`.emacs`' file:

```
(setq news-reply-mode-hook 'sc-perform-overloads)
```

If you are using RMAIL or PCMAIL, put this in your '`.emacs`' file:

```
(setq mail-setup-hook 'sc-perform-overloads)
```

If you are using GNEWS, put this in your '`.emacs`' file:

```
(setq news-reply-mode-hook 'sc-perform-overloads)
(setq gnews-ready-hook 'sc-perform-overloads)
```

Now go back and follow the directions for getting the Emacs 19 MUAs connected to Supercite. Be sure to see Section 5.2 [Emacs 18 MUAs], page 12 on substitutes for Emacs 19's `add-hook` function.

# 6 Replying and Yanking

## 6.1 Reply Buffer Initialization

Executing `sc-cite-original` performs the following steps as it initializes the reply buffer:

1. *Runs* `sc-pre-hook`. This hook variable is run before `sc-cite-original` does any other work. You could conceivably use this hook to set certain Supercite variables based on the reply buffer's mode or name (i.e., to do something different based on whether you are replying or following up to an article).
2. *Inserts Supercite's keymap.* Supercite provides a number of commands for performing post-yank modifications to the reply buffer. These commands are installed on Supercite's top-level keymap. Since Supercite has to interface with a wide variety of MUAs, it does not install all of its commands directly into the reply buffer's keymap. Instead, it puts its commands on a keymap prefix, then installs this prefix onto the buffer's keymap. What this means is that you typically have to type more characters to invoke a Supercite command, but Supercite's keybindings can be made much more consistent across MUAs.

   You can control what key Supercite uses as its keymap prefix by changing the variable `sc-mode-map-prefix`. By default, this variable is set to `C-c C-p`; a finger twister perhaps, but unfortunately the best default due to the scarcity of available keybindings in many MUAs.
3. *Turns on Supercite minor mode.* The modeline of the reply buffer should indicate that Supercite is active in that buffer by displaying the string '`SC`'.
4. *Sets the "Undo Boundary".* Supercite sets an undo boundary before it begins to modify the original yanked text. This allows you to easily undo Supercite's changes to affect alternative citing styles.
5. *Processes the mail headers.* All previously retrieved info key-value pairs are deleted from the info alist, then the mail headers in the body of the yanked message are scanned. Info key-value pairs are created for each header found. Also, such useful information as the author's name and email address are extracted. If the variable `sc-mail-warn-if-non-rfc822-p` is non-`nil`, then Supercite will warn you if it finds a mail header that does not conform to RFC822. This is rare and indicates a problem either with your MUA or the original author's MUA, or some MTA (mail transport agent) along the way.

   Once the info keys have been extracted from the mail headers, the headers are nuked from the reply buffer. You can control exactly which headers are removed or kept, but by default, all headers are removed.

   There are two variables which control mail header nuking. The variable `sc-nuke-mail-headers` controls the overall behavior of the header nuking routines. By setting this variable to `'all`, you automatically nuke all mail headers. Likewise, setting this variable to `'none` inhibits nuking of any mail headers. In between these extremes, you can tell Supercite to nuke only a specified list of mail headers by setting this variable to `'specified`, or to keep only a specified list of headers by setting it to `'keep`.

If `sc-nuke-mail-headers` is set to `'specified` or `'keep`, then the variable `sc-nuke-mail-header-list` is consulted for the list of headers to nuke or keep. This variable contains a list of regular expressions. If the mail header line matches a regular expression in this list, the header will be nuked or kept. The line is matched against the regexp using `looking-at` rooted at the beginning of the line.

If the variable `sc-blank-lines-after-headers` is non-`nil`, it contains the number of blank lines remaining in the buffer after mail headers are nuked. By default, only one blank line is left in the buffer.

6. *Selects the attribution and citation strings.* Once the mail headers have been processed, Supercite selects a attribution string and a citation string which it will use to cite the original message. See Chapter 7 [Selecting an Attribution], page 18, for details.

7. *Cites the message body.* After the selection of the attribution and citation strings, Supercite cites the original message by inserting the citation string prefix in front of every uncited line. You may not want Supercite to automatically cite very long messages however. For example, some email could contain a smaller header section followed by a huge uuencoded message. It wouldn't make sense to cite the uuencoded message part when responding to the original author's short preface. For this reason, Supercite provides a variable which limits the automatic citation of long messages to a certain maximum number of lines. The variable is called `sc-cite-region-limit`. If this variable contains an integer, messages with more lines that this will not be cited at all, and a warning message will be displayed. Supercite has performed everything necessary, though, for you to manually cite only the small portion of the original message that you want to use.

   If `sc-cite-region-limit` contains a non-`nil` value, the original message will always be cited, regardless of its size. If the variable contains the value `nil`, the region will never be cited automatically. Use this if you always want to be able to edit and cite the message manually.

   The variable `sc-cite-blank-lines-p` controls whether blank lines in the original message should be cited or not. If this variable is non-`nil`, blank lines will be cited just like non-blank lines. Otherwise, blank lines will be treated as paragraph separators.

   Citing of the original message is highly configurable. Supercite's default setup does a pretty good job of citing many common forms of previously cited messages. But there are as many citation styles out there as people on the net, or just about! It would be impossible for Supercite to anticipate every style in existence, and you probably wouldn't encounter them all anyway. But you can configure Supercite to recognize those styles you see often. See Chapter 8 [Configuring the Citation Engine], page 22, for details.

8. *Runs* `sc-post-hook`. This variable is very similar to `sc-pre-hook`, except that it runs after `sc-cite-original` is finished. This hook is provided mostly for completeness and backward compatibility. Perhaps it could be used to reset certain variables set in `sc-pre-hook`.

## 6.2 Filling Cited Text

Supercite will automatically fill newly cited text from the original message unless the variable `sc-auto-fill-region-p` has a `nil` value. Supercite will also re-fill paragraphs when you manually cite or re-cite text.

However, during normal editing, Supercite itself cannot be used to fill paragraphs. This is a change from version 2. There are other add-on lisp packages which do filling much better than Supercite ever did. The two best known are *filladapt* and *gin-mode*. Both work well with Supercite and both are available at the normal Emacs Lisp archive sites. *gin-mode* works pretty well out of the box, but if you use *filladapt*, you may want to run the function `sc-setup-filladapt` from your `sc-load-hook`. This simply makes *filladapt* a little more Supercite savvy than its default setup.

Also, Supercite will collapse leading whitespace between the citation string and the text on a line when the variable `sc-fixup-whitespace-p` is non-`nil`. The default value for this variable is `nil`.

Its important to understand that Supercite's automatic filling (during the initial citation of the reply) is very fragile. That is because figuring out the `fill-prefix` for a particular paragraph is a really hard thing to do automatically. This is especially the case when the original message contains code or some other text where leading whitespace is important to preserve. For this reason, many Supercite users typically run with `sc-auto-fill-region-p` (and possibly also `sc-fixup-whitespace-p`) set to `nil`. They then manually fill each cited paragraph in the reply buffer.

I usually run with both these variables containing their default values. When Supercite's automatic filling breaks on a particular message, I will use Emacs' undo feature to undo back before the citation was applied to the original message. Then I'll toggle the variables and manually cite those paragraphs that I don't want to fill or collapse whitespace on. See Section 9.3 [Variable Toggling Shortcuts], page 26.

If you find that Supercite's automatic filling is just too fragile for your tastes, you might consider one of these alternate approaches. Also, to make life easier, a shortcut function to toggle the state of both of these variables is provided on the key binding *C-c C-p C-p* (with the default value of `sc-mode-map-prefix`; see Chapter 9 [Post-yank Formatting Commands], page 25).

You will noticed that the minor mode string will show the state of these variables as qualifier characters. When both variables are `nil`, the Supercite minor mode string will display '`SC`'. When just `sc-auto-fill-region-p` is non-`nil`, the string will display '`SC:f`', and when just `sc-fixup-whitespace-p` is non-`nil`, the string will display '`SC:w`'. When both variables are non-`nil`, the string will display '`SC:fw`'. Note that the qualifiers chosen are mnemonics for the default bindings of the toggling function for each respective variable. See Section 9.3 [Variable Toggling Shortcuts], page 26.

Why are these variables not set to `nil` by default? It is because many users won't manually fill paragraphs that are Supercited, and there have been widespread complaints on the net about mail and news messages containing lines greater than about 72 characters. So the default is to fill cited text.

# 7 Selecting an Attribution

As you know, the attribution string is the part of the author's name that will be used to composed a non-nested citation string. Supercite scans the various mail headers present in the original article and uses a number of heuristics to extract strings which it puts into the *attribution association list* or *attribution alist*. This is analogous, but different than, the info alist previously mentioned. Each element in the attribution alist is a key-value pair containing such information as the author's first name, middle names, and last name, the author's initials, and the author's email terminus.

## 7.1 Attribution Preferences

When you cite an original message, you can tell Supercite which part of the author's name you would prefer it to use as the attribution. The variable `sc-preferred-attribution-list` controls this; it contains keys which are matched against the attribution alist in the given order. The first value of a key that produces a non-`nil`, non-empty string match is used as the attribution string, and if no keys match, a secondary mechanism is used to generate the attribution. See Section 7.2 [Anonymous Attributions], page 19.

The following preferences are always available in the attribution alist (barring error):

`"emailname"`
: the author's email terminus.

`"initials"`
: the author's initials.

`"firstname"`
: the author's first name.

`"lastname"`
: the author's last name.

`"middlename-1"`
: the author's first middle name.

`"sc-lastchoice"`
: the last attribution string you have selected. This is useful when you recite paragraphs in the reply.

`"sc-consult"`
: consults the customizable list `sc-attrib-selection-list` which can be used to select special attributions based on the value of any info key. See below for details.

`"x-attribution"`
: the original author's suggestion for attribution string choice. See below for details.

Middle name indexes can be any positive integer greater than zero, though it is unlikely that many authors will have more than one middle name, if that many.

At this point, let me digress into a discussion of etiquette. It is my belief that while the style of the citations is a reflection of the personal tastes of the replier (i.e., you), the attribution selection is ultimately the personal choice of the original author. In a sense it is his or her "net nickname", and therefore the author should have some say in the selection of attribution string. Imagine how you would feel if someone gave you a nickname that you didn't like?

For this reason, Supercite recognizes a special mail header, '`X-Attribution:`', which if present, tells Supercite the attribution string preferred by the original author. It is the value of this header that is associated with the `"x-attribution"` key in the attribution alist. Currently, you can override the preference of this key by changing `sc-preferred-attribution-list`, but that isn't polite, and in the future Supercite may hard-code this. For now, it is suggested that if you change the order of the keys in this list, that `"x-attribution"` always be first, or possible second behind only `"sc-lastchoice"`. This latter is the default.

The value `"sc-consult"` in `sc-preferred-attribution-list` has a special meaning during attribution selection. When Supercite encounters this preference, it begins processing a customizable list of attributions, contained in the variable `sc-attrib-selection-list`. Each element in this list contains lists of the following form:

```
(infokey ((regexp . attribution)
          (regexp . attribution)
          (...)))
```

where *infokey* is a key for `sc-mail-field` and *regexp* is a regular expression to match against the *infokey*'s value. If *regexp* matches the *infokey*'s value, the *attribution* is used as the attribution string. Actually, *attribution* can be a string or a list; if it is a list, it is `eval`uated and the return value (which must be a string), is used as the attribution.

This can be very useful for when you are replying to net acquaintances who do not use the '`X-Attribution:`' mail header. You may know what nickname they would prefer to use, and you can set up this list to match against a specific mail field, e.g., '`From:`', allowing you to cite your friend's message with the appropriate attribution.

## 7.2 Anonymous Attributions

When the author's name cannot be found in the '`From:`' mail header, a fallback author name and attribution string must be supplied. The fallback author name is contained in the variable `sc-default-author-name` and the fallback attribution string is contained in the variable `sc-default-attribution`. Default values for these variables are `"Anonymous"` and `"Anon"`, respectively. Note that in most circumstances, getting the default author name or attribution is a sign that something is set up incorrectly.

Also, if the preferred attribution, which you specified in your `sc-preferred-attribution-alist` variable cannot be found, a secondary method can be employed to find a valid attribution string. The variable `sc-use-only-preference-p` controls what happens in this case. If the variable's value is non-`nil`, then `sc-default-author-name` and `sc-default-attribution` are used, otherwise, the following steps are taken to find a valid attribution string, and the first step to return a non-`nil`, non-empty string becomes the attribution:

1. Use the last selected attribution, if there is one.
2. Use the value of the `"x-attribution"` key.

3. Use the author's first name.
4. Use the author's last name.
5. Use the author's initials.
6. Find the first non-`nil`, non-empty attribution string in the attribution alist.
7. `sc-default-attribution` is used.

Once the attribution string has been automatically selected, a number of things can happen. If the variable `sc-confirm-always-p` is non-`nil`, you are queried for confirmation of the chosen attribution string. The possible values for completion are those strings in the attribution alist, however you are not limited to these choices. You can type any arbitrary string at the confirmation prompt. The string you enter becomes the value associated with the `"sc-lastchoice"` key in the attribution alist.

Once an attribution string has been selected, Supercite will force the string to lower case if the variable `sc-downcase-p` is non-`nil`.

Two hook variables provide even greater control of the attribution selection process. The hook `sc-attribs-preselect-hook` is run before any attribution is selected. Likewise, the hook `sc-attribs-postselect-hook` is run after the attribution is selected (and the corresponding citation string is built), but before these values are committed for use by Supercite. During the post-selection hook, the local variables `attribution` and `citation` are bound to the appropriate strings. By changing these variables in your hook functions, you change the attribution and citation strings used by Supercite. One possible use of this would be to override any automatically derived attribution string when it is only one character long; e.g. you prefer to use `"initials"` but the author only has one name.

## 7.3 Author Names

Supercite employs a number of heuristics to decipher the author's name based on value of the '`From:`' mail field of the original message. Supercite can recognize almost all of the common '`From:`' field formats in use. If you encounter a '`From:`' field that Supercite cannot parse, please report this bug. See Chapter 13 [The Supercite Mailing List], page 33.

There are a number of Supercite variables that control how author names are extracted from the '`From:`' header. Some headers may contain a descriptive title as in:

```
From: computer!speedy!doe (John Xavier-Doe -- Decent Hacker)
```

Supercite knows which part of the '`From:`' header is email address and which part is author name, but in this case the string `"Decent Hacker"` is not part of the author's name. You can tell Supercite to ignore the title, while still recognizing hyphenated names through the use of a regular expression in the variable `sc-titlecue-regexp`. This variable has the default value of `"\\\\s +-+\\\\s +"`. Any text after this regexp is encountered is ignored as noise.

Some '`From:`' headers may contain extra titles in the name fields not separated by a title cue, but which are nonetheless not part of the author's name proper. Examples include the titles "Dr.", "Mr.", "Ms.", "Jr.", "Sr.", and "III" (e.g., Thurston Howe, the Third). Also, some companies prepend or append the name of the division, organization, or project on the author's name. All of these titles are noise which should be ignored. The variable `sc-name-filter-alist` is used for this purpose. As implied by its name, this variable is an association list, where each element is a cons cell of the form:

```
(regexp . position)
```

where *regexp* is a regular expression that is matched (using `string-match`) against each element of the '`From:`' field's author name. *position* is a position indicator, starting at zero. Thus to strip out all titles of "Dr.", "Mr.", etc. from the name, `sc-name-filter-alist` would have an entry such as:

```
("^\\(Mr\\|Mrs\\|Ms\\|Dr\\)[.]?$" . 0)
```

which only removes them if they appear as the first word in the name. The position indicator is an integer, or one of the two special symbols `last` or `any`. `last` always matches against the last word in the name field, while `any` matches against every word in the name field.

# 8 Configuring the Citation Engine

At the heart of Supercite is a regular expression interpreting engine called *Regi*. Regi operates by interpreting a data structure called a Regi-frame (or just *frame*), which is a list of Regi-entries (or just *entry*). Each entry contains a predicate, typically a regular expression, which is matched against a line of text in the current buffer. If the predicate matches true, an associated expression is `eval`uated. In this way, an entire region of text can be transformed in an *awk*-like manner. Regi is used throughout Supercite, from mail header information extraction, to header nuking, to citing text.

While the details of Regi are discussed below (see Section 8.1 [Using Regi], page 22), only those who wish to customize certain aspects of Supercite need concern themselves with it. It is important to understand though, that any conceivable citation style that can be described by a regular expression can be recognized by Supercite. This leads to some interesting applications. For example, if you regularly receive email from a co-worker that uses an uncommon citation style (say one that employs a '`|`' or '`}`' character at the front of the line), it is possible for Supercite to recognize this and *coerce* the citation to your preferred style, for consistency. In theory, it is possible for Supercite to recognize such things as uuencoded messages or C code and cite or fill those differently than normal text. None of this is currently part of Supercite, but contributions are welcome!

## 8.1 Using Regi

Regi works by interpreting frames with the function `regi-interpret`. A frame is a list of arbitrary size where each element is a entry of the following form:

`(`*pred* *func* `[`*negate-p* `[`*case-fold-search*`]])`

Regi starts with the first entry in a frame, evaluating the *pred* of that entry against the beginning of the line that '`point`' is on. If the *pred* evaluates to true (or false if the optional *negate-p* is non-`nil`), then the *func* for that entry is `eval`uated. How processing continues is determined by the return value for *func*, and is described below. If *pred* was false the next entry in the frame is checked until all entries have been matched against the current line. If no entry matches, '`point`' is moved forward one line and the frame is reset to the first entry.

*pred* can be a string, a variable, a list or one of the following symbols: `t`, `begin`, `end`, or `every`. If *pred* is a string, or a variable or list that `eval`uates to a string, it is interpreted as a regular expression. This regexp is matched against the current line, from the beginning, using `looking-at`. This match folds case if the optional *case-fold-search* is non-`nil`. If *pred* is not a string, or does not `eval`uate to a string, it is interpreted as a binary value (`nil` or non-`nil`).

The four special symbol values for *pred* are recognized:

`t` Always produces a true outcome.

`begin` Always executed before the frame is interpreted. This can be used to initialize some global variables for example.

`end` Always executed after frame interpreting is completed. This can be used to perform any necessary post-processing.

`every` Executes whenever the frame is reset, usually after the entire frame has been matched against the current line.

Note that *negate-p* and *case-fold-search* are ignored if *pred* is one of these special symbols. Only the first occurrence of each symbol in a frame is used; any duplicates are ignored. Also note that for performance reasons, the entries associated with these symbols are removed from the frame during the main interpreting loop.

Your *func* can return certain values which control continued Regi processing. By default, if your *func* returns `nil` (as it should be careful to do explicitly), Regi will reset the frame to the first entry, and advance '`point`' to the beginning of the next line. If a list is returned from your function, it can contain any combination of the following elements:

the symbol `continue`
: This tells Regi to continue processing entries after a match, instead of reseting the frame and moving '`point`'. In this way, lines of text can have multiple matches, but you have to be careful to avoid entering infinite loops.

the symbol `abort`
: This tells Regi to terminate frame processing. However, any `end` entry is still processed.

the list `(frame . `*newframe*`)`
: This tells Regi to substitute *newframe* as the frame it is interpreting. In other words, your *func* can modify the Regi frame on the fly. *newframe* can be a variable containing a frame, or it can be the frame in-lined.

the list `(step . `*step*`)`
: Tells Regi to move *step* number of lines forward as it continues processing. By default, Regi moves forward one line. *step* can be zero or negative of course, but watch out for infinite loops.

During execution of your *func*, the following variables will be temporarily bound to some useful information:

`curline` The current line in the buffer that Regi is `looking-at`, as a string.

`curframe` The current frame being interpreted.

`curentry` The current frame entry being interpreted.

## 8.2 Frames You Can Customize

As mentioned earlier, Supercite uses various frames to perform certain jobs such as mail header information extraction and mail header nuking. However, these frames are not available for you to customize, except through abstract interfaces such as `sc-nuke-mail-header`, et al.

However, the citation frames Supercite uses provide a lot of customizing power and are thus available to you to change to suit your needs. The workhorse of citation is the frame contained in the variable `sc-default-cite-frame`. This frame recognizes many situations, such as blank lines, which it interprets as paragraph separators. It also recognizes previously cited nested and non-nested citations in the original message. By default it will coerce non-nested citations into your preferred citation style, and it will add a level of citation to nested citations. It will also simply cite uncited lines in your preferred style.

In a similar vein, there are default frames for *unciting* and *reciting*, contained in the variables `sc-default-uncite-frame` and `sc-default-recite-frame` respectively.

As mentioned earlier (see Section 2.2 [Recognizing Citations], page 5), citations are recognized through the values of the regular expressions `sc-citation-root-regexp`, et al. To recognize odd styles, you could modify these variables, or you could modify the default citing frame. Alternatively, you could set up association lists of frames for recognizing specific alternative forms.

For each of the actions – citing, unciting, and reciting – an alist is consulted to find the frame to use (`sc-cite-frame-alist`, `sc-uncite-frame-alist`, and `sc-recite-frame-alist` respectively). These frames can contain alists of the form:

```
((infokey (regexp . frame) (regexp . frame) ...)
 (infokey (regexp . frame) (regexp . frame) ...)
 (...))
```

Where *infokey* is a key suitable for `sc-mail-field`, *regexp* is a regular expression which is `string-match`'d against the value of the `sc-mail-field` key, and *frame* is the frame to use if a match occurred. *frame* can be a variable containing a frame or a frame in-lined.

When Supercite is about to cite, uncite, or recite a region, it consults the appropriate alist and attempts to find a frame to use. If one is not found from the alist, then the appropriate default frame is used.

# 9 Post-yank Formatting Commands

Once the original message has been yanked into the reply buffer, and `sc-cite-original` has had a chance to do its thing, a number of useful Supercite commands will be available to you. Since there is wide variety in the keymaps that MUAs set up in their reply buffers, it is next to impossible for Supercite to properly sprinkle its commands into the existing keymap. For this reason Supercite places its commands on a separate keymap, putting this keymap onto a prefix key in the reply buffer. You can customize the prefix key Supercite uses by changing the variable `sc-mode-map-prefix`. By default, the `sc-mode-map-prefix` is *C-c C-p*; granted, not a great choice, but unfortunately the best general solution so far. In the rest of this chapter, we'll assume you've installed Supercite's keymap on the default prefix.

## 9.1 Commands to Manually Cite, Recite, and Uncite

Probably the three most common post-yank formatting operations that you will perform will be the manual citing, reciting, and unciting of regions of text in the reply buffer. Often you may want to recite a paragraph to use a nickname, or manually cite a message when setting `sc-cite-region-limit` to `nil`. The following commands perform these functions on the region of text between '`point`' and '`mark`'. Each of them sets the *undo boundary* before modifying the region so that the command can be undone in the standard Emacs way.

A quick note about Emacs 19. Unlike in Emacs 18, the region delimited by '`point`' and '`mark`' can have two states. It can be *active* or *inactive*. Although Emacs 19 and Lucid Emacs 19 use different terminology and functions, both employ the same convention such that when the region is inactive, commands that modify the region should generate an error. The user needs to explicitly activate the region before successfully executing the command. All Supercite commands conform to this convention.

Here is the list of Supercite citing commands:

`sc-cite-region` (*C-c C-p c*)
: This command cites each line in the region of text by interpreting the selected frame from `sc-cite-frame-alist`, or the default citing frame `sc-default-cite-frame`. It runs the hook `sc-pre-cite-hook` before interpreting the frame. With an optional universal argument (*C-u*), it temporarily sets `sc-confirm-always-p` to `t` so you can confirm the attribution string for a single manual citing. See Chapter 8 [Configuring the Citation Engine], page 22.

`sc-uncite-region` (*C-c C-p u*)
: This command removes any citation strings from the beginning of each cited line in the region by interpreting the selected frame from `sc-uncite-frame-alist`, or the default unciting frame `sc-default-uncite-frame`. It runs the hook `sc-pre-uncite-hook` before interpreting the frame. See Chapter 8 [Configuring the Citation Engine], page 22.

`sc-recite-region` (*C-c C-p r*)
: This command recites each line the region by interpreting the selected frame from `sc-recite-frame-alist`, or the default reciting frame

sc-default-recite-frame. It runs the hook sc-pre-recite-hook before interpreting the frame. See Chapter 8 [Configuring the Citation Engine], page 22.

Supercite will always ask you to confirm the attribution when reciting a region, regardless of the value of sc-confirm-always-p.

## 9.2 Insertion Commands

These two functions insert various strings into the reply buffer.

sc-insert-reference (*C-c C-p w*)
: Inserts a reference header into the reply buffer at 'point'. With no arguments, the header indexed by sc-preferred-header-style is inserted. An optional numeric argument is the index into sc-rewrite-header-list indicating which reference header to write.

  With just the universal argument (*C-u*), electric reference mode is entered, regardless of the value of sc-electric-references-p.

sc-insert-citation (*C-c C-p i*)
: Inserts the current citation string at the beginning of the line that 'point' is on. If the line is already cited, Supercite will issue an error and will not cite the line.

## 9.3 Variable Toggling Shortcuts

Supercite defines a number of commands that make it easier for you to toggle and set various Supercite variables as you are editing the reply buffer. For example, you may want to turn off filling or whitespace cleanup, but only temporarily. These toggling shortcut commands make this easy to do.

Like Supercite commands in general, the toggling commands are placed on a keymap prefix within the greater Supercite keymap. For the default value of sc-mode-map-prefix, this will be *C-c C-p C-t*.

The following commands toggle the value of certain Supercite variables which take only a binary value:

*C-c C-p C-t b*
: Toggles the variable sc-mail-nuke-blank-lines-p.

*C-c C-p C-t c*
: Toggles the variable sc-confirm-always-p.

*C-c C-p C-t d*
: Toggles the variable sc-downcase-p.

*C-c C-p C-t e*
: Toggles the variable sc-electric-references-p.

*C-c C-p C-t f*
: Toggles the variable sc-auto-fill-region-p.

*C-c C-p C-t o*
: Toggles the variable `sc-electric-circular-p`.

*C-c C-p C-t s*
: Toggles the variable `sc-nested-citation-p`.

*C-c C-p C-t u*
: Toggles the variable `sc-use-only-preferences-p`.

*C-c C-p C-t w*
: Toggles the variable `sc-fixup-whitespace-p`.

The following commands let you set the value of multi-value variables, in the same way that Emacs' `set-variable` does:

*C-c C-p C-t a*
: Sets the value of the variable `sc-preferred-attribution-list`.

*C-c C-p C-t l*
: Sets the value of the variable `sc-cite-region-limit`.

*C-c C-p C-t n*
: Sets the value of the variable `sc-mail-nuke-mail-headers`.

*C-c C-p C-t N*
: Sets the value of the variable `sc-mail-header-nuke-list`.

*C-c C-p C-t p*
: Sets the value of the variable `sc-preferred-header-style`.

One special command is provided to toggle both `sc-auto-fill-region-p` and `sc-fixup-whitespace-p` together. This is because you typically want to run Supercite with either variable as `nil` or non-`nil`. The command to toggle these variables together is bound on *C-c C-p C-p*.

Finally, the command *C-c C-p C-t h* (also *C-c C-p C-t ?*) brings up a Help message on the toggling keymap.

## 9.4 Mail Field Commands

These commands allow you to view, modify, add, and delete various bits of information from the info alist. See Chapter 3 [Information Keys and the Info Alist], page 6.

`sc-mail-field-query` (*C-c C-p f*)
: Allows you to interactively view, modify, add, and delete info alist key-value pairs. With no argument, you are prompted (with completion) for a info key. The value associated with that key is displayed in the minibuffer. With an argument, this command will first ask if you want to view, modify, add, or delete an info key. Viewing is identical to running the command with no arguments.

: If you want to modify the value of a key, Supercite will first prompt you (with completion) for the key of the value you want to change. It will then put you in the minibuffer with the key's current value so you can edit the value as you wish. When you hit ⟨RET⟩, the key's value is changed. For those of you running Emacs 19, minibuffer history is kept for the values.

If you choose to delete a key-value pair, Supercite will prompt you (with completion) for the key to delete.

If you choose to add a new key-value pair, Supercite firsts prompts you for the key to add. Note that completion is turned on for this prompt, but you can type any key name here, even one that does not yet exist. After entering the key, Supercite prompts you for the key's value. It is not an error to enter a key that already exists, but the new value will override any old value. It will not replace it though; if you subsequently delete the key-value pair, the old value will reappear.

`sc-mail-process-headers` (*C-c C-p g*)
: This command lets you re-initialize Supercite's info alist from any set of mail headers in the region between '`point`' and '`mark`'. This function is especially useful for replying to digest messages where Supercite will initially set up its information for the digest originator, but you want to cite each component article with the real message author. Note that unless an error during processing occurs, any old information is lost.

## 9.5 Miscellaneous Commands

`sc-open-line` (*C-c C-p o*)
: Similar to Emacs' standard `open-line` commands, but inserts the citation string in front of the new line. As with `open-line`, an optional numeric argument inserts that many new lines.

`sc-describe` (*C-c C-p h* and *C-c C-p ?*)
: This function has been obsoleted by the TEXinfo manual you are now reading. It is still provided for compatibility, but it will eventually go away.

`sc-version` (*C-c C-p v*)
: Echos the version of Supercite you are using. With the optional universal argument (*C-u*), this command inserts the version information into the current buffer.

`sc-submit-bug-report` (*C-c C-p C-b*)
: If you encounter a bug, or wish to suggest an enhancement, use this command to set up an outgoing mail buffer, with the proper address to the Supercite maintainer automatically inserted in the '`To:`' field. This command also inserts information that the Supercite maintainer can use to recreate your exact setup, making it easier to verify your bug.

# 10 Hints to MUA Authors

In June of 1989, some discussion was held between the various MUA authors, the Supercite author, and other Supercite users. These discussions centered around the need for a standard interface between MUAs and Supercite (or any future Supercite-like packages). This interface was formally proposed by Martin Neitzel on Fri, 23 Jun 89, in a mail message to the Supercite mailing list:

```
Martin> Each news/mail-reader should provide a form of
Martin> mail-yank-original that

Martin> 1: inserts the original message incl. header into the
Martin>    reply buffer; no indentation/prefixing is done, the header
Martin>    tends to be a "full blown" version rather than to be
Martin>    stripped down.

Martin> 2: ‘point’ is at the start of the header, ‘mark’ at the
Martin>    end of the message body.

Martin> 3: (run-hooks ’mail-yank-hooks)

Martin> [Supercite] should be run as such a hook and merely
Martin> rewrite the message.  This way it isn’t anymore
Martin> [Supercite]’s job to gather the original from obscure
Martin> sources. [...]
```

This specification was adopted, but with the recent release of Emacs 19, it has undergone a slight modification. Instead of the variable `mail-yank-hooks`, the new preferred hook variable that the MUA should provide is `mail-citation-hook`. `mail-yank-hooks` can be provided for backward compatibility, but `mail-citation-hook` should always take precedence. Richard Stallman (of the FSF) suggests that the MUAs should `defvar mail-citation-hook` to `nil` and perform some default citing when that is the case. Take a look at Emacs 19's ‘`sendmail.el`’ file, specifically the `mail-yank-original` defun for details.

If you are writing a new MUA package, or maintaining an existing MUA package, you should make it conform to this interface so that your users will be able to link Supercite easily and seamlessly. To do this, when setting up a reply or forward buffer, your MUA should follow these steps:

1. Insert the original message, including the mail headers into the reply buffer. At this point you should not modify the raw text in any way, and you should place all the original headers into the body of the reply. This means that many of the mail headers will be duplicated, one copy above the `mail-header-separator` line and one copy below, however there will probably be more headers below this line.
2. Set ‘`point`’ to the beginning of the line containing the first mail header in the body of the reply. Set ‘`mark`’ at the end of the message text. It is very important that the region be set around the text Supercite is to modify and that the mail headers are within this region. Supercite will not venture outside the region for any reason, and anything within the region is fair game, so don't put anything that **must** remain unchanged inside the region. Further note that for Emacs 19, the region need not be

set active. Supercite will work properly when the region is inactive, as should any other like-minded package.

3. Run the hook `mail-citation-hook`. You will probably want to provide some kind of default citation functions in cases where the user does not have Supercite installed. By default, your MUA should `defvar` `mail-citation-hook` to `nil`, and in your yanking function, check its value. If it finds `mail-citation-hook` to be `nil`, it should perform some default citing behavior. User who want to connect to Supercite then need only add `sc-cite-original` to this list of hooks using `add-hook`.

If you do all this, your users will not need to overload your routines to use Supercite, and your MUA will join the ranks of those that conform to this interface "out of the box."

# 11 Version 3 Changes

With version 3, Supercite has undergone an almost complete rewrite, and has hopefully benefited in a number of ways, including vast improvements in the speed of performance, a big reduction in size of the code and in the use of Emacs resources, and a much cleaner and flexible internal architecture. The central construct of the info alist, and its role in Supercite has been expanded, and the other central concept, the general package Regi, was developed to provide a theoretically unlimited flexibility.

But most of this work is internal and not of very great importance to the casual user. There have been some changes at the user-visible level, but for the most part, the Supercite configuration variables from version 2 should still be relevant to version 3. Below, I briefly outline those user-visible things that have changed since version 2. For details, look to other sections of this manual.

1. Supercite proper now comes in a single file, '`supercite.el`', which contains everything except the unsupported noodlings, overloading (which should be more or less obsolete with the release of Emacs 19), and the general lisp packages '`reporter.el`' and '`regi.el`'. Finally, the TeXinfo manual comes in its own file as well. In particular, the file '`sc.el`' from the version 2 distribution is obsolete, as is the file '`sc-elec.el`'.
2. `sc-spacify-name-chars` is gone in version 3.
3. `sc-nickname-alist` is gone in version 3. The `sc-attrib-selection-list` is a more general construct supporting the same basic feature.
4. The version 2 variable `sc-preferred-attribution` has been changed to `sc-preferred-attribution-list`, and has been expanded upon to allow you to specify an ordered list of preferred attributions.
5. `sc-mail-fields-list` has been removed, and header nuking in general has been greatly improved, giving you wider flexibility in specifying which headers to keep and remove while presenting a simplified interface to commonly chosen defaults.
6. Post-yank paragraph filling has been completely removed from Supercite, other packages just do it better than Supercite ever would. Supercite will still fill newly cited paragraphs.
7. The variable `sc-all-but-cite-p` has been replaced by `sc-cite-region-limit`.
8. Keymap hacking in the reply buffer has been greatly simplified, with, I believe, little reduction in functionality.
9. Hacking of the reply buffer's docstring has been completely eliminated.

# 12 Thanks and History

The Supercite package was derived from its predecessor Superyank 1.11 which was inspired by various bits of code and ideas from Martin Neitzel and Ashwin Ram. They were the folks who came up with the idea of non-nested citations and implemented some rough code to provide this style. Superyank and Supercite version 2 evolved to the point where much of the attribution selection mechanism was automatic, and features have been continuously added through the comments and suggestions of the Supercite mailing list participants. Supercite version 3 represents a nearly complete rewrite with many of the algorithms and coding styles being vastly improved. Hopefully Supercite version 3 is faster, smaller, and much more flexible than its predecessors.

In the version 2 manual I thanked some specific people for their help in developing Supercite 2. You folks know who you are and your continued support is greatly appreciated. I wish to thank everyone on the Supercite mailing list, especially the brave alpha testers, who helped considerably in testing out the concepts and implementation of Supercite version 3. Special thanks go out to the MUA and Emacs authors Kyle Jones, Stephen Gildea, Richard Stallman, and Jamie Zawinski for coming to a quick agreement on the new `mail-citation-hook` interface, and for adding the magic lisp to their code to support this.

All who have helped and contributed have been greatly appreciated.

# 13 The Supercite Mailing List

The author runs a simple mail expanding mailing list for discussion of issues related to Supercite. This includes enhancement requests, bug reports, general help questions, etc. To subscribe or unsubscribe to the mailing list, send a request to the administrative address:

`supercite-request@python.org`

Please be sure to include the most reliable and shortest (preferably Internet) address back to you. To post articles to the list, send your message to this address (you do not need to be a member to post, but be sure to indicate this in your article or replies may not be CC'd to you):

`supercite@python.org`

If you are sending bug reports, they should go to the following address, but *please*! use the command `sc-submit-bug-report` since it will be much easier for me to duplicate your problem if you do so. It will set up a mail buffer automatically with this address on the '`To:`' line:

`supercite-help@python.org`

# Concept Index

## R



## S



## T



## U



## Y

# Command Index

Since all supercite commands are prepended with the string "sc-", each appears under its sc-*command* name and its *command* name.

## U



## V

# Key Index

# Variable Index

Since all supercite variables are prepended with the string "`sc-`", each appears under its `sc-`*variable* name and its *variable* name.

## S



## T



## U

# Short Contents

# Table of Contents