Reference Manual

See also the introductory document, which currently only exists in html.

1 Commands

To assign a new meaning to a key, use the bind statement:

```
bind some-command key-description
```

The command name should be all lowercase, as .sheetsrc files are case sensitive.

Most key descriptions take the form of a modifier key (like control or shift) followed by the main key to be pressed. To specify the control key, write `ctrl-` in front of the main key; write `shift-` for the shift key. The two can be combined, as in `ctrl-shift-F1.` The name of the main key is always uppercase. For alphanumeric keys, the name of the key is the key itself. For other keys, we use the following abbreviations:

`UP, DOWN, LEFT, RIGHT, PGUP, PGDN, HOME, END, ENTER, BACKSPACE, DELETE, TAB, BACKSLASH, LBRACKET, RBRACKET, F1-F12`

Java refuses to recognize some key combinations, so don't be surprised if you can't assign a command to a not-so-ordinary key. Also, Sheets only allows rebindings involving alphanumeric keys if the ctrl key is specified. (In other words, you can rebind `ctrl-A,` but not `A` or `shift-A`) Rebinding the mouse buttons is not currently supported.

This is a list of all commands you can bind to a key. Most, but not all, of these commands have default key bindings.

1.1 Cursor Motion

forward-char [RIGHT]

Move cursor forward (right) one character

forward-char-select

Move cursor forward (right) one character while extending/shrinking the text selection

forward-word

Move cursor forward (right) one word

forward-word-select

Move cursor forward (right) one word while extending/shrinking the text selection

backward-char

Move cursor back (left) one character

backward-char-select

Move cursor back (left) one character while extending/shrinking the text selection.

backward-word

Move cursor back (left) one word

backward-word-select

Move cursor back (left) one word while extending/shrinking the text selection

up-line

Move cursor up one line

up-line-select

Move cursor up one line while extending/shrinking the text selection

down-line

Move cursor down one line

down-line-select

Move cursor down one line while extending/shrinking the text selection

start-line

Move cursor to beginning of line

start-line-select

Move cursor to beginning of line while extending the text selection.

end-line

Move cursor to end of line

end-line-select

Move cursor to end of line while extending the text selection.

backward-selection

Moves to an earlier selection within the current window. Repeated usage will take you to successively older selections. Use forward-selection to go the other way.

forward-selection

Moves to an more recent selection within the current window. Repeated usage will take you to successively newer selections. Use backward-selection to go the other way.

up-fragment

Move cursor up one fragment, leaving the cursor at the end of the new fragment.

down-fragment

Move cursor down one fragment, leaving the cursor at the beginning of the new fragment.

start-fragment

Move cursor to the beginning of fragment.

start-fragment-select

Move cursor to the beginning of fragment selecting the text between the current position and the start.

end-fragment

Move the cursor to the end of the current fragment.

end-fragment-select

Move the cursor to the end of the current fragment selecting text between the current position and the end.

start-sheet

Move cursor to the beginning of the top-level sheet in the current edit window.

end-sheet

Move the cursor to the end of the top-level sheet in the current edit window.

1.2 Scrolling

scroll-up-page

Scroll the window up a page without moving the cursor.

scroll-up-line

Scroll the window up a line without moving the cursor.

scroll-down-page

Scroll the window down a page without moving the cursor.

scroll-down-line

Scroll the window down a line without moving the cursor.

1.3 Miscellaneous Editing Commands

backward-char-delete

Deletes the character before the cursor.

forward-char-delete

Deletes the character after the cursor.

cut-text

Cuts the selected text into the clipboard.

copy-text

Copies the selected text into the clipboard.

paste-text

Interprets the contents of the clipboard as text, and inserts it where the text cursor is.

select-word

Select the word under the cursor

split-line

Splits a line into two lines, and moves the cursor to the beginning of the second line. (In other words, this is the command for the ENTER key)

open-line

Like split-line, except doesn't move the cursor.

kill-line

Corresponds to the EMACS command of the same name. If at the end of the line, it deletes the newline. Otherwise, it deletes all characters between the cursor and the end of the line. (Unlike the Emacs command, our kill-line doesn't put anything in the cut buffer)

indent-line

Indents the line according to the automatic code indentation rules.

wrap-paragraph

Do line wrapping (filling) on an ASCII text paragraph. Useful primarily on comments, since DocSheet paragraphs are automatically wrapped.

local-undo

Undoes the last edit command performed on the selected fragment. Multiple undo is supported.

local-redo

Redoes the last edit command on the selected fragment that was undone.

show-replace-dialog

Displays the `Search and replace` dialog.

1.4 Fragment-based commands

cut-fragment

Cuts the selected fragment(s) into the clipboard.

copy-fragment

Copies the selected fragment(s) into the clipboard.

paste-fragment

Interprets the contents of the clipboard as fragments, and inserts them before the selected fragment. Does nothing if the contents of the clipboard can't be interpreted as fragments.

remove-fragment

Removes the selected fragment(s) from the current sheet, but does not affect any other references. See removing and destroying.

destroy-fragment

Destroys the selected fragments, removing them from all sheets and attributes where they are currently referenced. See removing and destroying..

commit-fragment-edit

Commits the edit mode on the current fragment. This is equivalent to pushing the "Commit" button.

abort-fragment-edit

Equivalent to hitting the "Abort" button for this fragment.

begin-fragment-edit

Makes sure that the current fragment is in edit mode.

toggle-edit-mode

If the selected fragment is being edited, commits the edit. Otherwise, begins editing the selected fragment.

show-all-attributes

Shows all of the attributes for the selected fragment.

hide-all-attributes

Stop displaying any attributes on the current fragment. (Note -- this must be run from the fragment itself, rather than from one of the attributes.)

1.5 Container-based Commands

show-container-properties-dialog

Show the "properties" dialog for the selected (or enclosing) container. This will allow you to set the title, export properties, etc.

commit-container-edits

Finds all edits in progress in the current container, and commits them.

abort-container-edits

Finds all edits in progress in the current container, and aborts them.

clone-sheet

Creates a temporary copy of the current sheet. The contents will be the same, but other special properties (such as constraints or graphability) will be left behind.

recursive-flatten

Creates a new temporary sheet which contains all of the fragments nested (at any depth) underneath the selected container. (Note that it will not remove the fragments from the container itself -- it simply creates new references to them.)

sort-sheet-alphabetic

Sorts all of the fragments on the current sheet alphabetically by name. If the current sheet is "permanent" it will first be copied into a temporary sheet.

sort-sheet-java

Sorts all of the fragments on the current sheet, by package, class, kind and name. If the current sheet is "permanent" it will first be copied into a temporary sheet.

change-project-of-everything-reachable

Recursively walks through every fragment within the current (or enclosing) container, and set it be in the same project. (However, remember that Java fragments usually take their project from the package rather than being set directly. They will therefore only be indirecly affected by this command.)

change-views-to-full

Changes every viewer within the current (or enclosing) container to be in the "full" view, showing the entire contents of each fragment.

change-views-to-header

Changes every viewer within the current (or enclosing) container to be in the "header" view, showing the interfaces of each fragment.

change-views-to-summary

Changes every viewer within the current (or enclosing) container to be in the "summary" view, showing just one line summaries of each fragment.

1.6 Creating New Fragments

See creating fragments in the Sheets reference for general information.

extend-fragment

Tries to create a new fragment like the currently selected one. Will not work for every sort of fragment.

insert-java-fragment

Insert a new java fragment.

insert-separator

Insert a seperator fragment (a horizontal line).

insert-sheet

Insert a new sheet at the current location. This will bring up the sheet properties dialog.

insert-image

Insert a new image fragment. The image lives in a file outside the Sheets database. The file name is stored in the `filename` attribute.

insert-text-fragment

Insert a new ASCII text fragment. These fragments are deprecated, you should use DocSheet paragraphs instead.

The following commands all create various XML fragment types:

insert-definition

Inserts a new DocSheet definition fragment.

insert-edit-command

Inserts a new DocSheet edit command fragment. (Note, this is probably only useful for Sheets implementors.)

insert-edit-variable

Inserts a new DocSheet edit variable fragment. (Note, this is probably only useful for Sheets implementors.)

insert-list

Inserts a new DocSheet list fragment. You control whether it is an ordered or unordered list via the "ordered" attribute.

insert-paragraph

Inserts a new DocSheet ordinary paragraph.

insert-source

Inserts a new DocSheet "preformatted" paragraph

insert-section

Inserts a new DocSheet section. (If it is a top-level section, you may wish to export it via the "properties" dialog.)

1.7 Navigation

show-query-dialog

Displays the Query dialog, and defaults to `text search` on the full view.

show-query-dialog-stringref

Displays the Query dialog, and defaults to `text search` on the name view.

show-query-dialog-stringdef

Displays the Query dialog, and defaults to `text search` on the full view.

show-query-dialog-javadef

Displays the Query dialog, and defaults to `Definitions of` search.

show-query-dialog-javaref

Displays the Query dialog, and defaults to `References to name` search.

show-query-dialog-javacomp

Displays the Query dialog, and defaults to `References to the selected fragment search`.

show-query-dialog-javamembers

Displays the Query dialog, and defaults to `Members of a class` search.

show-query-dialog-javahierarchy

Displays the Query dialog, and defaults to `Hierarchy Query`.

goto-word

Searches for all fragments whose name matches the current word. This is typically slower and less accurate than middle-clicking, but it can find "near matches" as well as exact matches.

goto-word-references

Finds all Java fragments which reference the word under the cursor. Deprecated, use `goto-context` (middle click) and then goto-fragment-references.

goto-word-definitions

Finds all Java fragments which have the same name as the word under the cursor. Deprecated, use `goto-context` (middle-click).

goto-fragment-references

Finds all Java fragments which reference the selected fragment.

show-fragment-in-context

Finds all containers in which the selected fragment is contained. Successive executions will cycle through all containers.

show-highlight-dialog

Displays the `Hightlight String` dialog.

copy-fragment-to-temp-sheet

Creates a new temporary sheet holding all of the selected fragments. Not really a navigation command, but it often used after a navigation/query command leaves you with multiple selections on a sheet.

1.8 File Commands

auto-sync : boolean = true

If true, bring the database up to date after each command that modifies it. This insures that changes won't be lost on a crash, but can cause a noticable slowdown when working on large databases.

show-new-project-dialog

Pops up a dialog allowing you to import or create a new project. Use this for starting new programs or packages or for importing external sources.

show-import-dialog

Displays the import dialog used to fetch Java code or DocSheet XML documentation into the database.

export-dirty-fragments

Export all of the Java fragments that have been modified or that might be affected by some modification.

export-all-fragments

Export all fragments that can be exported, including DocSheet fragments.

show-save-dump-file-dialog

Show the dialog for saving dump files for all the projects currently in the database.

1.9 Window Management

show-projects-sheet

Show the sheet listing all the loaded projects.

show-affected-by-sheet

Show the affected-by sheet.

show-edit-sheet

Show the window displaying all the fragments currently being edited.

close-window

Close the current window. If it is viewing a temporary sheet, the sheet will also be destroyed.

redraw-window

Repaints the contents of the window. You shouldn't ever have to do this, but bugs to happen.

exit-program

Exits from sheets, closing all windows and syncing the database.

1.10 Compilation

(For setup information, see the compile-command variable and the introduction to compilation.)

compile-program

Compiles the program, doing the minimal amount of recompilation necessary.

recompile-program

Recompiles the program from scratch.

show-compile-dialog

Displays the `Compile Program` dialog.

1.11 Java Specific Commands

java-documentation-url : String = (Sun's web site)

The location of Java HTML documentation, in JavaDoc format. The format of this field is a sequence of PairOfStrings, where the first string is a package and the second is the URL to use on that package. If a package name ends with ".*", we do globbing, so that the user can specify the location of java.* and not have to list every last package.To find documentation, Sheets will take this value and append the fully qualified name of the

class plus ".html".

In order to allow pairs of strings to be specified, we make another creative extension to the `set` statement:

```
set java-documentation-url["java.*"] "http://some-url"
```

complete-java-word

Completes the identifier under the text cursor. See section on word completion for details.

goto-documentation

Launches a web browser to display the documentation for the selected fragment.

goto-documentation-for-selected-fragment

Tries to find a web page which describes the current fragment and show it to you in a Web browser.

graph-java-hierarchy

Performs a hierarchy query on the selected fragment, and graphs the results.

graph-selected-fragments

Attempts to show a graph (or "forest") of all the selected Java fragments, plus their parents and descendents. (Note that Sheets is not currently capable of displaying "forests", so you are better off using "graph-java-hierarchy".

make-java-class-table

Creates a "table" displaying common properties for the selected Java classes. (Note: This is an experimental feature.)

make-java-vtable

Creates a "table" displaying methods and inheritance for all subclasses of the selected Java class. (Note: This is an experimental feature.)

comment-out-selection

Comments out all lines within the selected range of a Java comment. If you haven't selected a range, or if you have selected several fragments, then the entire fragments will be commented out.

uncomment-selection

Tries to remove the comments from the select range of lines in a Java fragment, or from multiple selected Java fragments.

transform-field-to-method

Edits the current Java field fragment so that it has a procedural interface instead. (You still have the choice to modify the generated methods or abort rather than committing the

edits.)

extend-class-methods

Edits a Java class fragment to contain "stub" routines for inherited methods.

show-software-metrics-dialog

Analyzes the Java code in the current database and displays reasonably comprehensive software metrics.

1.12 XML (DocSheet) Commands

create-link

Creates an explicit user link within an XML (DocSheet) fragment. The selected text is made into a link to whatever fragment(s) are in the cut buffer.

convert-urls-to-native-links

Converts HTML "URL-style" links to native Sheets links. Traditionally, you use this after importing an HTML file into Sheets and then selecting all imported fragment.

strip-anchors

Clears out the "anchors" attribute of an XML (DocSheet) fragment.

transform-fragment-to-definition

Tries to transform the current fragment into an XML (DocSheet) fragment.

transform-fragment-to-edit-command

Tries to transform the current fragment into an XML (DocSheet) edit command fragment. (Note -- this is probably only useful for Sheets implementors.)

transform-fragment-to-edit-variable

Tries to transform the current fragment into an XML (DocSheet) edit variable fragment. (Note -- this is probably only useful for Sheets implementors.)

transform-fragment-to-list

Tries to transform the current fragment into an XML (DocSheet) list fragment.

transform-fragment-to-paragraph

Tries to transform the current fragment into an XML (DocSheet) paragraph fragment.

transform-fragment-to-preformatted

Tries to transform the current fragment into an XML (DocSheet) preformatted paragraph fragment.

transform-fragment-to-section

Tries to transform the current fragment into an XML (DocSheet) section fragment.

transform-fragment-to-sheet

Tries to transform an XML (DocSheet) container fragment into an ordinary sheet.

transform-fragment-to-text-fragment

Tries to transform an XML (DocSheet) paragraph into an "ordinary" textual fragment. (Note -- at this point there is probably no reason to use these fragments.)

demote-doc-fragment

An experimental command which tries to move or transfrom an XML (DocSheet) fragment into something "less important".

promote-doc-fragment

Attempts to make a XML (DocSheet) fragment "more important" by transforming or moving it. (Note: This is command should be considered experimental.)

1.13 Miscellaneous

show-key-bindings-dialog

Pops up a dialog listing all commands and their key bindings. This also allows you to look up the documentation for any command.

show-global-undo-dialog

Pops up a dialog which allows you to undo (or potentially redo) fragment- and container-level commands.

check-consistency

Checks the consistency of your database. This is probably only useful for sheets maintainers.

2 Variables

In a .sheetsrc file, you can change the value of a variable by writing

```
set variable-name value
```

The name of the variable should be in lowercase, as .sheetsrc files are case sensitive. Different variables have different sets of legal values. For boolean variables, the value should be either `true` or `false`. For string variables, the string should be surrounded by double quotes (`"string"`).

The following is a list of variables that can be set in a `.sheetsrc` file; default values are given after the equal sign.

2.1 Compilation

To compile programs within Sheets, you will need to set the following variables in your .sheetsrc file:

compile-command : String = "javac %java%" recompile-command : String = "javac %JAVA%"

These are the default command lines which will be used by the compile-program and recompile-program commands. If you have a more complex program, you can also use it to invoke a "make" utility. See sections on compilation and makefiles for details.

Note to non-Windows users: Java requires that you specify the full path to the executable file.

Note to Windows users: Because of limitations in some Windows virtual machines, command output may not be captured properly. We therefore provide a program named "wrapper.exe" which will redirect ouput properly. If `Sheets` detects that you are running on a windows architecture it will automatically wrap your compile commands with this program. As a pleasant side-effect, the wrapper program knows about your `path` environment variable, and thus you don't have to specify the full path of your program.

error-pattern : String = (special)

A regular expression which is used to interpret compiler error messages. The parenthesized sub-patterns should match the filename, line-number, and the actual error message. This is a magic variable: you can set it multiple times, and the result is cumulative. Values are automatically installed for Sun's javac and for Microsoft's jvc.

2.2 Documentation

java-documentation-url : String = (Sun's web site)

The location of Java HTML documentation, in JavaDoc format. The format of this field is a sequence of PairOfStrings, where the first string is a package and the second is the URL to use on that package. If a package name ends with ".*", we do globbing, so that the user can specify the location of java.* and not have to list every last package.To find documentation, Sheets will take this value and append the fully qualified name of the class plus ".html".

In order to allow pairs of strings to be specified, we make another creative extension to the `set` statement:

```
set java-documentation-url["java.*"] "http://some-url"
```

web-browser : String = (no useful default)

The command line to run the web browser. Note that you should specify the full path to the executable file, as required by Java.

sheets-documentation-url : String = "SheetsHome/doc/reference.html"

The location of the Sheets reference manual. This is used by the `Help` dialog when you select a command and click the `Help` button.

2.3 Miscellaneous Configuration

beep-file : String = "SheetsHome/beep.au"

The absolute pathname of a sound file (in AU format) that is used as the "beep" sound for commands that can't be executed. If this variable is not set, you will get a visible warning instead.

favorite-project : String = "unknown"

If for some reason, Sheets feels a need to guess which project a fragment should go in, this value is used. If you are having problems with stuff showing up in the "`unknown` project, you may want to set this to the name of the project you are working on.

auto-sync : boolean = true

If true, bring the database up to date after each command that modifies it. This insures that changes won't be lost on a crash, but can cause a noticable slowdown when working on large databases.

2.4 Preferences

These variables are called preferences in that sheets will work as intended if you don't set any of them, however you may find that you very much want to set some of them.

emacs-bindings : boolean = false

Whether or not to use sort of vaguely Emacs-like keybindings. If you set this variable, you should do so **before** any bind statement.

default-sheet-view : word = header

The default view for fragments when displayed on a sheet. Since not all fragments support all views, there is actually a prioritized list of default views. When you use `set` on this variable, what it actually does is add that view to the head of the list.

Most fragments do recognize the views "`summary`", "`header`", and "`full`".

show-with view-modifier show-without view-modifier

Turns defaults on or off for various view modifiers. This is a distinct `.sheetsrc` statement, not either a `set` or a `bind`. For example:

```
show-with java-package-name
```

View modifiers simply change some property of a fragment display. The modifiers `java-package-name` and `java-class-name` affect how the names of Java

fragments are shown. The modifier `sheet-contents` affects whether sheets are displayed "open" or "closed".

goto-existing-window : boolean = true

Whether show-in-context commands can reuse existing windows, or should always create new ones.

context-sheet-shows-most-general-first : boolean = true

If true, then the context sheet will show the most general (superclass) methods at the top of the list. If false, the order is reversed.

global-undo-history-size : integer = 10

The number of commands that Global Undo will let you back up.

recenter-agressively : boolean = false

If true, Sheets will center the selected fragment/location more often. If false, Sheets will recenter mainly to keep the selection on the screen.

allow-graph-reordering : boolean = false

If this is set, we get prettier graphs because we can avoid node crossing. On the other hand, if it isn't set, graph traversals are more predictable.

condense-doc-toc : boolean = true

This controls whether ordinary doc paragraphs are shown normally in the TOC or are hidden.

standard-java-indent : integer = 2

The number of spaces that each nested level of a Java fragment should be indented. The Sheets project advocates a 2 space indentation, but some people find 4 spaces preferable.

2.4.1 Fonts

In another amazing syntactic innovation, the `.sheetsrc` supports a record-like syntax for specifying information about how a particular sort of text should be displayed. You can specify the point `size` (an integer), the `font-name` (a string) and whether a `bold` face should be used (boolean).

Each kind of text is associated with a sort of variable which can be set with the `set` statement, but only when a particular font attribute is designated. For example, you say:

```
set code-font.font-name = "courier"
```

all-fonts

A double-magic font pseudo-variable. If you set an attribute of this font, it sets the attribute of all of the other display fonts. For example, this will set all fonts to

twelve-point:

```
set all-fonts.size = 12
```

code-font : font = Courier

The font used for displaying Java code.

toc-font : font = Helvetica

The font used to display the table-of-contents

summary-font : font = Helvetica

The font used to display summaries in the context help window.

graph-font : font = Helvetica

The font used to display graph labels.

documentation-font : font = TimesRoman

The font used to display DocSheet documentation text.

2.4.2 Display Layout

show-verbose-dialogs : boolean = true

If true, "verbose" message dialogs will be popped up. If false, the same messages will be written to the console window from which Sheets was launched.

show-obvious-commands : boolean = true

Whether pull down menus should show "obvious" commands, like text cut + paste. The theory is that the user will always call those commands using the keyboard shortcut, so there's no point in even displaying the obvious commands. Try to be pretty conservative in your notion of what is "obvious".

show-pull-down-menus : boolean = true

Whether or not to display pull down menus. (If the pull down menus are hidden, the user will have to rely on popup menus.)

show-affected-by-sheet : boolean = false

Whether the affected-by sheet is shown by default.

show-context-sheet : boolean = true

Whether the context window is shown by default.

context-help-delay : integer = 100

The number of milliseconds that `Sheets` will wait between executing a cursor motion command and updating the context sheet. You should consider increasing it if you find that the display isn't keeping up with your keystrokes.

show-edit-sheet : boolean = true

If true, then display the window showing all edits currently in progress.

sheet-height : integer = (chosen based on screen size)

The default height (in pixels) of a window that contains a sheet.

short-sheet-height : integer = (chosen based on screen size)

The default height (in pixels) for the context window and the scratch sheet window.

windows-task-bar-height : integer = 28

The height of the MS-Windows task bar, in pixels. Default is 28, which works on at least one machine. This is used to determine where to pop up the windows which are created at the bottom of the screen. If you are using some other OS or have "auto-hide" set, 0 is an excellent alternative.

wide-component-spacing : boolean = false

If this is true, we "double-space" between components -- otherwise it's closer to 1.5 spaces.

toc-width : integer = 20

The default width in characters of the TOC pane in an edit window.

editor-width : integer = 80

The default width in characters of the edit panel in an edit window.

3 The DocSheet Documentation Facility

3.1 What is documentation?

In Sheets, a documentation fragment is a piece of natural language text. Like everything else in Sheets, documentation is broken into fragments. Documentation fragments include paragraphs, sections, bulleted lists, and pre-formatted fragments.

Sheets documentation is based on XML. It looks a lot like HTML: you have angle brackets, and you have to use `<` and `>` instead of the < and > characters. Normally, when you view documentation fragments, the tags are invisible. However, when you begin editing a fragment, the tags become visible again.

Some differences from HTML: XML is case sensitive; HTML is not. XML supports minimization, which appears as a normal tag with a `/` character before the closing > -- this is a tag that closes itself. Examples:

```
blah blah <normal-tag>contents</normal-tag> blah blah
blah blah <minimized-tag/> blah blah
```

(Minimization is not used much in DocSheet)

3.2 Basic Editing

You can create a new documentation fragment by using the `Insert new` menu item, which is available on the pop up menu. You can also simply find an existing documentation fragment, and type new text at the bottom, much like you would with a Java fragment. When you commit the fragment, new documentation fragments will be created.

You can edit the text of the fragment in the usual way, just like you do code. Additional operations are available: you can turn a paragraph into a section using the transforms menu of the pop up menu. There are also DWIMish `promote-doc-fragment` and `demote-doc-fragment` commands, which attempt to mimic the behavior of the tab key in Microsoft Word's outline view . The promote command takes a documentation fragment and makes it "more important." For instance, a paragraph will turn into a section, and a section will be moved up in the hierarchy to make it a more important section. The promote command is by default bound to control-shift-TAB.

The demote command does roughly the opposite of promote, making a fragment less important. Sections turn into paragraphs, and paragraphs turn into items in lists. Demote is bound to control-TAB.

Note that promote are demote are not completely reversible -- demoting several times and then re-promoting the same number of times will give you something equally "important" as what you started with, but it may not be identical -- you may be left with a paragraph rather than, say, a preformatted.

Also note that promote and demote will refuse to work if the fragment is contained in more than one container (has more than one parent) -- this is a problem MS-Word never had to worry about!

3.3 More Documentation Fragments: lists, definitions, and Preformatted

A <list> is simply a list of fragments, and can be ordered or unordered. These are similar to HTML ordered and unordered lists: ordered lists should be displayed with numbers, and unordered lists are displayed with bullets. (This is how they appear when run through the XML->HTML renderer. Inside Sheets, however, both kinds of lists look identical, save for the title of the DocListFragment.)

A <definition> is where you give the definition of a word. There are two fields. In the Sheets representation, definitions look like sub-sheets; the title of the definition is the word being defined, while the body (contained fragments) is the descriptive text. Typically, the descriptive text will be a paragraph fragment, although you can put any kind of fragment

there that you want. The title is not limited to a single term--if you want to have multiple terms, put each of them on their own line.

A <preformatted> fragment is like the HTML <pre> tag. In a preformatted fragment, there is no word wrap, and a fixed width font is used. Preformatteds do not yet use the rich text engine, so all entities and tags are quite visible and ugly within Sheets.

3.4 Styles

A style is font information that does not cross fragment boundaries. Styles are abstract concepts which apply to a range of concepts, but have no single visual representation. For instance, there is an *em* style, which may be rendered in either italics or bold.

Styles may be nested, but they may not otherwse overlap, nor may they cross fragment boundaries. To add styles to existing text, you simply type in XML tags for the style.

The built-in styles are: **strong**, *em*, and `code` (a fixed width font).

3.5 Links and anchors

To create a hyperlink, copy the target of the link into the cut buffer. (Note that the target of the link does not have to be a documentation fragment) Then, select the start point of the link and execute the `create-link` command, which can be found on the pop up menu. This will create a link to the component in the cut buffer. You may then typing in a description of the link, or if you leave that space blank, Sheets will use a summary view of that component. (The link text will only appear when you export the file, unfortunately) This kind of linking is known as *native links*. Example:

blah blah <link slot="0">This is a link</link> blah blah

To traverse a hyperlink, simply click on the link with the middle mouse button. If your mouse does not have a third mouse button, you can simulate it by holding down the ALT key while pressing the left mouse button.

To remove a link, simply delete the start and end tags. To change a link, edit the `links` attribute.

Sheets also allows you to create documents that interact with HTML documents, using URL links and anchors. To link directly to a URL, rather than to a fragment, write it as `<link URL = "whatever">`. To create a URL anchor for a component, typing into the anchor attribute of the fragment. If you wish for the fragment to contain more than one anchor, put each anchor name on a new line. (Note that the anchor is for the entire fragment; there is no way to create an anchor for a specific point within a fragment)

Another form of linking is the `documentation` attribute, which all fragments have. Simply pick a fragment you wish to document, usually but not necessarily a Java code fragment, and display its documentation attribute by selecting the `Show Attribute` command from the pop up menu. Next, copy the target fragment (usually a documentation fragment) into the documentation lists attribute.

There is a corresponding attribute, `documentation-for`, which is the reverse of the `documentation` attribute. Sheets will automatically maintain the bi-directional of the of these attribute, and you can get it either one.

3.6 Converting Documentation into HTML (rendering)

Sheets has three representations for documentation: an internal database representation, external representation based on XML (also known as DocSheet), and an HTML representation. The internal representation is what is used inside the database. To convert from the Internal representation to the ex mill representation, you must export the fragments. (This situation is analogous to the internal database representation vs. file representation of Java programs)

For best viewing, you need to convert the XML documentation into HTML format, a process known as rendering. There is no way to directly convert from the internal representation to HTML. There is more than one way to render a given XML document; for instance, you might render the emphasis style as either italics or bold. Sheets comes with one renderer, but you can use a different one if you want. To run the default renderer, type:

```
docsheet2html < input.html > output.xml
```

The default renderer will automatically create a table of contents for your HTML document. It will also create URL anchors for <edit-command>s and <edit-variable>s; the anchor will be the same as the name of the command/variable.

If you wish to write your own render, consider using the XSL style sheet facility available with most XML implementations. (The default renderer does not use XSL, because at the time of implementation XSL was unstable. However, it has probably gotten better in the intervening months...) You'll need to consult Appendix B for the structure of XML documentation.

3.7 Converting HTML into sheets documentation

Sheets comes with an `html2docsheet` utility (found in the `util` directory) to assist conversion from HTML into sheets documentation. The tool is not perfect, however. In fact, it can't be perfect--HTML is simply too unstructured to convert perfectly. Instead, the HTML to DocSheet tool uses heuristics to do the best it can.

One of the limitations of the conversion tool is that it only handles correct HTML. A surprising number of web pages are actually not legal HTML; the converter will not be able to handle them. If you encounter problems, try loading the HTML file into Netscape page composer, and then resaving the file. This will fix most errors; any remaining problems will have to be fixed by hand.

Command line usage:

```
html2docsheet < input.html > output.xml
```

Once you have a file in XML format, you can import it into Sheets much like you import Java files. Select `Import` from the `File` menu, and select the file you wish to import. You may need to do additional cleanup, as the HTML to XML converter is not perfect (see above). When you import XML files, any hyperlinks will use URL's rather than native links. (See section on linking for details) To convert these links into native links, select all fragments you wish to converge which contain links that you wish to convert, and run the `convert-urls-to-native-links` command. You'll be informed about any links that can be converted into native links, because the corresponding anchor could not be found.

To remove URL anchors from fragments, select the fragments which have encouraged that what you wish to remove, and run the `strip-anchors` command.

Tip: Use the `recursive-flatten` command to flatten all fragments in the a document, and then select all the fragments using a shift click with a mouse. That way, you can easily convert to native links or strip anchors in a single command.

3.8 Appendix A: Local extensions to DocSheet

An <edit-command> fragment is useful for describing the editor commands of Sheets. It is similar to a definition fragment, but allows you to optionally specify a default key binding for the command.

Similarly, <edit-variable> lets you describe variables that you can set in your `.sheetrc` profile. Variables have a name, and optionally a type and a default value.

3.9 Appendix B: informal DTD of DocSheet

3.9.1 Structure Tags

Structure tags have a well defined grammar; some structure tags only go inside specific structures. In Sheets, structured tags act as fragment boundaries.

```
// Toplevels will probably be component boundaries in Sheets
toplevel
    <para>
  | <section>
```

```
  | <block-quote>
  | <itemized-list>
  | <ordered-list>
  | <definition>
  | <literal-layout>
  | <graphic>
  | <edit-command>
  | <edit-variable>
<para>
  styled-text
<section>
  <title>
    styled text
  toplevel+
<unordered-list>
  <list-item>+
<ordered-list>
  <list-item>+
// DocSheet equivalent of HTML's <dt><dd> combo.
<definition>
  <term>+
    plain-text
  top-level+
// Because Microsoft's XML parser throws out whitespace
// (in violation of the XML spec), we make liberal use of entities.
// Use   for a space character, and <newline> for newlines.
// Tools should hide this kluge from users.
<preformatted>
  styled-text
// Description for one or more edit commands
<edit-command>
  <command>+
    <name>
    <default-key-binding>?
  top-level+
// Description for one or more edit variables
<edit-variable>
  <variable>+
    <name>
    <type>?
    <default-value>?
  top-level+
```

3.9.2 Links and Anchors

Native links are always turned into URL links upon export, thus there is no XML representation of native links. We use object ids as the URL.

URL anchors are represented as XML attributes on the tag; if there is more than one anchor for a fragment, we have multiple attributes that begin with "anchor" and end with a unique integer. Example:

<para anchor1="what" anchor2="ever">

3.10 Appendix C: Relation to DocBook

DocSheet (the XML representation of documentation) started off as a slimmed down version of DocBook, an industry standard SGML DTD. It diverged pretty quickly, however. I soon concluded that it was more important to be familiar to HTML users than to maintain similarity to DocBook. At this point, DocSheet has very little in common with DocBook.

3.11 Appendix D: Future work

Fix bugs w/ bidirectional attributes, and create the various promised attributes.

Implement <list-item> (currently, each item in the list is expected to be a single fragment).

Move ImageFragments into the Doc world.

Completely hide all aspects of XML from the user.

Have preformatteds do font-lock.

Allow links inside section titles.