Grammatik Rule Designer Utility

Parts of a Custom Style Guide

To effectively use the features Grammatik offers for style guide creation, you need to understand how Grammatik structures a writing style.

Rules make up the basic building blocks of a Grammatik style guide. Grammatik uses rules to check for errors in documents. Rules can deal with simple problems, like the presence of a specialized jargon word in a document, as well as complex problems, like the use of an adjective where the grammatical structure of the sentence requires an adverb.

Rule classes provide an easy way to organize rules into related groups. Without rule classes, keeping track of all the different rules would be impossible.

The rule classes take their names from the types of rules they contain. For example, any rules dealing with phrases that are wordy or rambling are located in the Wordy rule class. The rule class organization is especially helpful when you create your own rules. You can assign rules you create to a rule class containing similar rules. If you create a rule that does not fit into a regular rule class, you can always assign that rule to one of three custom rule classes.

Writing style thresholds allow you to increase the readability of your writing. You can control overly long, confusing sentences by setting a maximum sentence length, maximum number of consecutive nouns, and maximum number of consecutive prepositional phrases. To keep your writing more active, you can set the number of allowed passive sentences. You can even control split infinitives and the way numbers appear in your writing.

The writing style includes rules, rule classes, writing style thresholds, and a default formality level. When you create a custom style guide, you might need to create a special writing style that includes only the rules, rule classes, special thresholds, and formality level that you want for the type of writing you do.

Each rule and writing style in Grammatik is assigned a formality level. The formality level of a writing style, which you can change to fit the formality of each document you check, tells Grammatik which rules to use to check a document. Writing styles can have formality levels of informal, standard, or formal. The formality level of a rule, informal, standard, formal, or any combination of the three, tells you when Grammatik uses the rule to check your documents.

For example, you might create rules to locate and point out jargon words used in your company so that you can eliminate them from public press releases. But you would not need to do this if you were just writing an informal memo to another company employee.

Steps in Creating a Custom Style Guide

To create a custom style guide, you use both Grammatik and a special program called the Rule Designer. The exact steps you follow to create your custom style guide depend on whether you can use a predefined writing style, must create a new writing style, or must create, edit, or remove any rules. Generally, the custom style guide creation process involves:

- Planning your custom style guide
- Adding words to the Mor.Proof Dictionary
- Creating new writing styles
- Creating new rules
- Editing rules
- Removing rules

Planning your Custom Style Guide

You begin to identify your style guide needs by breaking down your writing tasks into related groups. If you work for a large business, you might write memos and reports designed for fellow employees, plus press releases and advertisements for the general public. All four types of writing require different levels of formality, vocabulary, sentence structure complexity, and so on. Determining the differences between your different writing tasks provides you with information on the types of things your custom style guide must address.

After you determine the style requirements of your different types of writing, you can recognize grammar or style problems that seem to crop up repeatedly in company writing. Style and usage errors that consistently appear could be flagged by either a predefined Grammatik rule or one that you create.

Now that you've analyzed your writing, you can begin to determine what your custom style guide needs to contain. Decide on the tone, use of passive voice, length of sentences, and use of jargon. Create sample sentences demonstrating the problems you commonly see in your writing. Consider what solutions you could create to address the problems. Finally, if the types of writing you do vary widely in terms of audience and purpose, you should consider whether you need one or more custom writing styles.

Testing Grammatik's Resources

After you have an idea of what you want your custom style guide to accomplish, begin to test Grammatik's predefined writing styles and rules by running interactive checks on test sentences with Grammatik. Once you have tested them, you can decide which writing style to use as a template, what rule classes you must turn on or off, and what rules you must create on your own.

Look at the ten predefined writing styles in Grammatik. Decide which writing styles you should test. To test the writing styles, follow the instructions below.

1. Create test sentences that contain the common grammar and style problems you find in your writing. Save the test sentences in a file in ASCII text format or a word processor format that Grammatik can check.

2. Using a different predefined writing style each time, run an interactive check on the file you created. Note the error messages you receive, the type of error, the rule, the rule class, and the suitability of the rule's advice. Also note the errors that Grammatik does not catch.

3. Determine which predefined writing style seems most effective in catching the errors and problems in the test sentences.

4. Determine which rule classes you need to deactivate to eliminate error flags that you consider invalid for your type of writing. You should be able to determine this from the information gathered during the checks.

5. Decide if you need to create any new rules. Include any rules concerning company jargon or grammar and style problems that Grammatik did not catch in your testing.

You now know what writing styles to use, what modifications need to be made to those writing styles, and what special rules you need to create.

If you need to create custom writing styles for your custom style guide, refer to the WordPerfect Reference manual's appendix on Grammatik for instructions. If you need to create custom rules for one of the predefined Grammatik styles, continue to the next section.

Using the Rule Designer

You can use the Rule Designer to create, edit, or remove rules in either predefined or custom writing styles.

NOTE If the rules you create contain words that do not appear in Grammatik's dictionary, you must add them to the dictionary before your rules will work properly. Refer to the WordPerfect Reference to learn how to add words to Grammatik's dictionary.

Installing the Rule Designer

To install the Rule Designer, insert the diskette into your floppy drive. Copy the Rule Designer files into the directory where Grammatik is (usually WPC60).

Making a Backup Copy of the Grammatik Rule Dictionary File

A file essential to the operation of both Grammatik and the Rule Designer is the file called the

Grammatik Rule Dictionary (GK51US.RUL). This file contains all rules and rule classes Grammatik uses to check documents.

When you add, edit, or delete rules, you make changes to the Grammatik Rule Dictionary. In particular, when you edit or remove a rule, you are removing the original rule from that file. You permanently lose any rules removed in this way. To protect against the permanent loss of original Grammatik rules, make a backup copy of the Grammatik Rule Dictionary before you use the Rule Designer for the first time. Give the backup copy a different name so that the Rule Designer does not find it.

You can put the backup copy of the Grammatik Rule Dictionary file either on your hard drive or, if you don't have enough space on your hard drive, on a floppy disk.

Starting the Rule Designer

Once you know that the Rule Designer is installed on your computer's hard disk drive and you make a backup copy of the Grammatik Rule Dictionary file, you can begin to use the Rule Designer to create, edit, and remove rules.

1. Make sure that the directory containing the Rule Designer is the default directory.
2. Type
   GKRDEN
3. Press Enter. The Rule Designer welcome screen appears.

NOTE Because of the number and complexity of rules in Grammatik, the Rule Designer might need a few minutes to read all of the rule information from the Grammatik Rule Dictionary file.

After the Rule Designer finishes loading the rules from the Grammatik Rule Dictionary file, the Rule Designer main window appears.

Selecting a Rule

Before you can edit a rule, remove a rule, or use a rule as a template for a new rule, you must find and select it from the rule list. Since the Grammatik Rule Dictionary file contains thousands of rules, the Rule Designer provides you with several search methods to make it easier to find the rule you want.

When you are looking for a rule, you can narrow the list by choosing one of the search methods. The Rule Designer creates a new rule list that matches your search information.

You begin selecting a rule at the Rule Designer main menu. Press Alt+s. The Select Rules menu appears, listing the ways you can narrow down the rule list.

The Select Rules menu lists eight different rule search methods. Select a search method by using the arrow keys to highlight a method name and pressing Enter. An explanation of each method

follows.

Reselect all rules - This method displays the complete list of rules. It is especially useful if you have narrowed the rule list too much and need to begin the search again.

By Rule Class - This method displays a list of the rules in the rule class of your choice. When you select this search method, the Rule Designer displays a dialog box containing a list of all rule class names. Move through the list one rule class at a time by pressing PgUp and PgDn. If the Rule Designer cannot not find any rules in that rule class, press Esc to return to the complete rule list.

NOTE Some rule classes appear to have no rules in them. This is because the rules in these classes are built into Grammatik's internal logic and cannot be displayed, edited, or removed.

By Pattern Contained in Rules - This method displays a list of all rules containing a certain "pattern," or group of characters, that you enter. When you select this search method, the Rule Designer displays a text entry box. Type in the pattern you want to search for and press Enter. The Rule Designer searches through all the rules and displays a list of rules that contain the pattern you typed. If the Rule Designer cannot find any rules that contain the pattern, press Esc to return to the complete rule list.

By Rule Formality - This method displays a list of all rules that have a certain formality level. When you select this search method, the Rule Designer displays a list of different formality levels. After you answer the question, the Rule Designer searches the rule list and displays the rules that have the formality level you specified. If the Rule Designer cannot find any rules that have that formality level, press Esc to return to the complete rule list.

By Ignored Phrases - This method displays a list of the rules in the Ignore This Phrase rule class. Unless you add rules to the Ignore This Phrase rule class, the class will be empty. If the Rule Designer cannot find any rules in this class, press Esc to return to the complete rule list.

By Modified Rules - This method displays a list of the rules you have edited in the current editing session. If you edit many rules during one session, this search method becomes quite useful as a review tool. If the Rule Designer cannot find any rules that you have edited during the current editing session, press Esc to return to the complete rule list.

By User Added Rules - This method lists all rules in Grammatik Rule Dictionary that you have added either through the Rule Designer or through merging ASCII text into Grammatik Rule Dictionary. If the Rule Designer cannot find any rules that you have added, press Esc to return to the complete rule list.

By Duplicates - This method displays a list of duplicate rules. Unlike the other search methods, searching for duplicate rules searches through all the rules regardless of how far you have narrowed down the rule list. If the Rule Designer cannot find any duplicate rules, press Esc to return to the complete rule list.

If you find the rule you want in the list, use the arrow keys to highlight it. The Rule Designer lists the rule's parts on the Rule Designer main window. You have successfully selected the rule. You can now edit the rule, remove the rule, or use it as a template for a new rule.

Using the "Find Rule" Command

If you know some character pattern in the rule you want to select or cannot find the rule by using the search methods on the Select menu, you can locate it with the Find Rule command. For example, you could use a rule's pattern and the Find Rule command to locate and select the rule.

NOTE This search method is different from the By pattern contained in rule method, which changes the entire rule list to display only those rules containing a specific pattern.

To use the Find Rule command:

1. When you select this search method, a text entry box displays.
2. Type in the search pattern you want to search for and press Enter.

The Rule Designer searches through the rule list. If it finds a rule that contains the search pattern you entered, the Rule Designer highlights the rule in the rule list and displays the rule's parts. You can edit the rule, remove the rule, or use the rule as a template for a new rule. If the rule the Rule Designer displays is not the one you want, press Enter to resume the search.

If the Rule Designer cannot find a rule that contains the character pattern for which you are looking or has reached the end of the rule list, it displays the End of Rule List dialog box.

Press Home to return to the last rule list you created. Press Esc to return to the complete rule list.

Editing a Rule

In some cases you might find that some rules Grammatik uses need to be changed slightly to meet the needs of your custom style guide. Except for some special rules Grammatik uses to operate correctly, you can edit rules to meet your needs.

CAUTION When you edit or remove a rule, you are deleting it from the file Grammatik Rule Dictionary. If you do not have a backup copy of the original Grammatik Rule Dictionary file, you will permanently lose any original rules that you edit or remove.

You can edit the rule pattern, rule advice, rule class, and rule formality. You begin the process at the Rule Designer main window.

NOTE Some rule classes appear to have no rules in them. This is because the rules in these classes are built into Grammatik's internal logic and cannot be displayed, edited, or removed.

1. Select the rule you want to edit.

2. Press Alt+e to display the Edit menu.

3. Use the arrow keys to highlight Edit selected rule and press Enter. The Rule Editing window appears with Pattern: highlighted.

4. Press Enter to display an edit box for the rule pattern. Edit the rule pattern and press F10 when you are finished. The Rule Designer highlights Advice:.

5. Press Enter to display an edit box for the rule advice. Edit the rule advice and press F10 when you are finished. The Rule Designer highlights Rule Class:.

6. Press Enter to display a list of all rule classes. The Rule Designer displays a list of the rule classes to which you can assign a new rule. You can move through the rule class list one rule class at a time by using the arrow keys. If you know the first letter in the name of the rule class you want, you can go directly to the rule class by typing that letter. Once you find the rule class you want, highlight it with the arrow keys and press Enter. The Rule Designer displays Formality:.

7. Press Enter to display the list of possible formality levels. Use the arrow keys to highlight the formality level you want and press Enter.

8. After you choose formality level, the Rule Designer displays the Rule Editing window and highlights Pattern: again. To edit a part of the edited rule, highlight the part and press Enter. Make the changes and return to the Rule Editing window. To save the edited rule, press F10. To erase the edited rule, press Esc and then Y.

Adding Replacements To a Rule

Some rules in Grammatik have replacements included in their advice. Replacements are words or phrases meant to present a better alternative to what the Grammatik rule flagged as an error. For example, the rule:

@#/ a great deal of \ Simplify.|much \some

has two possible replacements, some and much, for the phrase a great deal of. If Grammatik encountered the phrase a great deal of in a document, it would notify the user and present the advice and replacements. The user could either edit the document manually or choose one of the replacements.

You can use replacements effectively in your own rules. If you work for a company that has internal code names and official names for its products, you could create rules that ensure documents meant for public use would contain only the official product names. A rule would have a product's internal code names as the rule pattern, the statement Use only official product names in public documents as the advice, and a product's official product name as the replacement.

You can add as many replacement words or phrases as you want if the total size of the advice does not exceed 256 characters and the total size of the rule pattern and advice does not exceed 512 characters. To separate the replacements, place a backslash (\) between each replacement word or phrase.

Follow the instructions below to add a replacement to a rule's advice.

1. Select the rule to which you want to add a replacement.

2. Press Alt+e to display the Edit menu.

3. Use the arrow keys to highlight Edit selected rule and press Enter. The Rule Editing window appears with Pattern: highlighted.

4. Use the arrow keys to highlight Advice: and press Enter. The Rule designer displays an edit box for the rule advice.

5. Use the arrow keys to move to the very end of the advice text. Type| and the replacement. If there is more than one replacement, press Tab after each replacement until you have typed all replacements. When you are finished press F10. The Rule Designer highlights Rule Class:.

6. Look over the rule advice to make sure you've entered the replacements correctly. %GMK%If everything looks correct, press F10. You return to the Rule Designer main screen.

Creating a New Rule

1. Press Alt+e to display the Edit menu.

2. Use the arrow keys to highlight Add a new rule and press Enter. The Rule Editing window appears with Pattern: highlighted.

3. Press Enter to display an edit box for the rule pattern. type in the rule pattern and press F10 when you're finished. The Rule Designer highlights Advice:.

4. Press Enter to display an edit box for the advice. Type in the rule advice and press F10 when you're done. The Rule Designer highlights Rule Class:.

5. Press Enter to display a list of rule classes. Move through the list by using the arrow keys or PgUp and PgDn. If you know the first letter of the rule class, type the letter in to go directly to the rule class. When you've found the rule class you want, highlight it with the arrow keys and press Enter. The Rule Designer displays Formality:.

6. Press Enter to display the formality levels. Use the arrow keys to highlight a formality level and press Enter.

7. Press F10 to save the new rule. Press Esc, then y to erase the new rule.

Using the "Ignore This Phrase" Rule Class

The Ignore This Phrase rule class is unlike any other rule class. The rules that appear in this class do not appear when Grammatik checks documents. In contrast to all other rules, the rules in this class tell Grammatik what to ignore in documents. The reason for such a special rule class is to give you the power to make Grammatik ignore items it would usually flag as errors in your documents.

For example, if you used the name of a product like WhizBang Graphics Viewer in a document, Grammatik would normally flag the word WhizBang as a capitalization error. If you work for a company that has many products with names that contain capital letters in the middle of words, the error reports could become annoying. The Ignore This Phrase rule class allows you to eliminate such errors.

To make Grammatik ignore the product name WhizBang Graphics Viewer, you would use the Rule Designer to create a new rule that specifies WhizBang Graphics Viewer as the rule pattern, Ignore as the rule advice, the rule class Ignore This Phrase, and whatever formality rating you wanted. Now Grammatik will ignore WhizBang Graphics Viewer whenever it appears in a document. If needed, you could create more rules for other product names or any other phrase that Grammatik would consider an error.

NOTE If a word that you want to include in a rule does not appear in Grammatik's dictionary, you must add that word to the dictionary for your new rule to will work properly. Refer to the WordPerfect Reference manual's appendix on Grammatik to learn how to add words to Grammatik's dictionary.

Follow the instructions below to create a rule for the Ignore This Phrase rule class.

1. Press Alt+e to display the Edit menu.

2. Use the arrow keys to highlight Add a new rule and press Enter. The Rule Editing window appears with Pattern: highlighted.

3. Press Enter to display the rule pattern editing box. Type in the rule pattern and press F10 when you're done. The Rule Designer highlights Advice:.

4. Press the down arrow. The Rule Designer highlights Rule Class:.

5. Press Enter to display a list of all rule classes. Type I to find Ignore this Phrase in the list of rule classes. Highlight Ignore this Phrase with the arrow keys and press Enter. The Rule Designer highlights Formality:.

6. Press Enter to display the formality levels. Choose a formality level with the arrow keys

and press Enter.
7. Save the new rule by pressing F10. Erase the new rule by the pressing Esc, then y.

Deleting a Rule

Some rules that come with Grammatik might not apply to the type of writing you do. If you encounter rules that make more problems than they solve, you can delete the rules with the Rule Designer.

CAUTION When you delete a rule, you are removing it from the file Grammatik Rule Dictionary. If you do not have a backup copy of the original Grammatik Rule Dictionary file, you will permanently lose any original rules that you edit or delete.

Follow the instructions below to remove a rule.

1. Select the rule you want to remove.

2. Press Alt+e to display the Edit menu.

3. Use the arrow keys to highlight Remove rule and press Enter. The Rule Designer removes the rule from the screen and from the file GK51US.RUL.

If you decide that you need to bring back a rule you deleted during the current Rule Designer session, follow the instructions below.

1. Press Alt+e to display the Edit menu.

2. Highlight Restore removed Rules and press Enter.

3. The Rule Designer restores all rules you removed during the current Rule Designer session.

Quitting the Rule Designer

1. With the Rule Designer main screen displayed, press Alt+f. The File menu displays.

2. If you want the changes you made to be permanent, highlight Quit-save changes and press Enter. If you want to abandon any changes you made, highlight Cancel-no changes and press Enter.

Getting Help

Like Grammatik, the Rule Designer has an extensive on-line Help system. To get help from anywhere in the program, press F1. A Help menu will appear listing topics for which you can display help.

In Grammatik 5 the term "parsing rules" refers to rules that contain parsing tokens. Parsing tokens are @-tokens that specify parts of speech.

What Rule Patterns Contain

A rule pattern can contain up to 16 tokens. A token is any word or group of symbols that is set off by spaces. When you read text, you recognize the beginning and end of a word by the spaces that come before and after it. The same is true for a token in a rule pattern.

The three kinds of tokens used in Grammatik rule patterns are:

- Words
- Wild cards
- Tokens that begin with the symbol @

NOTE Either the first or second token in a rule pattern must begin with a letter from a-z.

Rules Using Words Only

The easiest kind of rule to write is one that contains only words.

Let's say you use the word assume in your writing too often. You can easily write a rule to flag this word in your documents. The rule pattern would look like this:

```
assume
```

This will match the word assume wherever it occurs. It will not match any other form of the verb, such as assumes or assuming. Each time assume is highlighted, you will see the advice you have written, perhaps "Try a less formal expression." If you included replacements in your rule, the rule pattern, advice, and replacements might look like this:

```
assume\Try a less formal expression.|say \ take for granted \
be sure \ am     sure \ are sure
```

You could then select one of the replacements to substitute directly into the document in place of assume. Note that each replacement is separated from the next by a tab.

Since a rule pattern can contain up to 16 tokens, you can write a rule pattern that contains more than one word. For example, the rule pattern:

```
we would like to call it to your attention that
```

will flag this exact phrase whenever it occurs in your writing.

NOTE Words in rule patterns are written in lowercase letters. Grammatik will match any character in a rule pattern to both its capital and lowercase forms in text. To match a word only when it is capitalized or only when it is not requires special tokens, which are explained in the next chapter under Matching Capitals.

To Match an Abbreviation

If you want to flag an abbreviation, you can include the period with the word just as it normally would look. For example, this rule pattern will flag the common abbreviation for California:

calif.

Wild Cards

What if you want to catch not only assume but also assumes, assumed, assuming, and assumption? You could write five rules, one for each word. But there is a limit to the number of rules you can add to the rule list. It would be better to catch all these words with one rule pattern. To do this, you need to use a wild card. Let's look at two of the wild cards used in designing rules.

The Asterisk (*) Wild Card

The asterisk (*) wild card can substitute for a whole word or for a series of characters in a word. For example, the rule pattern add* will match not only add but also adds, added, and adding. It will also match additional, additionally, adder, addressee, addlepated, and any word that begins with add.

Let's return to the rule pattern assume, writing it instead as:

assum*

This rule pattern will flag any word which begins this way, including:

assume, assumes, assumed, assuming, assumption, and assumable

Each time one of these words is highlighted, you will see the advice you have written. Since this rule will flag several different words, including several verb forms, a noun, and an adjective, it would not be practical to try to include replacements in the rule.

The rule pattern assum* uses the * wild card to match different endings. You can also put the * wild card at the beginning or in the middle of a group of letters. Here are some examples of its use in each position:

The rule pattern	matches
tak*	take, takes, taken, taking, takeoff, takeout
saleswom*n	saleswoman, saleswomen
*ed	walked, fed, feed, steed, armored, bed

NOTE You may use only one wild card in a word.

As you can tell from the variety of words matched in the *ed example above, it is not a good idea to use the * wild card with a series of fewer than three characters. This is especially true if the

rule pattern is a common prefix or suffix. For example, the rule pattern un* would match some 20,000 English words!

There is another reason to be as specific as you can when using the * wild card. A rule pattern with a wild card that is too general slows Grammatik down significantly since it must stop to flag each word or phrase that the rule pattern matches.

The * wild card can also stand alone as a complete token. In this case, it will match any single word in the phrase. For example, the rule pattern the * man will match the friendly man, the wounded man, and the first man, but not the man or the strong young man.

NOTE Remember that either the first or second token of a rule pattern must begin with a letter from a-z. This means that if you use * as a token alone or at the beginning of a word in one of the first two tokens, you must make sure that the other token begins with a letter. The Rule Designer will check the syntax of each rule you write and warn you if a rule fails to meet this requirement.

The Underscore (_) Wild Card

Grammatik's other wild card is the underscore (_). The _ wild card is used only at the end of a word. It is more specific than the * wild card, since it matches only the base form of a word and its plural form (ending in -s or -es). For example,

The pattern	matches both
book_	book and books
brush_	brush and brushes
glove_	glove and gloves

NOTE The _ wild card cannot match irregular plurals, like women or alumnae.

If you want to match only the base and the plural form of a noun, it is better to use the _ wild card than the * wild card. Using the _ wild card will exclude words with suffixes other than the plural forms. For example, friend_ will match only friend and friends. The token friend* will match not only friend and friends but also friendly, friendliness, friendship, and any other word beginning with friend.

NOTE You may use only one wild card of any kind in a word.

Matching a Comma

If you want to match a comma (,) in a rule pattern, make it a token by itself. Let's say you want to flag the expression if so but only when it comes before a comma. Your rule would look like

this:

if so,

The comma must be a separate token from the word before or after it in the rule pattern because Grammatik sees it as a separate token in text.

You cannot use the comma character alone as the first token of a rule pattern. In that position, it must be shown as @",". For example, to flag the phrase you know, but only after a comma, write:

@"," you know

You can use this special type of token to indicate a comma anywhere in the rule pattern, but it is really only necessary when the comma is the first token.

Using Simple @-Tokens

Any token that is not just a word, a wild card, or a combination of the two must begin with the @ symbol. An @-token may indicate:

- Position in a sentence (position token)
- How many words to skip in the middle of a phrase (span token)
- Literal strings to match or avoid matching (literal token)
- Where to start and stop highlighting an error (highlight token)
- One or more parts of speech (parsing token)

Position Tokens

Grammatik uses two position tokens, @#/ and @#. , to make rule patterns match only at the beginning or end of a sentence.

If you want to flag a word or phrase only if it occurs at the beginning of a sentence, begin the rule pattern with @#/ . For example, the rule pattern:

@#/ and

tells Grammatik to flag and every time it finds this word at the beginning of a sentence.

If you want to flag a word or phrase only if it occurs at the end of a sentence, put @#. at the end of the rule pattern. The rule pattern:

with @#.

tells Grammatik to report an error every time with appears at the end of a sentence.

Span Tokens

Let's say you want to match the phrase on (someone's) behalf. You really want to flag more than

one phrase, including:

on my behalf
on his behalf
on John's behalf
on John Smith's behalf

If you use a span token, you can flag all of these with one rule pattern. A span token is an @ followed by a number between 1 and 9. The span token tells Grammatik to match on two parts of a rule pattern with from 0 to x words between them, where x is the number after @. In other words, the number after @ determines the "span," or maximum number of words, that Grammatik will skip between the two parts of the match. For example, the span token @4 will skip from 0 to 4 words. To flag the examples above, we can write the rule pattern:

on @2 behalf

Note that this rule pattern will also match on behalf, as in on behalf of our company. This is because a span token also matches the two parts of the phrase with nothing between them. If you don't want to match on behalf, write:

on * @1 behalf

This rule pattern makes sure that at least one word, represented by the * wild card, comes between on and behalf.

The table below contains some examples of span rules.

The span rule	matches	but does not match
a @2 number of	a number of, a large number of a very large number of,	a most impressively large number
tak* @1 chanc_	take chances, taking a chance, chances taken unreasonable chances	take no unnecessary chances

You now know enough about creating rule patterns to create simple rules. To design more complex rule patterns and replacements, you should have some experience with computer programming or symbolic logic and understand grammar well.

Literal Tokens

A literal token, or literal, contains one or more words or strings of characters. Grammatik will match the rule pattern if it finds any one of these words or strings in the document. A literal has

the form @"...", inside of which a vertical line (|) separates the strings. The vertical line (|) acts as a logical OR symbol, telling Grammatik to make a match if it finds the first string, or the second, or the third, etc.

NOTE You type the vertical line symbol (|) by typing Shift+\.

Here is an example of a simple literal:

@"more|most" importantly ,

This rule pattern flags more importantly or most importantly before a comma.

A literal can contain more than two words or character strings. One example of a long literal is:

@"mother|father|sister|brother|son|daughter" in law

This rule pattern flags any of the terms inside the literal before the words in law.

A literal cannot include any spaces. Remember that a space in a Grammatik rule pattern signals the beginning of a new token. Because of this, if you want to flag both due to illness and because of illness, you cannot write the single rule pattern @"because of|due to" illness. For this kind of case, use two separate rule patterns, each matching just one phrase.

A literal alone cannot be a whole rule pattern. This is because it begins with @. The first (or second) token of a rule pattern must begin with a character from a-z.

NOTE Certain symbols have one meaning inside a literal token, but a different meaning in another type of @-token. The vertical line (|) is one of these. You will encounter its other meaning in the next chapter.

Wild Cards in Literal Tokens

Literals can contain wild cards. We have learned the use of two wild cards so far: * and _. Here are some examples of these wild cards in literals:

the @"implement*|execut*" of will match

the implementation of
the implementing of
the execution of
the executing of

actual @"fact*|experience_" will match

actual fact
actual facts
actual factor
actual factors
actual experience

actual experiences

Grammatik recognizes another wild card, the caret (^). The ^ symbol acts as a wild card only inside a literal. The ^ wild card takes the place of any single character.

NOTE If you know DOS syntax, it may help to think of the ^ wild card as the equivalent of the ? wild card in DOS.

Here is an example of a rule pattern using the ^ wild card:

```
@"^:^^|^^:^^" o'clock
```

The purpose of this rule pattern is to flag an error like 9:15 o'clock. Since time numerals are only written as one or two digits, followed by a colon plus two more digits, the literal shows only these two forms, separated by a logical OR symbol.

Here is another example:

```
april @"," @"1^^^|2^^^"
```

This rule pattern flags April followed by a comma and a year (April, 1983). The @"1^^^|2^^^" token represents any year beginning with a 1 or a 2.

As these rule patterns show, the ^ wild card is especially useful for matching digits.

The NOT Symbol

In some situations you might find it helpful to flag a word or phrase only when it is not together with a certain other word. For example, people sometimes confuse the words allusion and illusion, as in a sentence like He lost all his allusions about her. The rule pattern:

```
allusion_
```

will flag this but will also flag a correct use of allusion like He made an allusion to her new book. To avoid this, you can limit the match to allusion but NOT before the word to, by using the NOT symbol. The symbol for NOT is the exclamation point (!). The rule pattern looks like this:

```
allusion_ @!"to"
```

This rule pattern means: Match allusion or allusions only if the next word is not to.

This rule pattern illustrates that applying the NOT symbol to even one word requires the literal-token form.

Here is an example of a longer literal containing !:

```
continu* on * @!"road|path|route|street"
```

This rule pattern will flag phrases like continue on talking and continued on along, but it will not flag continued on the route.

The symbol ! can be used in any @-token. This means it can also be used in position tokens:

If a rule pattern has the token it will not match

@!#/ at the beginning of a sentence

@!#. at the end of a sentence

The ! symbol is very useful. But the example rule patterns above don't work as well as they should. This is because Grammatik normally highlights every word in the document that is referred to by a token in the rule pattern. Grammatik will therefore highlight even the word referred to by the token containing the NOT symbol.

To solve this problem, you need to use a highlight token.

Highlight Tokens

Accurate highlighting of a writing problem is important. It focuses attention on the problem and allows the replacements to work well. Grammatik normally highlights every word that a rule pattern matches.

You can control what Grammatik highlights in the rules you create by using the highlight tokens @{ and @}. Placing a @} highlight token in a rule pattern will highlight from the beginning of the rule pattern to the @}. Placing a @{ highlight token in a rule pattern will highlight from the @{ to the end of the pattern. Placing both the @{ token and the @} token in a rule pattern highlights whatever text falls between the two tokens.

The highlighting works this way:

The rule pattern	causes Grammatik to highlight
very @} capable employee	very
very @{ capable employee	capable employee
very @{ capable @} employee	capable
very capable @{ employee	employee

As another example, look again at allusion_ @!"to". In the sentence He lost all his allusions about her, it would be correct to offer illusions as a replacement for allusions. But Grammatik replaces the whole highlighted phrase. If the rule pattern highlights allusions about in this sentence, then the replacement illusions won't work. This is resolved by adding a highlight token to the rule pattern:

allusion_ @} @!"to"

Now Grammatik will match allusion if the next word is not to, but it will only highlight allusion. The replacement illusion will work perfectly.

When a highlight token is used before another @-token, the two tokens should be combined by

putting the two tokens together and deleting the second @. This adds to Grammatik's speed. For example, a better pattern than the one above is:

allusion_ @}!"to"

One special case where you must use a combined highlight token, instead of a highlight token standing alone, is right after a span token. Let's say you want to highlight only would have in a sentence like:

If your son would have come, he could have had cake.

The rule pattern:

If @3 @{ would have

will not work. The tokens @{ and would must be combined. The new rule pattern:

If @3 @{"would" have

matches and highlights correctly. Note that to combine a word with a highlight, you must put it in literal token form.

To summarize, use highlight tokens when you want to highlight only part of the text that the rule pattern matches. The most common reasons to do this are to make replacements work, or to focus attention on only part of the matched phrase. You will find further examples of rule patterns with highlight tokens in later sections.

Matching a Numeral

When you want to match any word whose first character is a numeral, you can use a special literal called the numeral token. The token is @"~9". For example, if you wanted to match the word June when it appears before a numeral, such as June 3 or June 3rd, you could write the rule pattern:

june @"~9"

NOTE Type the tilde symbol (~) by typing Shift + `

Normal wild cards can find the same things. However, since they cannot differentiate between letters and numerals, wild cards would report more false errors, making them less useful when you are looking for numerals only.

Matching Capitals

Grammatik normally matches a word without regard to case. For example, the rule pattern new will match new, New, NEW, neW, and so on. To match specific capitalization in a word, you need to use a special form of literal called a capital token. There are four capital tokens:

The token	matches
@"~Cap"	only when the first letter of the word is a capital. (Example: matches New only)

@"~UPPER"	only when the entire word is in capitals. (Example: matches NEW only)
@"~lower"	only when the entire word is in lowercase letters. (Example: matches new only)
@"~MiXed"	any form of the word not matched by one of the three tokens above. (Example: matches NEw, nEw, nEW)

The following rule pattern:

wrote to @"~Cap"

will match wrote to only before a name (for example, wrote to John or wrote to Xerox).

You can include more than one type of capital match in a single token, using the | to separate types. For example, if you want to include all-uppercase names in the above rule pattern, write the rule pattern as:

wrote to @"~Cap|~UPPER"

This way you will also match wrote to NASA or wrote to FDR. Notice that you need to keep the tilde (~) in each capital match, even after a |.

You will usually want to use capital tokens to match a specific word with certain capitalization. The capital token itself cannot contain the word you want to match. But there is a way to let the capital token refer to a word before it in the rule pattern. To do this, use a backshift.

Backshifts

The symbol < in a Grammatik rule pattern is called a backshift. A < always comes just after an @; it may or may not be followed by a digit (2 or greater). Both @< and @<3 are two examples of backshifts. A backshift is not a complete token by itself, but makes the token that it begins refer to another token somewhere before it in the rule pattern.

Backshifting to One Token Before

When used alone (not followed by a digit), @< makes the token that it begins refer to the token immediately before it in the rule pattern. The backshifted capital token in the rule pattern:

new @<"~Cap"

makes it match only the word New with an initial capital.

When counting back from a @<, count only tokens that actually stand for a word in the text. Do not count highlight tokens (@{ or @}) standing alone, or prior tokens containing @<.
For example, the pattern:

jew* @<"~lower" @<!"jewel*"

flags jew, jewish, jewry, etc. when the word is in all lowercase letters so that Grammatik can recommend capitalizing it. But jewel and jewelry begin with the same letters yet do not require a capital. The @<!"jewel*" token prevents these words from being flagged. Both it and the capital token refer to the same prior token.

Backshifting More Than One Token

Say you want to match close proximity, but not in close proximity. You could write:

@!"in" close proximity

This will match the way you want it to. However, it will also highlight whatever word comes before close proximity because the token @!"in" refers to that word. You could try adding a highlight token to limit the highlight to close proximity:

@!"in" @{ close proximity

This may look like a good solution. But it is not a legal rule pattern. Remember that one of the first two tokens in a rule pattern must begin with a character from a-z. The first two tokens above begin with @. So what can you do?

Using a backshift to change the order in the rule pattern will solve the problem. Start with the phrase you want to match, close proximity. Then add a highlight token to limit the highlight to that phrase. Lastly, tell the rule pattern not to match if the word before the highlighted phrase is in. The rule pattern looks like this:

close proximity @}<3!"in"

The last token, @<3!"in", counts three words back to check the word before close. If that word is not in, the rule pattern will match. However, since the highlight comes before the backshift, the word before close will not be highlighted.

NOTE Do not use a token containing a backshift immediately after a span token.

Let's say you want to flag any form of the verb plan before the word ahead. You would like to catch:

You should plan ahead carefully.

This requires planning ahead.

and other similar sentences. To match these, you could write the rule pattern:

plan* ahead

But plan can also be a noun. You do not want to flag the nouns plan or plans, as in:

He submitted the plans ahead of schedule.

They considered his plan ahead of its time.

You can restrict the match to a verb form of plan by using a backshift and a part of speech code for verb. Here is the rule pattern:

plan* @<|V ahead

The vertical line symbol (|) in a parsing token means intersect. The part-of-speech code for verb is V. This rule pattern means: Match any word starting with plan, but only if it is used as a verb, and only before ahead.

NOTE You type the vertical line symbol (|) by typing s+\.

At the end of this chapter is a complete list of the part of speech codes you can use in Grammatik 5 rule patterns.

The INTERSECT Symbol

When you use the INTERSECT (|) symbol in a parsing token, Grammatik checks to see if any part of speech after the | symbol matches any part of speech of the word in the token. If there is even one part of speech that is the same in both word and parsing token, the match will occur (as long as the rest of the rule pattern matches, too). The rule pattern:

calm @<|AV

tells Grammatik to match the word calm only if it finds calm acting as an adjective or a verb in the sentence. It will match calm in:

My boss is a calm person.

because calm acts as an adjective. It will also match:

She tried to calm the anxious child.

because calm acts as a verb. But it will not match:

We waited for the calm after the storm.

because calm acts as a noun in this sentence.

The intersect symbol is the only operator you should use in Grammatik 5 parsing tokens. Because Grammatik 5 parses so well, it normally assigns only one major part of speech to each word in a sentence. When your parsing token matches a word, you can almost always be certain you have matched what you intended.

What Does "Part of Speech of Word" Mean?

The rule pattern does not just match any possible parts of speech that the word might have in any context. It matches only the parts of speech that Grammatik has assigned the word in the exact sentence context where it is found.

When you write rule patterns containing parsing tokens, you need to have a good understanding of grammar. You also may want to check what part(s) of speech Grammatik will assign to the word you want to flag.

If you want to be sure what part(s) of speech the word in your rule pattern will be assigned in a particular sentence, do this:

1. Put a wordy phrase like all things considered into a test sentence containing what you want to match. This is to make sure that Grammatik will find an error to flag. (You can see the parts of speech in a sentence only when Grammatik flags an error.)

2. Run the sentence through Grammatik in interactive mode.

3. When the error flags, press F4 to see the parts-of-speech analysis.

A list of the abbreviations used in the parts-of-speech analysis can be found in the Help.

Noun and Verb Forms

Nouns and verbs, like other parts of speech, can be identified by single-letter codes alone: N will match any noun, and V will match any verb. But both nouns and verbs can take several different forms. For example, a noun can be singular or plural. A base verb like eat changes to ate when used in the past form and to eaten when used as a past participle.

You can restrict a match by adding lowercase codes for the different forms that nouns and verbs take. For example, Vt limits the match to the past form of the verb only; Vg limits the match to the present participle.

You can combine these lowercase verb form codes. For example, the rule pattern:

play @<|V

will match all verb forms: play, plays, played, and playing. But the rule pattern:

play @<|Vtg

will match only the verbs played and playing. The lowercase codes for the forms of nouns and verbs are all listed at the end of this chapter.

Using the NOT Symbol in a Parsing Token

You can use the NOT symbol (!) in a parsing token to prevent a match on a particular part of speech. For example, say you want to flag the wordy phrase in rare cases in a sentence like:

This is true only in rare cases.

You plan to offer the replacement rarely. You could use the phrase in rare cases by itself as the rule pattern.

But your rule pattern would also flag:

In rare cases of plagiarism, students have been expelled.

The replacement rarely would not work well for this case. To avoid flagging such sentences, you could write the rule pattern:

in rare cases @}!|P

This lets the phrase match only when the word after it is not a preposition.

You can also use the ! symbol in a parsing token that refers back to a previous token. For example, let's say you want to flag the term hereafter when it means from now on. But you don't want to flag it when it is a noun meaning heaven or life after death. You can write the rule pattern:

hereafter @<!|N

to flag the word only when it is not used as a noun.

Parts-of-Speech Subtypes

You have seen above that you can identify specific forms of nouns and verbs by adding certain lowercase letters to the uppercase code. For example, N will match any noun, whereas Nu will match only an uncountable noun. As examples of this, N would match both boy and water, whereas Nu would match only water.

Grammatik 5 recognizes many other distinctions relating to parts of speech besides the major forms of nouns and verbs. Grammatik's dictionary classifies words according to 96 different types and attributes. The Rule Designer lets you limit a match not only to a major part of speech, but also to its specific attributes, or subtypes.

For example, V will match any verb. But what if you want to match only a verb like be, seem, feel, appear, or look? These verbs are part of the group called linking verbs. The subtype code l stands for "linking verb." The token @|VWl will match only a verb of this kind.

Here is a case where you might want to match only linking verbs. Let's say your employees sometimes type please when they mean pleased. You want to flag an error in sentences like:

I am please to meet you.
His boss seemed please with the results.
But you don't want to flag:

The items he selects please our customers.
Well-cooked food should please the eye.
In both the correct and incorrect sentences, please comes after a verb. You could write

    @|V please

but this would flag correct sentences as well as incorrect ones.

You notice that the verbs before please in the error sentences are linking verbs, while the ones in the correct sentences are not. You can write the rule pattern to match only after a linking verb:

    @|VWl please

The token @|VWl means: Match if the word is a verb and belongs to the subtype linking verb.

This pattern will match any form of a linking verb before please.

After V (verb), you must use W to signal Grammatik that one or more subtypes follow. The l is the subtype code for "linking verb." After any part of speech other than verb, you must use Q before you add one or more subtype codes. For example, to match a pronoun only if it is plural, write the token:

    @|OQp

O is the part of speech code for pronoun, Q signals that a subtype follows, and p is the subtype code for plural pronoun. This pattern will match we, us, you, they, them, ours, ourselves, and so on, but will not match I, me, mine, she, him, herself, hers, and so on.

NOTE You cannot write a token to include a subtype without any part of speech code. For example, @|p and @|Qp are not legal tokens.

You can group several subtypes after a Q or a W, just as you can group several parts of speech after the | symbol.

NOTE For the match to succeed, the word must not only intersect with at least one of the parts of speech, but also intersect with at least one of the subtypes.

For example, the token:

@|EOQerm

means: Match a word that is either a determiner or a pronoun, but only if it is a plural determiner, a reflexive pronoun, or a possessive pronoun.

This token would match plural determiners like these and the, reflexive pronouns like myself and ourselves, and possessive pronouns like my or mine. It would not match singular determiners like a or singular pronouns like me.

A complete list of subtype codes that you can use in rule pattern parsing tokens appears at the end of this chapter.

The GLOBAL-NOT Symbol

We have discussed the use of the NOT symbol (!) in certain parsing tokens. However, parsing tokens containing subtypes present a complication. The NOT symbol applies separately to the part of speech codes and to the subtypes. The table below shows different tokens and the items they match.

The token	matches
@|D	any adverb (including quickly and never)
@|DQn	any adverb that has the subtype "negative" (matches never but does not match quickly)
@|D!Qn	any adverb that does not have the subtype "negative"(matches quickly but does not match never)
@!|DQn	any word that is not an adverb and that has the subtype "negative" (matches nothing but does not match never)
@!|D!Qn	any word that is not an adverb and does not have the subtype "negative"

This would seem to cover all bases. But what can you do if you want to match any word at all, except a negative adverb like never or nowhere? If you use the token:

@|D!Qn

you will match only adverbs that are not negative. You will not match any other part of speech. If you use the token:

@!|DQn

you will match any word that is not negative or an adverb.

The solution to this is to use the GLOBAL-NOT symbol, the tilde (~). The GLOBAL-NOT symbol can be only used before the INTERSECT symbol (|).
The token:

@~|DQn

will match any word except negative adverbs. Similarly, the token:

@~|VWl

will match any word, including any verb, that is not a linking verb.

In summary, if you want to indicate NOT before a parsing token that contains a subtype, you probably want the GLOBAL-NOT symbol. Here are two other useful GLOBAL-NOT tokens:

The token	matches any word except
@~|VWb	any form of the verb be
@~|EOQm	a possessive adjective like my or their

You now have all the information you need to write even the most complex rule patterns. The next two sections of this chapter list all the part-of-speech codes, including noun and verb forms, and the subtype codes.

Parts-of-Speech Codes

General Forms

Code	Meaning
A	adjective
C	conjunction
D	adverb
E	determiner
I	infinitive "to"
J	interjection
M	modal
N	noun
O	pronoun
P	preposition
S	comparative/superlative
U	number
V	verb
X	auxiliary (eg.: do, have, be, will)

Noun Forms (For use after the part of speech code N)

Code	Meaning
c	countable
o	possessive
p	plural
s	singular
u	uncountable

Verb Forms (For use after the part of speech code V)

Code	Meaning
b	base verb
g	present participle
r	past participle
s	third person present
t	past

Subtype Codes

Use the following subtype codes only after the V for verb. Put a W before these subtype codes.

Code	Meaning
b	form of "to be"
d	ditransitive
h	form of "have"
i	intransitive
l	linking (copulative)
o	form of "do"
s	next verb can be subjunctive
t	transitive

Use the following subtype codes after any part of speech except verbs. Put a Q before these subtype codes.

Code	Meaning
a	attributive adjective
b	subject pronoun
c	coordinating conjunction
d	singular determiner
e	plural determiner
f	non-count determiner

g	count determiner
i	intensifier
l	clause conjoiner (Exs. so, yet)
m	possessive pronoun or determiner (Ex. my)
n	negative word
o	object pronoun
p	plural pronoun
q	quantitative determiner (Exs. much, many)
r	reflexive pronoun
s	singular pronoun
t	semantic: relates to time
u	subordinating conjunction
v	relative pronoun
w	a "wh-" word (Exs. why, when)
x	predicative adjective
y	postpositive adjective
z	semantic: relates to measurement

General Rule Pattern Guidelines

The first token in a rule pattern can be:

- A string beginning with a character from a-z
- A string beginning with a * wild card
- A * wild card alone
- An @-token

If the first token does not begin with a character from a-z, then there must be a second token that does.

Size Limits for Rules and Parts of Speech

Grammatik 5 rules have maximum size limits that they cannot exceed. When you are creating Grammatik 5 rules, keep in mind that:

The	Cannot exceed
entire rule: (patterns + advice + replacements)	490 characters
rule pattern alone: (including spaces)	275 characters and/or 15 tokens

advice and replacement: 290 characters
(including spaces)

Offering Only the Right Form of Replacements

Let's say you write the rule pattern:

@|V representative of

to flag a sentence like:

This breakdown is representative of the kinds of problems we encounter while drilling.

You would like to offer a form of show, typify, or suggest as replacements. Since the rule pattern flags all forms of the verb before representative, you want to include as replacements all possible forms of the synonyms you propose.

You could write the replacement field like this:

|show \ shows \ showed \ showing \ typify \ typifies \ typified \ typifying \ suggest \ suggests \ suggested \ suggesting

But this replacement field is very long. A rule's advice, including replacements, can contain only 290 characters. If you had to write out all these replacement forms, you could include only one, or at most two, synonyms for the phrase you flagged.

You can solve this problem by using the STREAMLINE REPLACEMENTS symbol, represented in the rules by the plus symbol (+), before the base form of each replacement. The + symbol tells Grammatik to make the replacement take the exact grammatical form of the word it is replacing. When Grammatik sees the + symbol, it automatically makes the form of the replacement the same as the form of the first word that the rule pattern matches.

For example, instead of the long replacement field above, write this:

|+show \ +typify \ +suggest

Let's say your rule pattern has matched the phrase is representative of in the document. Only shows, typifies, and suggests will appear as replacements.

If the token whose form you want to match is not the first token in the rule pattern, you need to point to it explicitly with the backshift @<+. Say that the rule pattern is:

all boil* @<|V down to

and you want to suggest a form of reduce to as a replacement. You would like the replacement to match the form of boil*. Since boil* is not the first token in the rule pattern, you must use the @<+ token to make the STREAMLINE REPLACEMENT symbol apply to it. The revised rule pattern is:

all boil* @<|V @<+ down to

The replacement field is:

|+reduce to

NOTE Don't use the + symbol when offering forms of the verb "be" as replacements. For this verb, you need to write out each form in the replacement field (example: be, is, are, and so on).

You can make a rule pattern even more efficient by combining the + with another @-token, instead of using the separate token @<+. The rule pattern above is best written:

all boil* @<+|V down to

The token @<+|V tells Grammatik to:

· match only if the previous word is a verb

· make the replacements match the form of the previous word

You can combine the + symbol with any @-token except a span token.

Duplicating a Matched Word as a Replacement

Sometimes it is best to offer just one of the words matched as the replacement itself. This often applies to the noun in a wordy expression. For example, the redundant phrase:

actual experience_

can be shortened simply to experience or experiences.
To make the replacement match the form of experience found in the document, write the rule pattern as:

actual experience_ @<+

You could write the replacement field as:

|+experience

Grammatik would then analyze the form of experience it finds in the document to see if it is singular or plural, and offer a singular or plural replacement accordingly.

But it would be simpler to tell Grammatik just to offer the exact word that it has actually found in the document as the replacement. You can do this by using the IDENTICAL REPLACEMENT symbol, + : =. The replacement field will be just:

|+:=

This means: Offer the exact word matched by the rule pattern as the replacement.

If there is no + symbol in the rule pattern, the +:= symbol will duplicate the first word matched. If there is a + symbol in the rule pattern, the +:= symbol will duplicate the word indicated by the + symbol, as we saw above with the rule pattern:

actual experience_ @<+

For another example of the use of the +:= symbol, let's look at the replacements for the rule pattern:

@|VWl of @2 importance

This rule pattern flags phrases like:

appeared of importance
is of little importance

seems of very great importance

The replacement field for this rule is:

|+:= important \ +:= unimportant

Here is how the replacements would appear for the three phrases above, respectively:

appeared important \ appeared unimportant

is important \ is unimportant

seems important \ seems unimportant

Now that you've learned how to use all the different tools Grammatik and the Rule Designer provide, you can create the rules you need for your own custom style guide.

The Rule Designer provides an excellent step-by-step procedure for creating, editing, and removing rules. Most people who use Grammatik will find this the most convenient way to create rules for their custom style guide. There is, however, another way to create rules. You can create an ASCII text file and merge it with the existing rules in the file Grammatik Rule Dictionary. For users who have many rules to create and feel very comfortable with Grammatik's rule structure, this alternate way to create rules might save time.

To create rules in ASCII text format, you need a word processor that can create ASCII text files or a simple text editor, and the Rule Designer program. You also need to understand how Grammatik rules look when they are in ASCII text format. Many word processor packages have an option that allows you to create ASCII text files. Check the user's manual that came with your word processor to find out whether it can create ASCII text files.

NOTE The file containing your rules must be in ASCII text format. Non-ASCII word processor files contain special information that will make the file unreadable to the Rule Designer. Make sure any word processor you want to use can create ASCII text files before you begin.

Creating a Rule in ASCII Text Format

This section assumes you have already read the previous chapters of this manual and have created at least one rule in the Rule Designer. If you have not read the previous chapters of this manual or created a rule, you should do so before attempting to create a rule in ASCII text format.

You begin creating the rule in either a word processor or a text editor. When you look at a rule in the Rule Designer, the parts of the rule are presented individually. The pattern, advice, rule class, and formality of the selected rule appear in the Rule Editing window as distinct parts. In a rule in ASCII text format, the same parts appear in one long line, each part separated from the others by special characters understood by Grammatik.

A rule in ASCII text format must have this syntax:

rule pattern\rule class code-=+formality\advice|replacements

To create a rule in ASCII text format, open a new file in your word processor or text editor and follow the instructions below.

1. Type in the rule pattern. When you finish, type in a backslash (\).
2. Type in the rule class ASCII code. Each rule class has a corresponding code. Tables listing all of the rule class names and their corresponding codes follow this section.
3. Type -=+. These symbols must be present for the Rule Designer and Grammatik to successfully read rules in ASCII text format.
4. Type in the formality ASCII code for the formality levels you want the rule to have. Tables below list the formality level codes and their meanings. Depending upon the formality rating you want to give a rule, type in one or any combination of the three formality level characters. The i represents the informal formality level. The s represents the standard formality level. The f represents the formal formality level. Most of the rules in Grammatik have the informal-standard-formal formality level, represented in ASCII text as isf. After you type the formality codes, type a backslash (\).
5. Type in the rule advice. If there are any replacements, type in a vertical line (|) and type in the replacements with a Tab between each.

NOTE Remember that replacements must be separated by tabs for the Rule Designer and Grammatik to recognize them successfully.

6. Press Enter to complete the rule. Look over the rule and check for errors and proper structure. If the rule contains an error, edit it. If the rule appears correct, you can begin typing in the next rule or save your work in an ASCII text file.

The Rule Class Codes

The table below lists the rule class code for each rule class in the Grammatik Rule Dictionary. If you create or edit rules in ASCII text form, one of the codes below must appear in the rule.

The table below lists each rule class code and its corresponding rule class.

Rule Class Code	Rule Class
0	Ignore This Phrase
1	General Style
2	Advertising Style
3	Business Style
4	Documentation Style
5	Fiction Style
6	Journalism Style
7	Memo Style
8	Proposal Style
9	Report Style
:	Technical Style
>	User Style Class 1
?	User Style Class 2
@	User Style Class 3

A	Archaic
B	Unbalanced ( ), { }, [ ], or "
C	Capitalization
D	Doubled Word or Punctuation
E	End of Sentence Punctuation
F	Punctuation
G	Ellipsis
H	Cliche
I	Colloquial
J	Jargon
K	Foreign
L	Wordy
M	Commonly Confused
N	Relative Pronoun
O	Overstated
P	Passive Voice
Q	Quotation Marks
R	Redundant
S	Split Words
T	Trademark
U	Questionable Usage
V	Vague Adverb
W	Adverb
X	Split Infinitive
Y	Infinitive
Z	Pronoun Number Agreement
[	Paragraph Problem
\	Incomplete Sentence
]	Question Mark
^ (caret)	Subject-Verb Agreement
_ (underscore)	Similar Words
' (accent mark)	Possessive Form
a	Article
b	Homonym
c	Comparative/Superlative
d	Pronoun Case
e	Preposition
f	Long Sentence
g	Gender-Specific
h	Number Style
i	Double Negative
j	Incorrect Verb Form
k	Spelling
l	False Plural
m	False Friend

n	Abbreviation
o	Second-Person Address
p	Pejorative
q	Sequence of Tenses
r	Run-on Sentence
s	Subordination
t	Tense Shift
u	Comma Splice or Fused Sentence
v	Sentence Variety
w	Adjective
x	Conjunction
y	Noun Phrase
z	Mid-Sentence Adverb
{	Formalisms
|	Object of Verb
}	End-of-Sentence Preposition

Formality Codes

The table below lists the possible formality codes you might see or use when viewing or creating rules in ASCII text form. A rule in ASCII text form must have one of the codes to work properly in Grammatik. Most Grammatik rules use the isf code, so that the rule will flag in every writing style. Rules that are coded with another combination will flag only in the corresponding writing styles. For example, a rule coded sf will only be flagged in writing styles that have either Standard or Formal formality levels.

Formality Code	Meaning
i	Informal
s	Standard
f	Formal
is	Informal, Standard
if	Informal, Formal
sf	Standard, Formal
isf	Informal, Standard, Formal

Merging Rules in ASCII Text Form

If you create new rules in ASCII text format, you add them to the file Grammatik Rule Dictionary by using the Merge ASCII Rules command. You find the command on the Rule Designer's File menu. This makes the rules you created in the ASCII text file available in the Rule Designer and Grammatik.

NOTE The files you try to merge into Grammatik Rule Dictionary must be in ASCII text format. Non-ASCII word processor files cannot be read by the Rule Designer. Make sure that any file

you try to merge into Grammatik Rule Dictionary is an ASCII text file before you merge it. If any of the rules you created in ASCII text format contain words that do not appear in Grammatik's dictionary, you must add them to the dictionary before your rules will work properly.

Follow the steps below to merge an ASCII text file into Grammatik Rule Dictionary.

1. Start the Rule Designer.

2. Press Alt+f.

3. Highlight Merge ASCII rules and press Enter. The Rule Designer displays a box asking you to type in the file name of the ASCII text file you want to merge into GK51US.RUL.

4. Type the name of the ASCII file you want to merge and press Enter. The Rule Designer find the file and merges the rule text into GK51US.RUL.

If the Rule Designer encounters a rule whose structure is incorrect, it alerts you to the rule in question and asks you to press Esc to continue with the rest of the ASCII text file.

When the Rule Designer finishes merging all of the rules in the ASCII text file, it displays the Rule Designer main screen and awaits your next action. Unless the Rule Designer notified you of an error, the file Grammatik Rule Dictionary contains all the rules you created in the ASCII text file.