

Version 3.1

NetCloak

Reference Guide

*“Dynamic, Interactive Web Sites...
...No Programming Required”*

rev. C

Maxum Development Corporation
Post Office Box 315
Crystal Lake, IL 60039-0315

Disclaimer of Warranty and Limited Warranty on Media

Maxum will replace defective distribution media or manuals at no charge, provided you return the item to be replaced with proof of purchase to Maxum within 60 days of product purchase. The limited warranty shall continue for the replacement software the remainder of the 60 days or for 30 days from the date you receive the replacement, whichever is longer. Maxum does not warrant that the software is free of bugs, errors, and omissions.

Maxum makes no warranties, expressed or implied, including without limitation any implied warranties on the media and manuals, including warranties of merchantability and fitness for a particular purpose. Maxum makes no warranty or representation, either express or implied, with respect to NetCloak, NetCloak Pro, its quality, performance, merchantability, or fitness for a particular purpose. As a result, NetCloak Pro is licensed "as is" and you the purchaser are assuming the entire risk as to its quality and performance.

In no event will Maxum be liable for direct, indirect, special, incidental, or consequential damages resulting from any defect in NetCloak Pro or its documentation, even if advised of the possibility of such damages. In particular, Maxum shall have no liability for any programs or data stored using NetCloak Pro on data storage devices of any description, including the cost of recovering such programs or data.

THE WARRANTIES AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS ORAL OR WRITTEN, EXPRESSED OR IMPLIED. No Maxum development distributor, dealer, agent, or employee is authorized to make any modifications, extensions or additions to this warranty. Some states do not allow exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. In any case whatsoever, regardless of the form of action, Maxum liability to you for actual damages will be limited to the purchase price you paid for the software.

Copyright Notice

The NetCloak documentation, program design, and design concepts are copyrighted, with all rights being subject to the limitations and restrictions imposed by the copyright laws of the United States of America. Under the copyright laws, this manual may not be copied, in whole or in part, including translation to another language or format, without the express written consent of Maxum Development Corporation.

ABOUT THIS GUIDE.....	1
If You Need Help.....	1
SYNTAX REFERENCE.....	3
General SHOW/HIDE Form.....	3
Specifying Multiple Parameters.....	3
Comparison Operators	3
General INSERT Form.....	10
Specifying Optional Parameters.....	10
Parameter Wildcards.....	11
Whitespace Characters.....	12
Quotation Marks.....	13
DYNAMIC PAGE COMMANDS.....	15
Unconditional.....	16
Time and Date.....	18
Counters.....	26
Security.....	30
Client Type.....	39
Random.....	41
File Information.....	44
Server Information.....	47
NetCloak Configuration.....	59
Variables.....	64
Cookies.....	74
Utility.....	78
HTML FORM TAGS.....	89
The FORM Tag.....	89
Form Fields.....	91
FORMS-PROCESSING COMMANDS.....	95
Primary Directives	95
Supplemental Directives.....	105
Insertion Commands.....	114
Using Multiple Instances of a Field	126
ADJUSTING INTERNAL SETTINGS	127
INDEX.....	129

This manual is the NetCloak Reference Guide, and contains a comprehensive listing of commands, describing their use and syntax. It is meant as a convenient, random-access reference for users familiar with NetCloak. The companion User's Guide contains an introduction to NetCloak, instructions for installation and configuration, and tutorials on using NetCloak. It is intended to guide new users through the installation process and acquaint them with using NetCloak's dynamic page and forms-processing commands. The User's Guide tutorials are intended to teach methods and concepts, and so do not cover every single command.

Each guide covers both dynamic page and forms processing commands, even though the forms processing commands are only supported in NetCloak Pro. This is mentioned in appropriate sections, but the user is assumed to be familiar with the distinction.

This manual does not explain how to install or configure a web server or how to create HTML pages. For more information on setting up a web server or writing HTML, see the server software documentation or an HTML reference.

If You Need Help...

There are three ways for you to get help if you have a problem.

1. The place to start is the Maxum Web site:

<http://www.maxum.com/>

Here you will find:

- Software updates.
- Complete on-line documentation.
- Frequently Asked Questions (FAQs).
- Other timely information.

2. If the answer to your question isn't at our Web site, please send e-mail to:

support@maxum.com

We'll get your question answered quickly by e-mail.

ABOUT THIS GUIDE

3. Finally, you might want to subscribe to the NetCloak mailing list. For information, go to:

<http://www.maxum.com/maillists.html>

The quality of the comments and help on this list are exceptional.

All NetCloak dynamic page commands follow the same rules of syntax. Like HTML tags, every command must begin with an open angle-bracket (<), followed immediately by the command name. The command name is not case-sensitive, but it is conventional to use uppercase to make the NetCloak commands more recognizable. The examples here and in the User's Guide show commands in all upper case.

Following the command name may be one or more parameters and a closing angle-bracket (>). While the parameters may be optional, the closing bracket is required.

General SHOW/HIDE Form

SHOW and HIDE commands have the command name followed by a comparison and comparison constants. Together they look like this:

```
<SHOW_DAY BEGINS SAT>
```

The command above, when described in English, means “show if the current day begins with SAT”. BEGINS is the comparison operator, and SAT is a parameter.

Specifying Multiple Parameters

Most SHOW and HIDE commands will accept multiple parameters in a series, and show or hide if any of them are matched. For example, the SHOW_DAY command accepts a day name as a parameter and so allows multiple days to be specified, as in:

```
<SHOW_DAY BEGINS "SAT" "SUN">
```

This will show text on either Saturday or Sunday.

Comparison Operators

All NetCloak SHOW and HIDE commands let you specify one of thirteen comparison operators, which let you specify exactly how to compare a NetCloak variable to the parameters listed in the command.

SYNTAX REFERENCE

A comparison operator causes NetCloak to perform a logical test, which has either a true or false result. If the result is true, then a SHOW command causes text following it to be shown; and a HIDE command causes the text to be hidden. If the result of the test is false, then the command has no effect on the page.

The comparison operator may be omitted from the command. If it is, a default comparison operator is used which is specific to each command. The default comparison operator for each command is listed in the command's description later in this User's Guide. In all cases, the default comparison operator permits backward-compatibility with versions of NetCloak prior to 3.0.

Comparison operators fall into one of two categories: **text** comparisons and **numeric** comparisons. Be sure you know which kind of operator you are using, or you will get unexpected results in your web pages.

Text Comparisons

Text comparison operators treat the variable and the command parameters as sequences of alphabetic characters. All text comparisons are case-insensitive, which means that the capitalization of the characters is ignored. Therefore, bob and BoB are considered an exact match. The text operators, and the tests they perform, are listed below. For the examples shown below, assume the existence of a NetCloak global variable named Platform with the value Macintosh.

CONTAINS

The comparison is true if the NetCloak variable contains any of the given text parameters. For example, the following commands all will cause text to be hidden:

```
<HIDE_GLOBAL Platform CONTAINS "mac">  
<HIDE_GLOBAL Platform CONTAINS "into">  
<HIDE_GLOBAL Platform CONTAINS "MacInTosh">
```

IS (Text Equality)

The comparison is true if the NetCloak variable exactly matches any one of the given text parameters. Of the

following three commands, only the last one will cause text to be hidden:

```
<HIDE_GLOBAL Platform IS "mac">  
<HIDE_GLOBAL Platform IS "into">  
<HIDE_GLOBAL Platform IS "mAcInTosh">
```

IN

The comparison is true if any one of the given text parameters contains the value of the NetCloak variable. The following command would cause text to be hidden:

```
<HIDE_GLOBAL Platform IN "Welcome to  
Macintosh.">
```

BEGINS (Begins With)

The comparison is true if the NetCloak variable begins with the exact same sequence of characters as any one of the text parameters, including whitespace characters (spaces, tabs, and returns). Of the following three commands, only the first one will cause text to be hidden:

```
<HIDE_GLOBAL Platform BEGINS "mac">  
<HIDE_GLOBAL Platform BEGINS "into">  
<HIDE_GLOBAL Platform BEGINS "Macintosh  ">
```

The third example fails the test because the text parameter contains extra space characters.

ENDS (Ends With)

The comparison is true if the NetCloak variable ends with the exact same sequence of characters as any one of the text parameters, including whitespace characters (spaces, tabs, and returns). Of the following three commands, only the first one will cause text to be hidden:

```
<HIDE_GLOBAL Platform ENDS "tosh">  
<HIDE_GLOBAL Platform ENDS "into">  
<HIDE_GLOBAL Platform ENDS "  Macintosh">
```

The third example fails the test because the text parameter contains extra space characters.

SYNTAX REFERENCE

BEFORE (Alphabetically)

This operator uses the ASCII value for each character in the text to determine if the NetCloak variable would be sorted alphabetically before the text parameters. If multiple parameters are listed, this comparison returns true if the variable is alphabetically before **any one** of the parameters. For this reason, this operator should ordinarily be used with only a single text parameter.

Using ASCII values, whitespace characters sort before digit characters, and digit characters sort before all alphabetic characters. Also, if all characters in a short word match a longer word, the shorter word sorts before the longer word.

Note: the BEFORE operator is **not** a date/time comparison operator. You usually do not want to use this operator in date/time commands.

Of the following examples, only the second one will cause text to be hidden using our "Platform" global variable:

```
<HIDE_GLOBAL Platform BEFORE "mac">  
<HIDE_GLOBAL Platform BEFORE "windows">  
<HIDE_GLOBAL Platform BEFORE "123xyz">
```

AFTER (Alphabetically)

This operator uses the ASCII value for each character in the text to determine if the NetCloak variable would be sorted alphabetically after the text parameters. If multiple parameters are listed, this comparison returns true if the variable is alphabetically after **any one** of the parameters. For this reason, this operator should ordinarily be used with only a single text parameter.

Using ASCII values, whitespace characters sort before digit characters, and digit characters sort before all alphabetic characters. Also, if all characters in a short word match a longer word, the shorter word sorts before the longer word.

Note: the AFTER operator is **not** a date/time comparison operator. You usually do not want to use this operator in date/time commands.

Of the following examples, only the third one will cause text to be hidden using our “Platform” global variable:

```
<HIDE_GLOBAL Platform AFTER "Macintosh">
<HIDE_GLOBAL Platform AFTER "windows">
<HIDE_GLOBAL Platform AFTER "123xyz">
```

EXISTS

This comparison doesn't really compare the NetCloak variable to anything; it always evaluates true if the variable has been defined to have any value. In fact, if this comparison operator is used, the text parameters following it are completely ignored. This comparison operator comes in handy when you want to make sure you do not overwrite a variable's value once it has already been set, such as is often the case when setting cookie values (see the SET_COOKIE command) or global variable values. For example, if the global variable named “Platform” does not already exist, the following code sets its value; otherwise, it leaves it alone:

```
<HIDE_GLOBAL Platform EXISTS>
<SET_GLOBAL Platform = "Macintosh">
<SHOW>
```

Numeric Comparisons

Numeric comparison operators treat the variable and the command parameters as numbers, converting text parameters to numbers as necessary. Numeric values may only contain these characters:

- the digits 0 through 9,
- a plus sign (“+”) or a minus sign (“-”), or
- a decimal point (period) character

A variable that contains other characters will be treated as text by NetCloak, and will be converted to the number zero for a numeric comparison.

SYNTAX REFERENCE

In date and time commands, the date and time parameters are converted to a single number in seconds. This allows comparisons and other operations in date commands to work just like those for plain numbers.

The numeric operators, and the tests they perform, are listed below. For the examples below, assume the existence of a counter named Orders with a value of 250.

== (Numeric Equality)

The comparison is true if the NetCloak variable has a value that is exactly equal to any of the listed numeric values. For example, the following command will cause text to be hidden:

```
<HIDE_COUNT Orders == 50 150 250 350>
```

LT (Less Than)

The comparison is true if the value of the NetCloak variable is less than any of the listed numeric values. In the following examples, the third one will cause text to be hidden:

```
<HIDE_COUNT Orders LT 50>  
<HIDE_COUNT Orders LT 250>  
<HIDE_COUNT Orders LT 500>
```

LT= (Less Than or Equal)

The comparison is true if the value of the NetCloak variable is less than or equal to any of the listed numeric values. In the following examples, the second and the third one will cause text to be hidden:

```
<HIDE_COUNT Orders LT= 50>  
<HIDE_COUNT Orders LT= 250>  
<HIDE_COUNT Orders LT= 500>
```

GT (Greater Than)

The comparison is true if the value of the NetCloak variable is greater than any of the listed numeric values. In the following examples, the first one will cause text to be hidden:

```
<HIDE_COUNT Orders GT 50>
<HIDE_COUNT Orders GT 250>
<HIDE_COUNT Orders GT 500>
```

GT= (Greater Than or Equal)

The comparison is true if the value of the NetCloak variable is greater than or equal to any of the listed numeric values. In the following examples, the first one and the second one will cause text to be hidden:

```
<HIDE_COUNT Orders GT= 50>
<HIDE_COUNT Orders GT= 250>
<HIDE_COUNT Orders GT= 500>
```

Negating a Comparison Operator

Comparison operators in NetCloak SHOW and HIDE commands can be negated by inserting a # character in the command immediately before or after the comparison operator. This character reverses the result of the comparison, so that the HIDE or SHOW occurs if the test is NOT true. Some examples will make this clear.

```
<HIDE_CLIENT #CONTAINS "AOL">
Welcome America Online users!
<SHOW>
```

This command shows a special message only to America Online users by hiding the text if the client name does not contain the text AOL. Read the command like this: "Hide if the client does NOT contain AOL".

You can put the # character either at the end of the show or hide command, or either before or after the comparison operator. All of the commands below are equivalent:

```
<HIDE_CLIENT# CONTAINS "Mozilla">
<HIDE_CLIENT #CONTAINS "Mozilla">
<HIDE_CLIENT CONTAINS# "Mozilla">
```

Here is one more example:

```
<HIDE_VARIABLE OSPreference IS# "MacOS">
You have selected MacOS as your preferred
operating system. Congratulations!
<SHOW>
```

SYNTAX REFERENCE

This code hides the congratulatory message from users who have chosen some other OS. (The command reads “Hide if the form variable OSPreference is not MacOS”).

The negation operator can also be used with the default comparison by adding it directly to a HIDE or SHOW command. For example:

```
<HIDE><SHOW_DAY# SAT SUN>.
```

General INSERT Form

INSERT commands have the command name followed by one or more parameters. For example:

```
<INSERT_COOKIE moof>
```

This will insert the value of the cookie named moof. Parameters such as a name are common, but the exact parameters are appropriate for each command.

In contrast to show and hide commands, insert commands which accept more than one parameter only allow one of each. The parameters are recognized by the order in which they appear, so ordering must always be closely followed.

Specifying Optional Parameters

Some parameters may be optional. You may leave them out if you don't need what they do. For example, the INSERT_DATE command allows an optional offset and an optional format parameter. The command may be used without any parameters, or with only an offset parameter: If multiple parameters are used, they must be included in order:

```
<INSERT_DATE +24 SHORT>  
<INSERT_DATE SHORT +24>
```

Of the two commands above, only the first will work correctly because the parameters are in the correct order.

Now the INSERT_DATE command will allow either the format or the offset by itself. In some commands, though, to

use the second or later parameters all of the preceding parameters must be specified.

An example is the `INSERT_COUNTDOWN` command, which accepts the date you would like to count down to as the first parameter, and then the label for the weeks, days, hours, and minutes values. If you intended to use `INSERT_COUNTDOWN` with a different label for the days value, you might try using:

```
<INSERT_COUNTDOWN_D 1/1/00 "x24 hours">
```

This would not work as intended. NetCloak will interpret x24 hours as the label for weeks, instead of days because the parameters are only interpreted in order. To use the second or later parameter in a command without specifying the preceding parameters, use a pair of quotation marks as a placeholder. The correct command for the previous `INSERT_COUNTDOWN` example would be:

```
<INSERT_COUNTDOWN_D 1/1/00 "" "x24 hours">
```

The Commands section specifies the order of the parameters, whether the order is required and which parameters are optional for each command.

Parameter Wildcards

Many NetCloak commands allow you to specify an asterisk in place of a single character in a parameter, where the asterisk will match any character. This is commonly known as a wildcard.

For example, the following command will display the line following it to any 3.x version of a browser:

```
<HIDE><SHOW_CLIENT CONTAINS "3.*">  
Please upgrade to version 4 of your browser.
```

`INSERT` commands do not allow wildcards. In most cases, a wildcard would not make sense for these commands. Wildcards are similarly disallowed in `SHOW` and `HIDE` commands which take numbers as a parameter, including `COUNT`, `RANDOM` and `SAMERAND` commands, or

SYNTAX REFERENCE

commands which take a date or a time as a parameter. The MACRO and EXEC_CGI commands do not allow wildcards in any of their parameters.

Variable commands (SHOW and HIDE LOCAL, GLOBAL or VARIABLE) will accept wildcards in parameters, however, you must use them carefully because an asterisk is used as the multiplication operator in SET_VARIABLE commands. Enclosing parameters in quotation marks make the commands easier to read.

```
<SET_VARIABLE myVar "bob">  
<HIDE><SHOW_VARIABLE myVar IS "b*b">  
This will be shown.
```

This example will show the text following if the value of the variable is bob, bib or any other three letter combination starting and ending with b. Wildcards do not work in variable values, however.

```
<SET_VARIABLE myVar "b*b">  
<HIDE><SHOW_VARIABLE myVar IS "bob">  
This will not be shown!
```

The example above will not show the text even though the wildcard in b*b should match bob.

Whitespace Characters

Parameters within NetCloak commands are separated by one or more spaces, tab characters or line breaks (collectively known as “whitespace” characters). For example, the following command contains four parameters after the command name -- Country, CONTAINS, USA, and Canada:

```
<HIDE_GLOBAL Country CONTAINS USA Canada>
```

If you omit a space between parameters, the meaning of the command changes, and you will get unexpected results. For example, the following command only contains three parameters (Country, CONTAINS, and USACanada), and thus behaves differently:

```
<HIDE_GLOBAL Country CONTAINS USACanada>
```

You may use any combination of whitespace characters between command parameters to improve readability of the HTML source code. For example, the following two commands are treated exactly the same by NetCloak:

```
<SET_VARIABLE Total = Item1Total + Item2Total +  
Item3Total + Item4Total + Item5Total + Shipping +  
Tax>
```

```
<SET_VARIABLE  
  Total =  
    Item1Total +  
    Item2Total +  
    Item3Total +  
    Item4Total +  
    Item5Total +  
    Shipping +  
    Tax  
>
```

Quotation Marks

In most cases, parameters can be placed in quotes , but quotes are not required. For example:

```
<SHOW_CLIENT CONTAINS "mozilla">
```

is functionally equivalent to:

```
<SHOW_CLIENT CONTAINS mozilla>
```

Using quotes, however, allows you to include spaces within a parameter. For example:

```
<SHOW_CLIENT CONTAINS "mozilla 3.0">
```

is very different from:

```
<SHOW_CLIENT CONTAINS mozilla 3.0>
```

The first case looks for a client that contains mozilla 3.0, but the second will recognize either client names that contain mozilla or client names that contain 3.0. This is because the space between mozilla and 3.0 is interpreted by NetCloak as a delimiter between parameters instead of the space that appears in the browser name.

SYNTAX REFERENCE

When a parameter is set inside quotes, you must use two double-quote characters to specify a quote, so that NetCloak does not interpret the quote character as the end of the parameter. For example:

```
<HIDE_VARIABLE SomeVar "He said ""moof""!">
```

This command would look for the string He said "moof"!, because NetCloak interprets the two double-quote marks as a quote instead of the end of the parameter.

Quotes may not be used around variable names. If a variable name is enclosed in quotes it will be interpreted as a text string and not a variable name. For example, if you have a local variable called TestVar that is set to the value Moof, you might do something like this:

```
<SET_LOCAL NewVar = "TestVar" & " Says the Dogcow">  
<INSERT_LOCAL NewVar>
```

This would display:

TestVar Says the Dogcow

What is really desired is this:

```
<SET_LOCAL NewVar = TestVar & " Says the Dogcow">  
<INSERT_LOCAL NewVar>
```

Resulting in:

Moof Says the Dogcow

In the first example TestVar is specified as a string and NetCloak inserts it accordingly. In the second, NetCloak recognizes that TestVar may be some other kind of entity, and evaluates it correctly.

DYNAMIC PAGE COMMANDS

Dynamic Page commands are used right in HTML pages that will be processed by NetCloak. There are three types of dynamic page commands. The first type inserts dynamic information from the server or the request, like the date or the value of a cookie, into the page in place of the command. The second type of commands let you hide or show portions of your document based on similar information. The third allows you to set the value of variables or NetCloak configuration settings. There are also a few that don't fit into these three categories.

All of the NetCloak commands are listed below, grouped according to the information they work with or what their function relates to on the server or in the page.

Unconditional

HIDE

SHOW

FLIPMODE

Time and Date

INSERT_TIME

SHOW_TIME

HIDE_TIME

INSERT_DATE

SHOW_DATE

HIDE_DATE

INSERT_DAY

SHOW_DAY

HIDE_DAY

INSERT_COUNTDOWN

Counters

INSERT_COUNT

SHOW_COUNT

HIDE_COUNT

INSERT_COUNTERDATE

INSERT_COUNTERTIME

Security

INSERT_DOMAIN

SHOW_DOMAIN

HIDE_DOMAIN

INSERT_USERNAME

SHOW_USERNAME

HIDE_USERNAME

INSERT_PASSWORD

SHOW_PASSWORD

HIDE_PASSWORD

REQUEST_PASSWORD

SET_USER

Client Type

INSERT_CLIENT

SHOW_CLIENT

HIDE_CLIENT

DYNAMIC PAGE COMMANDS

Random

INSERT_RANDOM	SHOW_RANDOM	HIDE_RANDOM
INSERT_SAMERAND	SHOW_SAMERAND	HIDE_SAMERAND

File Information

INSERT_FILESIZE	INSERT_MODIFIED
-----------------	-----------------

Server Information

INSERT_HOST	SHOW_HOST	HIDE_HOST
INSERT_HEADER	SHOW_HEADER	HIDE_HEADER
SET_HEADER	INSERT_REFERRER	SHOW_REFERRER
HIDE_REFERRER	INSERT_THISURL	SHOW_THISURL
HIDE_THISURL	INSERT_PATH	SHOW_PATH
HIDE_PATH	INSERT_SEARCH	SHOW_SEARCH
HIDE_SEARCH		

NetCloak Configuration

INSERT_NETCLOAK	SHOW_NETCLOAK	HIDE_NETCLOAK
SET_NETCLOAK	NETCLOAK	

Variables

INSERT_VARIABLE	SHOW_VARIABLE	HIDE_VARIABLE
INSERT_LOCAL	SHOW_LOCAL	HIDE_LOCAL
INSERT_GLOBAL	SHOW_GLOBAL	HIDE_GLOBAL
SET_VARIABLE	SET_LOCAL	SET_GLOBAL
VARIABLE_LIST		

Cookies

INSERT_COOKIE	SHOW_COOKIE	HIDE_COOKIE
SET_COOKIE		

Utility

INSERT_BEEP	REDIRECT	REPEAT
EXEC_CGI	EXEC_PIX0	EXEC_FDML
MACRO	MODEMACRO	

DYNAMIC PAGE COMMANDS: UNCONDITIONAL

Commands: Unconditional

<HIDE>

Specifying `<HIDE>` in your document will hide any text, links, etc. in your document until you specify otherwise. HIDE is unconditional; when you use HIDE, text following the command is hidden from all users, all of the time. This command is most often used to set up a conditional SHOW command, when you want to display a portion of a page only in certain circumstances.

<SHOW>

Specifying `<SHOW>` makes all following HTML text viewable by all users, all the time.

For example, this cloaked document:

```
Here is my HTML text.  
<HIDE>  
This will never be seen by anyone.  
<SHOW>  
This is the end.
```

Is the exact same as this HTML:

```
Here is my HTML text.  
This is the end.
```

The SHOW command is most often used to set up a conditional HIDE command, or to turn text display back on for all user after a HIDE command.

NetCloak's HIDE and SHOW commands make it possible to include notes or comments in your HTML documents that won't be revealed to users if they view or save the source of the document from their browser.

<FLIPMODE>

The FLIPMODE command, which accepts no parameters, simply changes the current mode from SHOW to HIDE, or vice-versa. It is handy when you want to display one of two

DYNAMIC PAGE COMMANDS: UNCONDITIONAL

possible sections of HTML with IF/THEN/ELSE logic. For example, if you have two HTML snippets, one for table-compliant browsers and one for others, you could:

```
<HIDE><SHOW_CLIENT !SupportsTables>
Table tags go here!
<FLIPMODE>
This is visible only to browsers that don't
support tables:
<SHOW>
```

In this case, FLIPMODE doesn't save much, but it can be very handy when you have created a complex condition and simply want to turn on the faucet for everyone else.

Commands: Time and Date

<INSERT_TIME offset format>

INSERT_TIME will place the current time, as text, into your document. The time will be formatted using the time format set in the Date & Time control panel. The time will be inserted in either short or long form according to the current time format setting (See "Configuring NetCloak" in the User's Guide).

The offset and format parameters are both optional, but if both are specified, the offset must come first.

The offset parameter is a positive or negative number that specifies the fractional number of hours difference between the time set on the web server and the time that should be displayed. In other words, an offset of +2.5 will display the current time plus two and a half hours. An offset of -4 would display the time less four hours. This can be used to show times that vary between time zones.

For Example:

```
The time in Chicago is <INSERT_TIME>.
The time in California is <INSERT_TIME -2>.
The Time in New York is <INSERT_TIME +1>.
```

A plus (+) or minus (-) sign is always required when specifying an offset.

DYNAMIC PAGE COMMANDS: TIME & DATE

The format parameter allows you to override the current time format setting. Valid values for this parameter are either SHORT (for the short time format) or LONG (for the long time format).

Here is an example:

The current time is <INSERT_TIME LONG>.

In this case, the time will be inserted into the page, and will include seconds.

<HIDE_TIME operator time1 time2 ...>

The HTML text that follows this command will be hidden during time intervals that meet the criteria of the specified comparison operator.

The comparison operator parameter is optional. If omitted, the default is to perform a begins with text comparison.

Hours are specified as on a 24-hour clock. For example:

<HIDE_TIME 21>

This will hide text following it for the one hour interval between 9:00:00 PM and 9:59:59 PM. Leading zeros for single digits are required, and multiple times may be specified.

If a text comparison operator is specified, the current time will be compared to the time parameters as a text value. For example, to hide text during the second half hour of every hour, you could write (assuming the short time format):

<HIDE_TIME CONTAINS ":3*" ":4*" ":5*>

If a numeric comparison operator is specified, the current time is compared to the time parameters as a number, by converting them each to the number of seconds since the beginning of time according to the MacOS. Therefore, you can hide text during certain hours of the day. For example, to hide text before noon, you can write:

<HIDE_TIME LT 12:00>

DYNAMIC PAGE COMMANDS: TIME & DATE

<SHOW_TIME operator time1 time2 ... >

Any HTML text that follows this command will be visible to users during the specified times. The SHOW_TIME command is analogous to HIDE_TIME and its syntax is the same.

<INSERT_DATE offset format>

The INSERT_DATE command will place the current date, as text, into your document at the specified point.

The optional offset parameter allows you to display the date offset from the date and time set on the server. This is useful for displaying the date in another time zone. The offset must be preceded by a + for a forward offset or – for a backwards offset, and is specified in hours and tenths of hours. For example,

```
Today is <INSERT_DATE>
In 30 minutes it will be <INSERT_DATE +.5>
And yesterday was <INSERT_DATE -24>
```

The format of the date that NetCloak inserts will depend upon the date format you have selected in the Date & Time control panel, as well as the current date format setting. See the section Configuring NetCloak in the User's Guide for more information.

You can override the default date format setting by specifying either SHORT or LONG as the format parameter. The Date & Time control panel allows you to configure two date formats: a Long Date format and a Short Date format, and the format parameter allows you to select which should be used to display the date. The format may be specified without specifying an offset.

Here is an example:

```
The current date is <INSERT_DATE SHORT>.
Tomorrow will be <INSERT_DATE +24 SHORT>.
```

This will display the current date with the Short Date format specified in the Date & Time control panel. For the U.S. default date format, this will result in:

DYNAMIC PAGE COMMANDS: TIME & DATE

The current date is 10/31/97.
Tomorrow will be 11/1/97.

There are several additional format parameters which allow inserting a single part of the date. These are:

DayNum - Day of the month (1 - 31)
DayName - The day name (Sunday, Monday...)
WeekNum - The week number (1 - 53)
MonthNum - The month number (1 - 12)
MonthName - The month name (January, February...)
Year - The year (1904 - 2040)

These format parameters can be used with or without the optional offset parameter. For example:

Today is <INSERT_DATE DAYNAME>.
Tomorrow will be <INSERT_DATE +24 DAYNAME>

The commands above will insert the name of the day it is today in the first line and will be tomorrow in the second line.

The parameters which insert numbers will include a leading zero if NetCloak's global date format is set to LONG.

If the Year format is specified and NetCloak's global date format is set to LONG, the year will be inserted including the century. If it is short, the year inserted will only include the two-digit year.

<HIDE_DATE operator date1 date2 ... >
<SHOW_DATE operator date1 date2 ... >
OR
<HIDE_DATE beginDate endDate >
<SHOW_DATE beginDate endDate >

HIDE_DATE allows you to hide portions of your page during the specified dates. SHOW_DATE allows you to show portions of your page during the specified dates.

When using numeric comparison operators or begin and end dates, the date parameters must be specified using the short date format configured in your system's Date & Time control panel. In the US, this format is usually set to mm/dd/yy. Partial dates, such as 1997, 11/98 or 04/15, and wildcards

DYNAMIC PAGE COMMANDS: TIME & DATE

within date values, are not allowed. Leading zeros for single-digit months or days are optional, as are the century digits of the year.

If the first parameter following HIDE_DATE or SHOW_DATE in the tag is a comparison operator, then text is hidden/shown if the current date satisfies the comparison to any of the remaining date parameters.

For example, to hide text during all of 1997 and 1998 using comparison operators, you could write:

```
<HIDE>  
<SHOW_DATE LT 1/1/97>  
<SHOW_DATE GT= 1/1/99>  
This will be hidden during 1997 and 1998.  
<SHOW>
```

If the date formats on your server are configured to always include the century digits, you could also use:

```
<SHOW><HIDE_DATE CONTAINS "1997" "1998">  
This will be hidden during 1997 and 1998.  
<SHOW>
```

To display a message only a certain day of any year, you could use (assuming a U.S. short date format):

```
<HIDE><SHOW_DATE BEGINS "12/25">  
Merry Christmas!  
<SHOW>
```

If the first parameter is not a comparison operator, then the two parameters define a date range, and text will be hidden from the begin date until the end date. To work properly, the begin date must be before the end date. Only one begin and end date may be specified.

To suppress part of a page during 1998 this way, you could use:

```
<SHOW><HIDE_DATE 1/1/98 1/1/99>  
This will be hidden during 1998.  
<SHOW>
```

A single asterisk (*) can be used as the begin date to mean "any date earlier than". Using an asterisk for the end date

DYNAMIC PAGE COMMANDS: TIME & DATE

means “any date after”. An example is shown below, and another is included in the INSERT_COUNTDOWN section later.

NetCloak allows you to easily hide or show a section of a page on any single date, by simply omitting the end date parameter. In this case, NetCloak assumes an end date exactly 24 hours from the start date. For example:

```
<HIDE><SHOW_DATE 12/14/98>
Happy Birthday, John!
<SHOW>
```

NetCloak does take the time of day into account when the HIDE/SHOW_DATE command is used. You can include a specific time of day by following the date with an at sign ('@') and the time. For example,:

```
<SHOW><HIDE_DATE * 12/14/98@5:34>
Hidden until 12/14/98 at 5:34 a.m.
<SHOW>
```

It is important to note that when dates are specified without a time, the time is actually computed as midnight on the morning of the date specified. In other words, a date of 1/1/98 technically specifies the stroke of midnight on New Year's eve, the very beginning of 1998. So 1/1/98 and 12/31/97@24:00 refer to the exact same time.

Because of this, using the numeric equality operator '=' will make the comparison including the time, even though it is not explicitly specified. Using it will result in text being shown or hidden for **one second** at the specified date and time, so you typically do *not* want to use it with SHOW/HIDE_DATE commands. In other words, this:

```
<HIDE><SHOW_DATE == 12/25/98>
```

...is *not* equivalent to this:

```
<HIDE><SHOW_DATE 12/25/98>
```

The first shows text for one second at midnight; the second shows text for 24 hours.

DYNAMIC PAGE COMMANDS: TIME & DATE

<INSERT_DAY offset>

The INSERT_DAY command will place the current day of the week, as text, into your document. NetCloak will insert the day in the format selected in the Date & Time control panel. Here is an example:

Today is <INSERT_DAY>.

For U.S. date format, this would appear in the page like this:

Today is Friday.

As in the INSERT_DATE and TIME commands, the offset parameter is an optional parameter that specifies a number of hours and tenths to offset the day. Note that the offset is not specified in days, because you may want to show the day in a different time zone. Only one offset may be specified.

<HIDE_DAY operator day1 day2 ...>

<SHOW_DAY operator day1 day2 ...>

HIDE_DAY hides the HTML text on the specified day(s). SHOW_DAY shows the HTML text following it on the specified day(s).

The comparison operator is optional. If omitted, a begins with comparison is used.

The days are specified as the first three characters of the day, and depend on the date format selected in the Date & Time control panel on the server. For U.S. format, the valid days are MON, TUE, WED, THU, FRI, SAT, SUN. As with most other commands, multiple days may be specified.

This example will hide text during the work week:

<HIDE_DAY MON TUE WED THU FRI>

<INSERT_COUNTDOWN_WDWM date@time "Weeks" "Days" "Hours" "Minutes">

Counting down to a specified day or time is handy when you have some event approaching that you would like to highlight. For example, you might want to count down the

DYNAMIC PAGE COMMANDS: TIME & DATE

days until a big sale, or a mid-term exam, or the roll out of a new feature of your Web site. Using the command is simple, just include the command and specify the date (and optionally the time) you would like to countdown to. For example:

```
The new century starts in  
<INSERT_COUNTDOWN_WDHM 01/01/2000>.
```

The INSERT_COUNTDOWN command has several variations, specified by the WDHM at the tail end of the command. The letters WDHM stand for Weeks, Days, Hours, and Minutes.

Any combination of these characters are allowed, and they give you control over exactly what is displayed. For example, to countdown the hours until Christmas of 1998, you would use:

```
Christmas '98 is in  
<INSERT_COUNTDOWN_H 12/25/1998>.
```

You may also countdown to a specific time (with time expressed in military format), as in:

```
I'll have Thanksgiving dinner in  
<INSERT_COUNTDOWN_M 11/28/1998@16:00>  
<P>  
Too many minutes? How about <INSERT_COUNTDOWN_DM  
11/28/98@16:00>?
```

Valid dates are between January 1st, 1904 and January 1st, 2030.

Also, the INSERT_COUNTDOWN command will also count up, if the date (and/or time) specified is in the past.

For example:

```
My son is <INSERT_COUNTDOWN_WD 5/6/93> old, and my  
daughter is <INSERT_COUNTDOWN_WD 12/2/94> old.
```

By default, NetCloak uses the labels Weeks, Days, Hours and Minutes when displaying the countdown. In the example above, the page would appear:

```
My son is 231 Weeks 5 Days old, and my daughter is  
149 Weeks 4 Days old.
```

DYNAMIC PAGE COMMANDS: TIME & DATE

However, you can change the labels by specifying the label parameters. If you choose to include your own labels, you should specify all four, in order. In the example above, we can use lower case labels and include a comma after weeks by changing the command to:

```
My son is <INSERT_COUNTDOWN_WD 5/6/93 "weeks,"  
"days" "hours" "minutes"> old, and my daughter is  
<INSERT_COUNTDOWN_WD 12/2/94 "weeks," "days"  
"hours" "minutes"> old.
```

Notice that even though hours and minutes won't be displayed, they are included for good form and consistency.

To avoid having the counter reverse itself and begin counting upward, you may want to use the SHOW_DATE and HIDE_DATE commands. For example:

```
<HIDE><SHOW_DATE LT 12/25/98>  
Christmas is in  
<INSERT_COUNTDOWN_WDH 12/25/98>.  
<HIDE><SHOW_DATE 12/25/98>  
Merry Christmas!  
<HIDE><SHOW_DATE GT= 12/26/98>  
Christmas is in  
<INSERT_COUNTDOWN_WDH 12/25/99>.  
<SHOW>
```

Commands: Counters

<INSERT_COUNT counter >

INSERT_COUNT will place an incremental number in the document. The number increases by one each time the command is executed.

The counter field allows you to specify a name for different counters. A name is not required, but is used so that you can maintain individual page counters. NetCloak will track the value of counters with different names separately. If no counter is specified, then a default counter is used. The counter name is not case sensitive.

Here is an example:

DYNAMIC PAGE COMMANDS: COUNTERS

Cloaked documents served: <INSERT_COUNT>

<P>

This page served: <INSERT_COUNT ThisPage>

You could include this example in any number of cloaked documents, changing the counter name ThisPage in each document. The first counter in this example (without the counter name) would be incremented each time any of these pages was accessed, whereas the second counter would be incremented for each individual page.

To display a counter without updating it, use the variation INSERT_COUNT#. For example:

This document served: <INSERT_COUNT MyPage>

<P>

That's <INSERT_COUNT# MyPage> times!

<P>

<INSERT_COUNT# MyPage> is quite a few times!

<P>

The Home Page has been accessed <INSERT_COUNT#> times.

This page would only increment the MyPage counter once, and would not increase the default counter (used, in this case, for the Home Page) at all.

You can also increment counters without displaying them by using the variation INSERT_COUNT*. This allows you to increment counters on each page of your server and then display them on a separate statistics page.

Counters may be preset to any value using the Counters tab of the Lists window. See the "Configuring NetCloak" section of the User's Guide for details. When you define a counter in the Lists window, you can also have the counter automatically reset itself at specified intervals.

NetCloak limits the number of individual counters to 512. See the section "Adjusting Internal Settings" for information on how this limit can be increased, if necessary.

**<HIDE_COUNT counter operator val1 val2 ... > OR
<HIDE_COUNT lowval highval counter >**

In addition to displaying page counters, you can show and hide based on their values.

DYNAMIC PAGE COMMANDS: COUNTERS

If the first parameter is a number, HIDE_COUNT checks the counter with the name specified by the counter parameter to see if it falls within the range defined by the lowval and highval parameters. If the counter name is omitted, the default counter is assumed. Here is an example:

```
This document served <INSERT_COUNT MyPage> times.  
<HIDE_COUNT 0 50 MyPage>  
<P>This line will be shown only after the page has  
been accessed 50 times.  
<SHOW>
```

If the first parameter is not a number, it is treated as a counter name, and the text is hidden if the counter's current value satisfies the comparison with any of the given value parameters. If the comparison operator is omitted, the counter must exactly equal one of the given values. To test the value of the default counter, use either a pair of double-quotes, or use the name Default. Thus, the previous example could also be written as:

```
This document served <INSERT_COUNT MyPage> times.  
<HIDE_COUNT MyPage LT 50>  
<P>This line will be shown only after the page has  
been accessed 50 times.  
<SHOW>
```

Note that the HIDE_COUNT command does not increment the counter, so you will normally need to use this command in conjunction with either INSERT_COUNT or INSERT_COUNT*.

**<SHOW_COUNT counter operator val1 val2 ... > OR
<SHOW_COUNT lowval highval counter >**

This command is the SHOW version of the count commands. See HIDE_COUNT for details on how it works. The SHOW_COUNT command works exactly the same way, except that text becomes shown instead of hidden. For example, the following section would give a special note to the 100th person accessing your site:

```
<INSERT_COUNT* Winner>  
<HIDE><SHOW_COUNT Winner == 100>  
You are lucky visitor number 100!  
<SHOW>
```

DYNAMIC PAGE COMMANDS: COUNTERS

<INSERT_COUNTERDATE counter offset format>

Counters are commonly reset occasionally, and NetCloak can do this for you at predefined intervals. To create a counter that periodically resets, see the “Configuring NetCloak” section in the User’s Guide.

When a counter resets, you may want to display the date the counter started. The INSERT_COUNTERDATE command gives you an easy way to do this. For example:

There have been <INSERT_COUNT PageHits> visitors to this page since <INSERT_COUNTERDATE PageHits>.

When the counter PageHits is reset by NetCloak, the page will be automatically updated to display the date the counter was reset.

The optional offset parameter allows you to display the date offset from the current date and time. This is useful for displaying the date in another time zone. The offset must be preceded by a + for a forward offset or – for a backwards offset, and is specified in hours and tenths of hours. For example,

<INSERT_COUNTERDATE PageHits +3.5>

The optional format parameter is used to control how the date appears. If needed, use LONG for the Long Date specified in the Date & Time control panel, and SHORT for the Short Date format. The format may be specified without specifying an offset. For example,

<INSERT_COUNTERDATE PageHits SHORT>

If the specified counter does not exist, the text *Unknown Counter* is inserted into the page. If the specified counter exists, but has never been reset, the text *Never Reset* is inserted into the page.

See the INSERT_DATE command for more examples and sample dates in each format.

DYNAMIC PAGE COMMANDS: COUNTERS

<INSERT_COUNTERTIME counter offset format >

The INSERT_COUNTERTIME has the same purpose and functions in the same way as the INSERT_COUNTERDATE command above. Of course, instead of displaying the date a counter was last reset, this command displays the exact time. To expand on the example above:

There have been <INSERT_COUNT PageHits> visitors to this page since <INSERT_COUNTERTIME PageHits> on <INSERT_COUNTERDATE PageHits>.

Like the INSERT_TIME command, an offset can be specified as a number of hours difference between the system date of the server and the time to be displayed. The optional format parameter can also be used to tell NetCloak whether seconds should be displayed or not.

If the specified counter does not exist, the text *Unknown Counter* is inserted into the page. If the specified counter exists, but has never been reset, the text *Never Reset* is inserted into the page.

See the INSERT_TIME command for more details.

Commands: Security

<INSERT_DOMAIN>

This command will place the user's address, as text, into your document. INSERT_DOMAIN may be a somewhat misleading name (it might better be called INSERT_ADDRESS), but is used to keep the naming consistent with the HIDE and SHOW commands where a sub-domain is normally specified as opposed to a complete address. Here is an example:

Your TCP/IP address is: <INSERT_DOMAIN>.

If you have turned on your web server's option to look up the DNS name associated with the client address for each connection, the INSERT_DOMAIN command will insert the DNS name found by the web server. Otherwise, it will insert the numeric IP address of the client.

DYNAMIC PAGE COMMANDS: SECURITY

<HIDE_DOMAIN operator domain1 domain2 ... >

<SHOW_DOMAIN operator domain1 domain2 ... >

Any HTML text that follows HIDE_DOMAIN will be hidden from those people in the domains that satisfy the comparison operator, until the next SHOW or HIDE command. Similarly, any HTML text that follows SHOW_DOMAIN will be shown to users in those domains. You can use the HIDE/SHOW_DOMAIN commands to tailor the content of your web pages to the people within your intranet or domain.

Multiple domains may be specified. For example, to hide text from sub-domains 123.456 and 123.89, use:

```
<HIDE_DOMAIN BEGINS 123.456 123.89>
```

Each domain must be separated by whitespace, and you may specify any number of domains (limited only by the maximum length of a NetCloak command – currently 4096 characters).

When the comparison operator is omitted from the command, NetCloak performs a begins with text comparison. This means that the address of the user accessing the page must begin with the given domain parameter for text to be hidden (or shown in the SHOW_DOMAIN command).

When performing text comparisons, you may specify as much or as little of the domain address as you like. For example:

```
<HIDE_DOMAIN 123.456.>
```

and

```
<HIDE_DOMAIN 123.>
```

would both hide text from the browser with the address of 123.456.78.9. The second command would also hide the text from a browser at the address 123.654.932.10, but the first would not. Specifying more of the address just makes the command more restrictive.

If you use a numeric comparison operator (= , LT , LT = , GT , GT =), you must specify the IP address parameters fully, because NetCloak converts the client address and each

DYNAMIC PAGE COMMANDS: SECURITY

parameter to an integer value to perform the comparison. Using a numeric comparison operator allows you to check for IP addresses within a certain range which may be less than a full class C block of addresses. For example, to show some text only to clients in the address range 123.45.67.1 through 123.45.67.100, you would use:

```
<SHOW>
<HIDE_DOMAIN LT 123.45.67.1>
<HIDE_DOMAIN GT 123.45.67.100>
This text is only shown to clients within the
specified IP address range.
<SHOW>
```

To use the HIDE/SHOW_DOMAIN commands with domain names instead of numeric IP addresses, you must turn on DNS lookups in your web server (in WebSTAR 2.x, check the Use DNS setting in the Misc. Settings dialog) then use one of the text comparison operators. For example, the command

```
<HIDE_DOMAIN ENDS .maxum.com>
```

would hide text from anyone coming from the domain maxum.com (as long as you have your web server set to resolve domain names).

Note: using a contains comparison can lead to unexpected results. For example, if you wanted to hide text from all clients in the .com domain you might use:

```
<HIDE_DOMAIN CONTAINS .com>
```

This would work as you would expect, but would also hide the text following it from a client in the domain home.common.edu or many.computers.se which may be undesirable. When using HIDE_DOMAIN this way, you should be as specific as possible. Usually, a begins with or ends with comparison operator is a better choice.

For backward compatibility with versions of NetCloak prior to 3.0, the syntax: HIDE_DOMAIN domain1 domain2 ... > is still supported, operating identically to a HIDE_DOMAIN CONTAINS domain1 domain2 ... > command. The new syntax is preferred.

DYNAMIC PAGE COMMANDS: SECURITY

<INSERT_USERNAME>

This command will place the username specified for basic authentication into your document at the specified point. The example below inserts the user's name:

Hello <INSERT_USERNAME>!

The username inserted is whatever the user entered into an authentication dialog box displayed by their Web browser. Normally the user will not have entered a username or password unless they have loaded a page with a NetCloak REQUEST_PASSWORD command or protected by a realm or other security on your web server. Until the user enters a username and password, the username will be blank.

<HIDE_USERNAME operator user1 user2 ...>

<SHOW_USERNAME operator user1 user2 ...>

HIDE_USERNAME hides the HTML text that follows it from clients who have entered a username that satisfies the comparison to any one of the specified username parameters. SHOW_USERNAME shows the HTML text to clients whose username satisfies the comparison.

If the comparison operator is omitted, an exact text match (an is comparison) is required. However, letter case is ignored, so bob and BoB are considered an exact match.

As with other NetCloak commands, you may specify multiple usernames. For example, to hide text from everyone who has entered the username John or Bob, use:

<HIDE_USERNAME IS JOHN BOB>

To see if the user has entered a name, you can use the command with no parameters, or you can use the EXISTS comparison operator. This will hide/show text from anyone who has entered any username. The example below will display the username, or a message if the user has not entered one yet.

DYNAMIC PAGE COMMANDS: SECURITY

```
<HIDE><SHOW_USERNAME>  
Hello <INSERT_USERNAME>!  
<SHOW><HIDE_USERNAME>  
You haven't entered a user name yet.  
<SHOW>
```

<INSERT_PASSWORD>

This command will place the user's password, as text, into your document at the specified point. For example:

You entered the password <INSERT_PASSWORD>

This command should be used with caution, as it displays sensitive information on a page. Note that the actual password, not a series of bullets, asterisks, or blanks, will be displayed in plain text on the user's browser.

<HIDE_PASSWORD operator passwd1 passwd2... > <SHOW_PASSWORD operator passwd1 passwd2... >

HIDE_PASSWORD hides the HTML text from users whose password satisfies the comparison with any one of the given passwords. SHOW_PASSWORD shows text to those same users.

If the comparison operator is omitted, an exact text match (an is comparison) is required. As always, capitalization is ignored.

As with the other NetCloak commands, you may specify a full or partial password, and you may specify any number of passwords in the command. For example, to hide text from all people with the password MOOF:

```
<HIDE_PASSWORD MOOF>
```

You can use the command without parameters, or with the EXISTS comparison operator, to see if the user has entered any password. The command HIDE_PASSWORD>, used by itself, will hide text from anyone who has entered any password. The following example will display the user's password if they have entered one, or a message if they haven't.

DYNAMIC PAGE COMMANDS: SECURITY

```
<HIDE><SHOW_PASSWORD>
Password confirmed as: <INSERT_PASSWORD>.
<SHOW><HIDE_PASSWORD>
You have not entered a password yet.
<SHOW>
```

<REQUEST_PASSWORD prompt user1,pass1 user2,pass2 ... >

This command allows you to request a password from the user for a page. If a valid username and password are not entered, NetCloak will return the No Access page instead of the requested page.

The prompt parameter used in this command is sent to the browser and will be displayed in the username/password dialog box presented to the user. The prompt text is typically displayed by the browser with a message like "Please enter a password for [prompt]."

In addition, you can specify the valid usernames and passwords right inside the REQUEST_PASSWORD command. To specify a user, enter the username and password within a pair of double-quotes, separated only by a comma (no spaces). For example, the username John with a password of Moof would be defined as John,Moof.

To allow John and a couple of other people access to a page, you might use a command like:

```
<REQUEST_PASSWORD "Name" "John,Moof" "Bob,Whopper"
"Dawn,fbr192">
```

The REQUEST_PASSWORD command requires an exact match between the specified parameters and the username and password but ignores capitalization. The command:

```
<REQUEST_PASSWORD "my server" "lo,go">
```

would allow a username of Lo or LO but not Lonnie or Carlos, and would allow a password of Go or GO but not gone or fargo.

Note that the password request can be conditional when used in conjunction with HIDE and SHOW commands.

DYNAMIC PAGE COMMANDS: SECURITY

Whenever text is hidden, the REQUEST_PASSWORD command will also be hidden and therefore not activated. In the example above, we could allow everyone access to the page on the weekend, but only allow John, Bob, and Dawn access during the week:

```
<HIDE_DAY BEGINS SAT SUN>  
<REQUEST_PASSWORD "Weekday Access" "John, Moof"  
"Bob, Whopper" "Dawn, fbr192">  
<SHOW>
```

On Saturday and Sunday, the REQUEST_PASSWORD command will be hidden, allowing everyone access to the page. During the week, a username and password will be required to gain access to the page.

Note that NetCloak's REQUEST_PASSWORD command is completely independent from your web server's built-in security or any security add-ons. In many cases, NetCloak is a simpler and more reliable way to secure a single page than realm-based or other security built into the web server. With the SHOW_USERNAME and SHOW_PASSWORD commands, NetCloak also allows you to apply password security to portions of an HTML page.

How it works: The REQUEST_PASSWORD command may seem somewhat like magic, but it is really quite simple. When NetCloak encounters a REQUEST_PASSWORD command in a page, it first checks to see if the incoming request from the user's web browser already includes a username and password that matches one of the username,password pairs in the command. If no match is found, the rest of the page is ignored. Instead of returning the page, NetCloak sends an Authorization Required response to the browser, which includes a special HTTP response header along with the contents of the configured No Access page. When this special response is received by the browser, it recognizes it as a request to display the username/password dialog box to the user, and does so. If the user then enters a valid username and password into the dialog box, the same page containing the REQUEST_PASSWORD command is simply re-requested by the browser, but this time it includes the required

DYNAMIC PAGE COMMANDS: SECURITY

username and password in the request, so NetCloak displays the page. The browser will then send the username and password in every subsequent request until the browser is quit by the user. When the user cancels the username/password dialog box, the browser simply displays the contents of the No Access page it has already received from NetCloak.

<SET_USER realm username password>

The SET_USER command is used to add a user account to a named list of known users stored by NetCloak internally, and is used in conjunction with the REQUEST_PASSWORD command to require a password to access a particular page. A REQUEST_PASSWORD command can contain the name of a list instead the actual list of valid usernames and passwords, and the SET_USER command is used to add and remove users from the list. The named lists are commonly referred to as “realms”, but NetCloak’s username and password lists have no connection to security built into your web server.

For example, assume there is a page on your server called secret.html, which needs to be password protected. The page can be protected with the command:

```
<REQUEST_PASSWORD "Secret">
```

This command will cause the web browser to open an authentication dialog asking the user to enter a username and password. Valid username and password pairs can then be created using the SET_USER command, like this:

```
<SET_USER "Secret" "Bob" "Moof">
```

It is important to note that the SET_USER command is persistent, much like global variables. In other words, when you issue the command shown above, the user Bob will be able to access the secret.html page using the username Bob and the password Moof days, months, or even years later. Once a username and password are set with the SET_USER command, the user account is maintained until deleted (see below for details on deleting user accounts).

DYNAMIC PAGE COMMANDS: SECURITY

The SET_USER command is most often used on a page that is accessed based on a form submission. This allows you to create an input form that will create user accounts granting access to password protected pages. Any of the parameters in the SET_USER command can be variables, in which case the variable value will be used as parameter text. For example, you might create the following input form:

```
<FORM ACTION="reguser.nc1k" METHOD=POST>
Username: <INPUT TYPE=text Name="username">
<P>
Password: <INPUT TYPE=text Name="password">
<P>
<INPUT TYPE="submit" VALUE="Update">
</FORM>
```

This form accepts two input fields: username and password. The reguser.nc1k page might then look like this:

```
<SET_USER "Secret" username password>
You have been registered!
```

Whenever you need to provide a new user with access to the secret.html page that includes the REQUEST_PASSWORD command shown above, you can simply open the registration form in a Web browser, enter the username and password, and submit the form. From then on, that user will have full access to the protected page.

User accounts can be removed by specifying the keyword DELETE_USER as the password. For example, to delete the user Bob that was set above, use the command:

```
<SET_USER "Secret" "Bob" DELETE_USER>
```

The user account will then be permanently removed from the realm.

NetCloak stores the user accounts in a plain text file in the NetCloak Files folder. The file for each realm is named with the name of the realm followed by .reg, and can be opened with any text editor, database, or spreadsheet application. If you need to automate user account management, quit the server (or NetCloak, if you are using the CGI version), make

the needed changes to the .reg file, save it, and restart NetCloak.

Commands: Client Type

<INSERT_CLIENT>

This command will place the name of the user's Web browser, as text, into your document at the specified point. The browser name is defined by the browser itself.

Here is an example:

Your Web browser is <INSERT_CLIENT>.

<HIDE_CLIENT operator client1 client2 ...>

<SHOW_CLIENT operator client1 client2 ...>

HIDE_CLIENT hides the HTML text from clients using any Web browser that satisfies the comparison operator. SHOW_CLIENT shows text to those clients.

These commands are most commonly used to tailor elements of a page to match the features of particular browsers. They actually work on the identification string that the browser sends with each request, which may not exactly match the browser's name. The two most popular browsers, Netscape Navigator and Internet Explorer, each have some peculiarities.

Web crawlers or "spiders" from search engines such as Excite or HotBot also identify themselves the same way that browsers do. The identification string frequently also includes the version number and platform of the browser or spider. This information creates additional uses for HIDE_CLIENT, allowing you to provide special content to Web robots that will control which links on your site are crawled or optimize their index of your pages.

The comparison operator is optional. If omitted, NetCloak performs a contains comparison, which means that any browser whose identification string contains any of the text parameters will satisfy the command.

DYNAMIC PAGE COMMANDS: CLIENT TYPE

Navigator identifies itself as Mozilla, not Navigator or Netscape. This is further complicated by the fact that most versions of Internet Explorer also identify themselves as Mozilla, followed by “compatible” and “MSIE”. Differentiating the two is no problem with NetCloak, but requires a bit of extra work. Here are the identifications of some common browsers:

```
Mozilla/2.0 (compatible;MSIE 2.1;Mac_PowerPC)
Mozilla/2.0 (compatible;MSIE 3.0;Mac_PowerPC)
Mozilla/3.0 (compatible;MSIE 4.0;Mac_PowerPC)
Mozilla/2.0 (Macintosh; I; PPC)
Mozilla/3.0 (Macintosh; I; PPC)
Mozilla/4.03 (Macintosh; U; PPC)
```

To distinguish Navigator and Explorer (not to mention other Netscape “compatible” browsers), simply add an extra HIDE or SHOW to check for the word compatible, as in:

```
You are using
<HIDE>
<SHOW_CLIENT Mozilla>
<HIDE_CLIENT compatible>Netscape Navigator
<HIDE><SHOW_CLIENT MSIE>Internet Explorer
<SHOW>
```

When using a contains text comparison, there is no need to specify the entire browser name. It is best to specify enough of the browser name to uniquely identify it without specifying so much that different versions of the same browser are treated differently (unless that is the desired effect). For example, to hide a portion of your HTML document from Navigator clients, you would probably just use:

```
<HIDE_CLIENT CONTAINS Mozilla>
```

To specify a particular version, you could use something similar to:

```
<HIDE_CLIENT CONTAINS Mozilla/2>
```

Since the name specified in the command can appear anywhere in the identification, you can use it to match other information in the string. For example, the Macintosh version of most browsers include Mac in their name, so you can use the HIDE_CLIENT command to customize your pages for

DYNAMIC PAGE COMMANDS: CLIENT TYPE

Macintosh clients. To hide text from Macintosh browsers, you can simply use:

```
<HIDE_CLIENT CONTAINS Mac>.
```

Unfortunately, it is impossible to give a complete list of Web browser identifications, since new client applications and new versions of existing ones are certainly going to become available. The best way to find out the identification for a particular browser is to set up a cloaked page with an `INSERT_CLIENT` command, and then access it from the browser in question. The page will then show you exactly what the browser gives as its name. You can also look at your web server log file to see the identification string for Web browsers used by people visiting your Web site.

Commands: Random

<INSERT_RANDOM range>

`INSERT_RANDOM` allows you to display a random number in your document. By default (without a range specified), the value will be between 1 and 100. Here are a few examples:

```
Random #(1-100): <INSERT_RANDOM>  
Random #(1-5): <INSERT_RANDOM 5>  
Random #(1-500): <INSERT_RANDOM 500>
```

<HIDE_RANDOM threshold range>

<SHOW_RANDOM threshold range>

These commands hide or show the HTML text to users based on a random number. If the random number generated by the command is less than the threshold specified, the HTML text which follows the command will be hidden or shown.

If no range is specified, `SHOW_RANDOM` will use a range of 100, and the threshold is equivalent to a percentage from 1 to 100. For example, to display a line randomly:

```
<HIDE><SHOW_RANDOM 30>  
This line seen 30% of the time.
```

DYNAMIC PAGE COMMANDS: RANDOM

Remember that each random line is independent of the others. If you place several `SHOW_RANDOM` commands in sequence, more than one of these lines may be shown, and it is also possible that none will be shown. In order to ensure that one and only one line in the group is shown on a page, use the `SAMERAND` commands explained below.

The optional parameter `range` allows you to adjust the range of possible random values. This will show the text following the command in a fraction of the times the page is loaded, equal to the threshold divided by the range. For example:

```
<SHOW_RANDOM 1 3>
```

This would show the text following it in 33% (1/3) of the times the page is served.

Specifying a range greater than the default of 100 allows for a finer level of control over the command than a single percentage, and in many cases simplifies the logic of your `HIDE` and `SHOW` commands. It also makes it possible to select from more than 100 possible options. The range for randomly generated values is 1 to 10,000.

<INSERT_SAMERAND>

This command inserts the most recently generated random number into the page without generating a new random number.

<HIDE_SAMERAND threshold>

<SHOW_SAMERAND threshold>

These commands hide or show the HTML text to users based on the random value generated in a preceding `HIDE`, `SHOW` or `INSERT_RANDOM` command. When the preceding `RANDOM` command has no range specified, the threshold is a percentage from 1 to 100.

`SHOW_SAMERAND` may only be used after a `SHOW_RANDOM`, `HIDE_RANDOM` or `INSERT_RANDOM` command, since these are the commands that generate the random value used by `SHOW_SAMERAND`.

DYNAMIC PAGE COMMANDS: RANDOM

A generated random value will trigger only a single SAMERAND command. For each show command that is not triggered, the chances of later SAMERAND commands triggering increases so that eventually one of the commands will trigger (assuming the total of the threshold values in the commands equal the range of the RANDOM command). If the total is less than the range of the RANDOM command, there will be a chance that none of the items will be shown.

For example, to show exactly one of four possible lines with equal probability:

```
<HIDE><SHOW_RANDOM 25>
This is line 1.
<HIDE><SHOW_SAMERAND 25>
This is line 2.
<HIDE><SHOW_SAMERAND 25>
This is line 3.
<HIDE><SHOW_SAMERAND 25>
  This is line 4.
<SHOW>
```

In this case, a random value is generated, between 1 and 100. If the value is between 1 and 25, the first line will be displayed. If the random value is between 26 and 50, the second line will be displayed. The third and fourth lines will be handled similarly. If you want to display the random value as well, use:

```
The random # is: <INSERT_RANDOM>
<P>
<HIDE><SHOW_SAMERAND 25>
This is line 1.
<HIDE><SHOW_SAMERAND 25>
This is line 2.
<HIDE><SHOW_SAMERAND 25>
This is line 3.
<HIDE><SHOW_SAMERAND 25>
This is line 4.
<SHOW>
```

Here is another example. In this case, one graphic will be randomly displayed:

DYNAMIC PAGE COMMANDS: RANDOM

```
<HIDE><SHOW_RANDOM 2>
<IMG SRC=/Images/VeryRare.GIF>
<HIDE><SHOW_SAMERAND 5>
<IMG SRC=/Images/Rare.GIF>
<HIDE><SHOW_SAMERAND 15>
<IMG SRC=/Images/Occasional.GIF>
<HIDE><SHOW_SAMERAND 30>
<IMG SRC=/Images/Common.GIF>
<HIDE><SHOW_SAMERAND 48>
<IMG SRC=/Images/Frequent.GIF>
<SHOW>
```

If a range other than the default has been specified in the preceding RANDOM command, the text following will be shown a fraction of the times the page is served defined by the threshold in the SHOW_SAMERAND command and the range in the preceding RANDOM command. For example:

```
<HIDE><SHOW_RANDOM 1 3>
This will only be shown 1/3 of the time.
<HIDE><SHOW_SAMERAND 1>
This will be shown 1/3 of the time when the first
line is not.
<HIDE><SHOW_SAMERAND 1>
This will be shown the remaining 1/3 of the time.
```

Commands: File Information

<INSERT_MODIFIED filepath offset format>

The INSERT_MODIFIED command allows you to easily insert the date that the given file was last modified.

The filepath parameter may be either relative to the Root Folder, or relative to the location of the page being served. It may be specified using either the URL style for file paths (using slash '/' characters to delimit folder names), or the MacOS style (using colons ':' to delimit folder names). If the first character of the file path is a colon or a slash, the file path is assumed to be relative to the Root Folder. Otherwise, it is assumed to be relative to the location of the page being served. For example, if the Root Folder is MacHD:Web:, then the command:

```
<INSERT_MODIFIED "/news/latest.html">
```

...will always insert the last-modified date of the file:

DYNAMIC PAGE COMMANDS: FILE INFO

MachD:Web:news:latest.html

If the command:

```
<INSERT_MODIFIED "local/weather.html">
```

...is located in the file:

MachD:Web:usa:indiana:default.html

...then NetCloak will insert the last-modified date of the file:

MachD:Web:usa:indiana:local:weather.html

...into the web page when it is served. If the file path contains spaces, be sure to put it in double-quotes.

If the filepath parameter is missing, then the command will display the last modified date of the current page being served. For example, many pages may end in:

This page last updated on <INSERT_MODIFIED>.

The remaining two optional parameters works the same way as they do in the INSERT_DATE command.

The offset parameter lets you specify an offset from the file's modification date, in fractions of hours. For example, the following command:

```
<INSERT_MODIFIED "news.html" +168>
```

...will insert a date 7 days after the file newstuff.html (located in the same folder as the current page) was last modified.

Why is this useful? When this command is nested inside a SHOW_DATE command, it gives you an easy way to automatically highlight files for a period of time whenever they are updated on your web site. The following code will automatically append a New! graphic to the hypertext link for seven days whenever the linked page is updated:

```
<A HREF="news.html">Recent News</A>
<HIDE><SHOW_DATE LT
  <INSERT_MODIFIED "news.html" +168>>
<IMG SRC="new.gif">
<SHOW> <BR>
```

DYNAMIC PAGE COMMANDS: FILE INFO

The format of the date inserted will depend on the date format you have selected in the Date & Time control panel, as well as the date format setting in NetCloak. See the “Configuring NetCloak” section in the User’s Guide for more information on the way that NetCloak formats dates.

You can override the default date format setting by specifying either SHORT or LONG in the format parameter. The Date & Time control panel allows you to configure two date formats: a long format and a short format, and the format parameter allows you to select which should be used to display the date. The format may be specified without specifying an offset.

The command can also be particularly useful in macros, where a single macro (perhaps a footer) may be used on many pages. In this case, the INSERT_MODIFIED command need only appear in a single macro but will always display the modification date of the page that has been served.

Note that if you use the INSERT_MODIFIED command inside a page that is not stored on disk (such as from a back-end CGI), NetCloak will not be able to determine a modified date and will insert the current date instead.

<INSERT_FILESIZE filepath>

This command inserts the size of the specified file as it is stored on disk. If a filepath is not specified, the size of the page being served is given. There are three variations of the command, including INSERT_FILESIZE_B, INSERT_FILESIZE_K, and INSERT_FILESIZE_M for displaying the size of the file in either bytes, kilobytes, or megabytes, respectively.

The filepath parameter may also refer to any file on the web server. Like all paths specified in NetCloak, when the path begins with a slash (/), the path is relative to the root folder defined in the NetCloak Configuration window. Here are some examples:

```
<INSERT_FILESIZE_B /MyFile.sit.hqx>  
<INSERT_FILESIZE_K /Folder/Moof.hqx>  
<INSERT_FILESIZE_M /BigFile.sit.hqx>
```

DYNAMIC PAGE COMMANDS: FILE INFO

The first example is a file (MyFile.sit.hqx) that resides in the root folder and the size is displayed in bytes. The second example gives the size, in kilobytes, of a file in a folder named Folder contained in the root folder. The last example shows the number of megabytes used by a file in the root folder.

Displaying the size in bytes will always show the exact size of the file, while displaying kilobytes will cause NetCloak to round up to the nearest kilobyte. When displaying the size in megabytes, NetCloak will round up to the nearest tenth of a megabyte.

Commands: Server Information

<INSERT_HOST>

Most browsers now support the Host field in the HTTP header. The Host field specifies the name of the server that the user has requested the document from. In most cases, this field will be the domain name of your server, or its IP address if the requested URL was specified by IP address.

This command can be very helpful in two situations. First, if your server has multiple domain names associated with it, the Host field is an excellent way to find out which domain name was requested by the client. For details on supporting multiple domains with the Host field, see the SHOW_HOST section below. Second, if you serve your Web site from multiple servers, you can create pages that are sensitive to the server they are being served from.

The INSERT_HOST command simply inserts the domain name (or IP address) that the URL specified when requesting the page.

Note that some older browsers do not send the Host field in the HTTP header. When pages are accessed by these clients, the INSERT_HOST command will not insert anything.

<HIDE_HOST operator host1 host2 ...>

The HIDE_HOST command allows you to hide portions of a page based upon the domain name (or IP address) as

DYNAMIC PAGE COMMANDS: SERVER INFO

specified in the URL used to access the page. Text after the command will be hidden if the requested host satisfies the comparison with any of the host parameters.

If the comparison operator is omitted from the command, a contains text comparison is used, so that a host name containing any one of the following parameters causes text to be hidden.

<SHOW_HOST operator host1 host2 ...>

The SHOW_HOST command allows you to hide portions of a page based upon the domain name (or IP address) as specified in the URL used to access the page. Text after the command will be shown if the requested host satisfies the comparison with any of the host parameters.

If the comparison operator is omitted from the command, a contains text comparison is used, so that a host name containing any one of the following parameters causes text to be hidden.

For example, assume your web server is used to serve pages for three different domains: www.greg.com, www.peter.com, and www.bobby.com. Each of these domain names points to the same address, which is the address of your server, and each web site is served from a sub-folder within the root folder (called greg, peter, and bobby). The default page for your server, at the root level, could be:

```
<HIDE><SHOW_HOST IS "www.greg.com">
<REDIRECT "http://www.brady.com/greg/">
<HIDE><SHOW_HOST IS "www.peter.com">
<REDIRECT "http://www.brady.com/peter/">
<HIDE><SHOW_HOST IS "www.bobby.com">
<REDIRECT "http://www.brady.com/bobby/">
<SHOW>
```

The requested host is unknown. Please select from the following list of domains:

In this case, the SHOW_HOST command is used to select a REDIRECT command which sends the client to the correct default page for the domain. If the Host isn't recognized (most likely due to lack of support for the field in the

DYNAMIC PAGE COMMANDS: SERVER INFO

browser), a default page is shown to the user so they can select the domain they are looking for.

Note that the MACRO command could be used in place of the REDIRECT commands to reduce processing, although the relative paths for URLs in the individual Home Pages could be confused. See the MACRO command for additional information.

<INSERT_HEADER>

The INSERT_HEADER command inserts the entire HTTP request header into the HTML document.

It is rare that you would want to display the HTTP header of an incoming request back to the user. There is, however, at least one very good reason for doing this. As new browsers become available with new HTTP features, you will want to add support for them into your documents. By using the INSERT_HEADER command in a few test pages, you can see exactly what information is being sent by various browsers. This makes it easy to try out different browsers on your pages to see just what the browser is sending so that you know what to look for when you are developing new pages.

The usage of this command is simple, and looks like this:

```
<PRE>
<INSERT_HEADER>
</PRE>
```

We use the PRE > HTML tag so that line breaks are displayed on the browser correctly. These formatting tags are not required, but the header will most likely be difficult to read without them.

<HIDE_HEADER operator text1 text2 ... >

The HIDE_HEADER command will hide HTML following it if the HTTP request header satisfies the comparison with any of the given text parameters.

DYNAMIC PAGE COMMANDS: SERVER INFO

If the comparison operator is omitted, a contains text comparison is performed, so text is hidden if the HTTP request header contains any one of the given text parameters.

The Web is based upon the HTTP protocol, which is constantly evolving. Every request that comes into your server is in the form of an HTTP header request that may or may not have additional information that is not captured by some other NetCloak command. The HEADER commands give you the ability to look into the header even when there is no direct NetCloak command for checking a particular piece of information. Let's look at the SHOW version of these commands for an easier explanation of how this works.

<SHOW_HEADER operator text1 text2 ... >

The SHOW_HEADER command will "turn the stream on" when the HTTP request header satisfies the comparison with any of the given text parameters.

The specified text can be an acceptable file type or language, the requested URL, or any other information included in incoming HTTP requests.

If the comparison operator is omitted, a contains text comparison is performed, so text is shown if the current HTTP request header contains any one of the given text parameters.

For example, many browsers support a field which specifies the languages they can accept and display. If you would like to selectively serve files in various languages, you can use this field to automatically select text to be displayed. For example, to display a section of text in French to browsers that support it, you might use:

```
<HIDE><SHOW_HEADER CONTAINS "Accept-Language: fr">  
Bon jour!  
<FLIPMODE>  
Hello!  
<SHOW>
```

In this example, if the browser sends the appropriate field, the French text will automatically be displayed by NetCloak.

DYNAMIC PAGE COMMANDS: SERVER INFO

The HTTP header can also be used to help determine file types the browser can accept and more. Essentially any information that the browser sends to the server is yours for the asking.

<SET_HEADER "HTTP Header Line">

Inserts a line (field) into the HTTP header when the page is returned to the client. The entire HTTP header line must be specified, normally according to the format:

FieldName: FieldValue

For example, to add a line to the HTTP header that will specify a user agent, you could use the command:

```
<SET_HEADER "User-Agent: SomeUserAgent/1.0">
```

Additional header fields may be used for cache control, supporting advanced HTTP features, etc. Header fields normally generated in the HTTP response by NetCloak will be over-ridden by the SET_HEADER command. For example, a page of text that must be served by NetCloak but is not an HTML file could include the command:

```
<SET_HEADER "Content-type: text/plain">
```

This would cause NetCloak to send text/plain as the file's content type instead of the normal text/html.

Finally, header lines may be generated dynamically using local variables and then inserted into the HTTP response header by referencing the variable name. For example:

```
<SET_LOCAL ExpHeader = "Expires: " &  
  <INSERT_DATE +48 LONG>>  
<SET_HEADER ExpHeader>
```

<INSERT_REFERRER>

This command will insert the URL of the last page visited into your document at the specified point. The URL will be exactly as reported by the browser, including server name and path and search arguments.

DYNAMIC PAGE COMMANDS: SERVER INFO

Here is an example:

```
You just came from: <INSERT_REFERRER>.  
If you want, you can:  
<A HREF="<INSERT_REFERRER>">Go Back!</A>
```

Note that this command will not work if the browser does not report the last page the user visited. In general, the referrer will be reported when the user clicks on a hypertext link to access the page. However, if the user manually enters the URL, the browser may not send the referrer, since the page the user was on when they entered the URL doesn't actually reference the page being served.

You can avoid giving the user a link to a non-existent referring page using a HIDE_REFERRER or SHOW_REFERRER command, described below.

Since REFERRER is commonly misspelled REFERER (including in early versions of the Internet specification for HTTP), NetCloak also supports that spelling for all REFERRER commands.

<HIDE_REFERRER operator referrer1 referrer2 ...>

This command hides the HTML text from users based on the last HTML document they accessed (the document that "referred" the user to this one). This allows you to tailor your document depending on where the user has been.

If the comparison operator is omitted, a contains text comparison is performed with the listed referrer values.

Note that if a user types an URL into their Web browser to get to your page, there will be no referrer, since the page the user was on when they entered the URL doesn't actually reference the page being served.

Some browsers may not support the Referrer field, and if the browser doesn't send it, NetCloak won't be able to use it. You can check for the existence of the field by using the command alone, with no referrers specified, or by using the EXISTS comparison operator. In other words,

DYNAMIC PAGE COMMANDS: SERVER INFO

HIDE_REFERRER> or HIDE_REFERRER EXISTS> will hide the text following it whenever a referrer is known.

<SHOW_REFERRER operator referrer1 referrer2...>

HTML text that follows this command will be visible to users when the most recent document they visited in their web browser satisfies the comparison with at least one of the specified referrer parameters.

If the comparison operator is omitted, NetCloak will show text if the referring page's URL contains any one of the listed referrer values.

For example, if your site was selected as the Cool Site of the Day, you might add this to your Home Page:

```
<HIDE>
<SHOW_REFERRER CONTAINS Cool>
Welcome, Cool Site of the Day users!
<SHOW>
```

This would show the text only to people who had just come from a page with the word Cool in the URL.

Use SHOW_REFERRER> with no parameters, or SHOW_REFERRER EXISTS>, to make sure the browser supports the referrer field and avoid providing a broken link to a nonexistent referrer, like this:

```
<HIDE><SHOW_REFERRER>
You just came from <INSERT_REFERRER>.
<A HREF="<INSERT_REFERRER">Go back there!</A>
<FLIPMODE>
Sorry, your browser does not support referrers, or
you entered the URL for this page manually.
<SHOW>
```

<INSERT_THISURL>

The INSERT_THISURL command ("This URL"), which accepts no parameters, will cause the file path and document name of the current page to be inserted into the page. For example, if the URL for a page is:

`http://www.web.com/bob/page.html$path?qry`

DYNAMIC PAGE COMMANDS: SERVER INFO

the `INSERT_THISURL` command would insert:

```
/bob/page.html
```

With servers that support virtual hosting of multiple Web sites, the path will be relative to the root folder for the requested site.

The `INSERT_THISURL` could be used in hidden fields on HTML forms to tell a forms processing CGI the URL of the entry form. It might also be used to create a “reload this page” link, or to give the user additional feedback on how your site is set up.

<HIDE_THISURL operator url1 url2 ... >

If the URL of the current page satisfies the comparison with any of the listed URL parameters, then this command shuts off the flow of text to the browser.

If the comparison operator is omitted, then the comparison is satisfied when the URL of the current page contains any one of the given URL parameters.

With the `THISURL` commands, NetCloak allows you to essentially create self-aware pages. By using `SHOW_THISURL` and `HIDE_THISURL`, you can create pages that change when they are moved from one folder (or server) to another. These commands can also be used in your macro file to create macros that vary depending on the page they are used on. In short, they allow you to create more portable, re-usable HTML.

<SHOW_THISURL operator url1 url2 ... >

This is the `SHOW` version of the `HIDE_THISURL` command above.

Here's an example:

DYNAMIC PAGE COMMANDS: SERVER INFO

```
<HIDE><SHOW_THISURL CONTAINS TestFolder>  
This page is in the folder 'TestFolder', and is  
still being tested. When the file is moved into  
its final place on my site, this test message will  
disappear.  
<FLIPMODE>  
This page is now in production use.  
<SHOW>
```

When a non-existent page is requested and the error page is served, `THISURL` is the requested (non-existent) URL. This makes the `SHOW_THISURL` command useful for making a smart error page. If users have a typo in their URL, the error page can automatically redirect them to the page they were looking for.

```
<HIDE><SHOW_THISURL CONTAINS faq>  
<REDIRECT  
"http://my.server/questions/faqlist.html">  
<SHOW>
```

You can use a similar technique to automatically redirect users accessing obsolete pages by putting the URL of the obsolete page in the `SHOW_THISURL` command and the new URL in the `REDIRECT` command.

With the example above on your error page, users who request a page that does not exist but contains FAQ will be automatically redirected to the FAQ List page.

Path and Search Arguments

In an HTTP URL, there are three parts of the URL that come after the server name. The first is a path to the requested file. The second part, called the path arguments, follows the name of the requested file. In MacOS web servers, the path arguments are separated from the filename by a dollar sign character ('\$'). Following the path arguments is the third part of the URL, called the query arguments or search arguments. The search arguments are separated from the path arguments (or the filename, if there are no path arguments) by a question mark character ('?'). Both path arguments and search arguments are URL encoded by browsers (in other words, all non-alphanumeric characters are turned into a percent sign '%' followed by the two-digit hexadecimal ASCII

DYNAMIC PAGE COMMANDS: SERVER INFO

code for the character). Here is an example URL containing both path arguments and search arguments:

```
http://www.maxum.com/netcloak/sample.html$path%20Args?search%20Args
```

In this example the path arguments consist of the text path arguments (which have been URL encoded, turning the space character into %20), and the search arguments consist of the URL encoded text search Args. It is important not to confuse path arguments with the requested file path. In the example above, the file path is /netcloak/sample.html.

Path and search arguments can be used to send additional data to NetCloak to be used in processing cloaked pages, simply by appending them to the URL of any cloaked page, then using the following commands within the cloaked pages.

<INSERT_PATH>

The INSERT_PATH command inserts the path arguments for the requested URL into the cloaked page. If there are no path arguments in the requested URL, this command does nothing.

If the file /netcloak/sample.html in the above example contained the following HTML:

```
This text follows the dollar sign in the requested URL: <INSERT_PATH>
```

The page seen at the browser would contain:

```
This text follows the dollar sign in the requested URL: path args
```

Important Note: If you are using the NetCloak CGI by calling it directly in your URLs, like this:
`http://www.yourdomain.com/NetCloak.cgi$/somefile.html`, then the path arguments are used to send the URL of the cloaked file to NetCloak, and are not available for you to use for other purposes. In this case, the INSERT_PATH command will insert the path to the cloaked file.

DYNAMIC PAGE COMMANDS: SERVER INFO

<HIDE_PATH operator path1 path2 ... >

The HIDE_PATH command turns off the flow of text in the page if the path arguments in the requested URL satisfies a comparison with any of the path parameters listed in the command. See SHOW_PATH for an example where this could be useful. If the comparison operator is omitted, a contains text comparison is performed.

<SHOW_PATH operator path1 path2 ... >

The SHOW_PATH command turns on the flow of text in the page if the path arguments in the requested URL satisfy the comparison with any of the parameters listed in the command.

When no comparison operator is used, the flow of text is turned on when the path arguments contain any one of the parameters listed.

The SHOW_PATH command can be useful when you want one page to contain all the logic for some interactive process, like an online questionnaire or game. Rather than spreading the logic for the interaction across multiple cloaked pages, you can put it all in one cloaked page, and use path arguments to determine which part of the process to send to the browser.

For example, a form can be submitted and processed by the same page, based on the path argument. The form command looks like this:

```
<FORM METHOD=POST  
ACTION="regkillerapp.html$step2">
```

Notice the path argument specified, Step2. Later in the page, the form is processed by enabling one section of HTML using the NetCloak command:

```
<HIDE><SHOW_PATH IS "step2">
```

Similarly, the form itself must be hidden once it has been submitted. This can be done like this:

DYNAMIC PAGE COMMANDS: SERVER INFO

```
<HIDE>  
<SHOW_PATH IS "step1">  
<SHOW_PATH #EXISTS>
```

This shows the form when either the first step is specified by the path argument, or when no path argument exists (as would be the case when the form is linked to from another page).

<INSERT_SEARCH>

The INSERT_SEARCH command inserts the search arguments for the requested URL into the cloaked page. If there are no search arguments in the requested URL, this command does nothing.

If the file /netcloak/sample.html in the INSERT_PATH example contained the following HTML:

This text follows the question mark in the requested URL: <INSERT_SEARCH>

Then the browser would see:

This text follows the question mark in the requested URL: search args

Note: When an HTML form specifies a method of GET instead of POST, the data in the form fields are appended to the URL specified in the action of the form, as search arguments. The SEARCH commands can be used to look at the raw form data. See the section “Commands: User Variables” for more information.

```
<HIDE_SEARCH operator search1 search2 ...>  
<SHOW_SEARCH operator search1 search2 ...>
```

HIDE_SEARCH turns off the flow of text in the page if the search arguments in the requested URL satisfy the comparison with any of the parameters listed in the command. SHOW_SEARCH turns on the flow of text if the search arguments in the requested URL satisfy the comparison.

DYNAMIC PAGE COMMANDS: SERVER INFO

A contains comparison is performed if no comparison operator is used.

Search arguments typically are used (as their name implies) to send text to be searched for, either in a database or in the files of the web site. But with NetCloak, they can be used just like path arguments, for just about any purpose. Here's an example where we use NetCloak commands to add better feedback and a consistent look-and-feel to the results returned from a simple file-searching CGI. If no search arguments are provided, we avoid calling the CGI altogether:

```
<HTML>
<HEAD><TITLE>Search Results</TITLE></HEAD>
<BODY>
<MACRO Header>
<HIDE_SEARCH #EXISTS>
<H1>Search Results</H1>
```

Here are all the matches found for the phrase
"<INSERT_SEARCH>":<P>

```
<HR>
<EXEC_CGI "SimpleSearch.cgi">
<!-- The CGI returns raw results -->
<HR>
<FLIPMODE>
<P>No search arguments were supplied. Please try
again.<P>
<MACRO Footer>
</BODY>
</HTML>
```

Commands: NetCloak Configuration

**<SET_NETCLOAK setting = value> OR
<NETCLOAK setting = value>**

The SET_NETCLOAK (or NETCLOAK) command is used to set many of NetCloak's configuration options for a single page, or even for only part of a page. These options, when set using this command, will supersede the default settings that have been defined in NetCloak. However, the revised setting will only be in effect for the page in which the SET_NETCLOAK command is used. Other pages being served

DYNAMIC PAGE COMMANDS: CONFIGURATION

will be unaffected by temporary configuration changes made with the SET_NETCLOAK command.

Here are the configuration options that can be set using this command. For each setting, the list of valid values is given.

Translation — This sets the character translation mode for text following the command. Possible values are:

Entities	Convert Mac characters to HTML Entities.
Latin	Convert Mac characters to the ISO-Latin-1 character set.
None	No character conversion.

For example, to have NetCloak perform MacRoman to ISO-Latin-1 character translation, you could use the command:

```
<SET_NETCLOAK Translation = Latin>
```

NewFileTranslation — Tells NetCloak what character translation to perform, if any, when creating new files with a CREATEDOC directive. The options for use with this setting are:

None	Perform no character conversions.
MacRoman	Convert “extended” ASCII characters (with ASCII values greater than 128) from the ISO-8859-1 (ISO-Latin-1) character set to the standard MacRoman character set, so that the file is in a standard Macintosh text format. When such characters are viewed in a Macintosh text application, they will appear as the expected Mac extended characters (the apple symbol, the bullet symbol, etc.). When viewed in a web browser, however, unexpected “garbage characters” may appear—particularly if the browser is running on something other than the Mac OS.
Entities	When this option is selected, extended characters in the form data are converted into the HTML-defined “entity” codes (such as “ ”), so that they appear

DYNAMIC PAGE COMMANDS: CONFIGURATION

correctly in HTML documents viewed in any Web browser.

LastMod – This allows you to change the Last Modified field of the HTTP header for the page on which the command appears. Possible values include:

- | | |
|-------------|---|
| File | Return the file's last modified date in the HTTP header. |
| Now | Return the current date and time in the last modified field of the HTTP header. |
| None | Leave the last modified field out of the header entirely. |

For example, assume you have a page with numerous sections of random content, but you have set NetCloak to return the file's modification date as the Last-Modified date of the HTTP header, by default. For the page in question, you can override this default by including the command:

```
<SET_NETCLOAK LastMod = Now>
```

This will cause the browser to recognize the fact that the page is constantly changing, and perform caching appropriately.

TimeFormat – This allows you to change the format of time values in the current page. Possible values include:

- | | |
|--------------|--|
| Short | Use the short time format when displaying time values. |
| Long | Use the long time format when displaying time values. |

DateFormat – This allows you to change the format of date values in the current page. Possible values include:

- | | |
|--------------|--|
| Short | Use the short date format when displaying date values. |
| Long | Use the long date format when displaying date values. |

Logging – Lets you enable or disable logging of pages on a per-page basis. Possible values include:

- | | |
|------------|-----------------------|
| Off | Turn hit logging off. |
|------------|-----------------------|

DYNAMIC PAGE COMMANDS: CONFIGURATION

Standard	Enable logging of date, time, status, client IP, script name, reply length.
Extended	Enable logging of standard information plus username, password, referrer, and user-agent.

ServerField – This text value is sent as the value of the Server: field in NetCloak's HTTP reply header.

PragmaNoCache – This determines whether or not NetCloak will tell the browser (and any other caching mechanisms in between) to not cache the page.

True	Include a "Pragma: no-cache" field in the HTTP reply header for dynamic pages.
False	Do not include a "Pragma: no-cache" field in the HTTP reply header.

DuplicateFiles – Tells NetCloak what to do when creating files via a CREATEDOC directive, and an existing file with the same name in the same folder already exists. Valid options for this setting are:

Overwrite	Always overwrite the existing file with the newly-created file.
Unique	Never overwrite the existing file; instead, a new file with a unique filename is created. The filename consists of the original filename with two alphanumeric characters appended to it.
Smart	Use "smart resubmit"; this means that the existing file will only be overwritten if it is less than 15 minutes old, and was submitted by the same user (IP address).

CreatorCode – The four-character creator signature NetCloak uses for created files. The creator code determines the icon of the file, as well as which application launches when the file is double-clicked.

ConvertBrackets – This allows you to determine how NetCloak will handle angle bracket characters in form field data when they are inserted into the resulting page. Options for this setting:

DYNAMIC PAGE COMMANDS: CONFIGURATION

True	Convert angle bracket characters to the corresponding “<” and “>” HTML entities.
False	Do not convert angle bracket characters. If you want to allow users to include HTML commands in submitted form fields, use this setting.

LinkNextText – Defines the text displayed in the hypertext link created by the LINKNEXT > tag in created documents. The value of this configuration setting can be any string of text not including double-quote characters.

LinkPrevText – Defines the text displayed in the hypertext link created by the LINKPREVIOUS > tag in created documents. The value of this configuration setting can be any string of text that does not include double-quote characters.

< INSERT_NETCLOAK setting >

INSERT_NETCLOAK inserts the current value of the specified configuration parameter into the page. In addition to the configuration settings listed above, this command also lets you insert the following information into the page:

Version – Inserts the NetCloak version number.

UpSince – Inserts the date/time NetCloak was last launched.

Hits – Inserts the total number of pages processed by NetCloak since it was last started.

CacheHits – Inserts the total number of pages served from cache.

LastHitTime – Inserts the date/time of the last hit.

LastHitType – Inserts the action name of the last hit.

LastHitHost – Inserts the hostname of the last hit.

LastHitPath – Inserts the file path of the last hit.

These commands show or hide the HTML that follows them (until the next SHOW/HIDE command) if the specified configuration parameter satisfies the comparison to the given

DYNAMIC PAGE COMMANDS: CONFIGURATION

value. As with other commands, you may specify a list of values to test against. If the comparison operator is omitted, an exact text match is required.

For example:

```
<SET_NETCLOAK DateFormat = Long>
```

This command will cause NetCloak to display all dates inserted by NetCloak commands in the remainder of the page using the Long Date format defined in the server's Date & Time control panel.

```
<HIDE><SHOW_NETCLOAK Logging IS "Off">  
Logging is currently disabled.  
<FLIPMODE>  
Logging is currently enabled.  
<SHOW>
```

This series of commands will tell the user whether or not their web server access is being logged by NetCloak.

The SHOW and HIDE_NETCLOAK commands are also useful in combination with the SET_NETCLOAK command to make conditional changes to NetCloak configuration settings.

Commands: Variables

About Variables

NetCloak supports four distinct types of variables: Local, User, Global, and Cookies. The first three types are stored by NetCloak. Cookies are different from the others in that they are stored at the client, so they will be discussed separately later.

Variables have a name and a value. A variable name consists of alphanumeric characters, may not contain quote marks, expression operators (+, -, *, /, &), the '#' character, or whitespace characters, and may not begin with a digit or an exclamation point. A variable may have any text or numeric value.

Local variables can be used only within a single HTML page. Local variables can be set at the top of the page and used

DYNAMIC PAGE COMMANDS: VARIABLES

repeatedly within the page, but once the page has been served completely the variable is forgotten. Local variables are useful for storing data that only needs to be used on another part of the same page, such as in math functions, simplifying complicated SHOW/HIDE expressions, passing variables into macros, looping in a REPEAT command, etc.

User variables remain in effect for a particular user during the time they are visiting your site. When a form is submitted to NetCloak, the fields on the form become NetCloak user variables automatically. User variables will not affect users other than the user they are set for, but they persist between page accesses for up to one hour or until another form is submitted by the same user. User variables are tracked either by the visitor's IP address or by a default cookie that can be generated by NetCloak. Note that when the user does not have a browser which supports cookies, then user variables will be tracked by IP address even if you set the default cookie. See the SET_COOKIE command description for information about creating a default cookie.

Global variables are variables that are retained indefinitely and have a single value for all pages being served to all users. You can use globals as mini-macros, making short blocks of text (a few characters or words) easy to use throughout your site. Global variables also allow you to have the actions taken by one user on your site to impact others. For example, you could have a form that allowed you, as the Webmaster, to turn on and off various portions of your Web site, by showing and hiding links to pages based on global variables that were affected by form submissions. You could also create multi-user games, implement opinion polls, and more.

Global variables can be created and updated one of three ways:

- using a SET_GLOBAL command in any web page.
- using the "Variables" dialog box of the NetCloak CGI application or the "Variables" page of the NetCloak web administration interface. This makes using global variables extremely simple and convenient for use as short macros.

DYNAMIC PAGE COMMANDS: VARIABLES

- via AppleScript, which enables *any* application capable of sending AppleEvents to affect the state of pages on your web site. See “Accessing Global Variables using AppleScript” later in this Guide.

Accepting Forms To Set Variables

NetCloak allows you to create HTML forms that ask the user to enter information and then use that information in your pages. The contents of the form are then automatically used to create User Variables.

Each field of your HTML form will have a name. Once the form is submitted, each field becomes a NetCloak User variable with the same name, whose value is the value entered into the field by the user. The variable can then be referenced by name on subsequent pages. You can either show the variables to the user in the text of your HTML document, or use SHOW and HIDE to have your page change based on what the user entered into the fields of the form.

The first step is to create an input form that uses the POST method for form processing. In the ACTION field of your FORM command, you will need to specify the URL of a document that will be processed by NetCloak. NetCloak will store data entered on the form and then serve the file specified. Here is an example of an HTML form:

DYNAMIC PAGE COMMANDS: VARIABLES

```
<FORM method=POST action="/VarDisplay.html">
Please fill out this simple form:<P>
Name:
<INPUT TYPE="text" NAME="YourName" SIZE="32"
MAXLENGTH="32">
<P>
Organization:
<SELECT name="Organization">
<OPTION selected>Commercial
<OPTION>K12 Education
<OPTION>Higher Education
<OPTION>Government
</SELECT>
<P>
Check here <INPUT NAME="MyCheckBox" TYPE=checkbox
VALUE="CHECKED">
<INPUT TYPE=submit VALUE="Submit Form">
<INPUT TYPE=reset VALUE="Start Over">
</FORM>
```

In this example, the action specified is /VarDisplay.html, which is simply the name of the html file we would like served when the user clicks the Submit Form button. Of course, the page will need to be served by NetCloak, so the file name specified in the action must end with an extension that is mapped to NetCloak. The example assumes that the web server is configured to cloak all HTML files.

Once a form has been processed, the fields on the form become User variables that may be used within cloaked documents.

Some form variables can have multiple values. These are called multivalue variables. An example is a set of checkboxes that all have the same field name but different values, or a SELECT field with the MULTIPLE attribute (which is usually displayed as a list box that allows multiple selections). NetCloak allows you to access all of the values of a multivalue variable, using square brackets after the variable name which include the number of the instance you want to use. For example, to access the third instance of a variable named Toppings, you could write:

```
<INSERT_VARIABLE Toppings[3]>
```

DYNAMIC PAGE COMMANDS: VARIABLES

Furthermore, you can use the name of another variable within the square brackets to access an instance of a multivalue variable, like this:

```
<INSERT_VARIABLE Toppings[gDailySpecial]>
```

This notation is supported in all NetCloak INSERT, SHOW, and HIDE commands, as well as the REPEAT command.

<SET_LOCAL varName = value>

<SET_VARIABLE varName = value>

<SET_GLOBAL varName = value>

Variables can be assigned values either by user input or by using the SET_LOCAL, SET_VARIABLE, or SET_GLOBAL commands. Each of the SET commands sets the variable VariableName to a given value. The equal sign (=) is optional. The Value can be a string, a number, or another local, user, global, counter or cookie variable (see below for information about cookies). In addition, five operators can currently be used to create a value from an expression. They are:

- | | |
|---|---|
| + | Performs simple addition. |
| — | Subtracts the second value from the first. |
| * | Multiplies two values. |
| / | Divides the first value by the second. |
| & | Concatenates two values (Adds the strings). |

For example, to put the value of 2 + 2 into the local variable Four, you could use the command:

```
<SET_LOCAL Four = 2 + 2>
```

Another way to put the number 4 into the variable Four would be:

```
<SET_LOCAL Two = 2>
```

```
<SET_LOCAL Four = Two * 2>
```

We could also set a variable called TwentyTwo by doing something like:

DYNAMIC PAGE COMMANDS: VARIABLES

```
<SET_LOCAL Two = 2>
<SET_LOCAL TwentyTwo = Two & "2">
```

When performing numeric operations in SET commands, NetCloak gives the resulting variable the same number of decimal places as the parameter with the *most* decimal places on the right-hand side of the equal sign. Thus, if no decimal places are given, then NetCloak will display no decimal places when the resulting variable is INSERTed into a page. Conversely, the following code:

```
<SET_LOCAL a = 12.3456>
<SET_LOCAL b = 3>
<SET_LOCAL c = a * b>
<INSERT_LOCAL c>
```

...will cause the value 37.0368 to be inserted into the page. See the INSERT variable commands, below, for more information about controlling the number of decimal places displayed.

You can include multiple operators in the same SET command, but be aware that NetCloak does not honor any operator precedence rules – it merely works from left to right. Thus, all of the following are allowed by NetCloak:

```
<SET_LOCAL newURL = "http://" & gHostName & "/" &
<INSERT_THISURL>>
```

```
<SET_LOCAL TwoTwentyTwo = 200 + 20 + Two>
```

```
<SET_LOCAL WhoKnows = 200 - 20 / 2>
```

```
<SET_LOCAL NotOne = 1.0/2.0 + 1.0/2.0>
```

...but only the first two will produce results you might expect. The third command assigns a value of 90 to the variable WhoKnows; the last assigns NotOne a value of 0.8.

It is standard practice to put strings in quotes and leave numeric values without, but NetCloak will accept a number in quotes or a string without. Also, the equal sign is not required, though is encouraged for readability, especially when performing numeric operations.

DYNAMIC PAGE COMMANDS: VARIABLES

When assigning a variable, you can reference any local, user, global, counter, or cookie variable on the right-hand side of the equal sign, *without* using a nested INSERT command. So, you can set a local variable to the value currently held in a global, or a global variable to the current value of a counter.

For example, let's say you have a counter on your home page called HomeHits, and another counter on every page of your server called TotalHits. You could display the percentage of hits your home page receives on your site with something like:

```
<SET_LOCAL HHTemp = HomeHits * 100>
<SET_LOCAL Percentage = HHTemp / TotalHits>
Home Page: <INSERT_LOCAL Percentage>%
```

To determine the value of a variable found on the right-hand side of the equal sign in a SET command, NetCloak searches for variables in order from most-specific scope to least-specific scope. In other words, it looks for variables that are most-specific to the current page first. It searches for variables in this order:

- Local variables
- User (form) variables
- Counters
- Cookies
- Global variables

Thus, if you have defined a local variable and global variable with the same name, the value of the local variable will be used first. If NetCloak does not find any matching variable name, the variable name itself is used as the value of that parameter.

```
<INSERT_LOCAL varName decimalPlaces>
<INSERT_VARIABLE varName decimalPlaces>
<INSERT_GLOBAL varName decimalPlaces>
```

These commands will place the contents of the variable into the page. The value inserted may be a computed variable

DYNAMIC PAGE COMMANDS: VARIABLES

created using a SET command, or it could be the field from a user-entered form.

For example, after the form above has been filled out and submitted, you could use the following command to display the user's name:

```
Your name is: <INSERT_VARIABLE YourName>
```

Inserted variables will appear as you would expect, with the value of the variable inserted at the point at which the INSERT command is placed. Numeric values will be displayed as numbers (not an ASCII representation of the value) so that a variable with a value of 65 will be displayed as 65 and not the letter A.

If the variable value is a number, you can control the number of decimal places displayed by providing the optional decimalPlaces parameter. If omitted, all decimal places from the value will be displayed. This parameter is ignored if the variable's value contains any characters other than a digit, period, plus, or minus character. For instance, the following code:

```
<SET_LOCAL a = 12.3456>  
<SET_LOCAL b = 3>  
<SET_LOCAL c = a * b>  
<INSERT_LOCAL c 1>
```

...will insert the value 37.0 into the page.

Important Note! It is possible for NetCloak to incorrectly identify users and confuse the variables. This can occur when browsers don't support cookies (or are configured not to allow them) and/or when user IP addresses are being dynamically assigned or users connect through a proxy server. For this reason, you should not use User variables to show and hide secure information, or for mission critical applications. In cases where you are certain that each user will have a unique IP address (for example, possibly on your internal LAN) User variables will be reliable.

Also note that expressions used to create values for variables (+, -, etc.) cannot be used in INSERT commands. To display

DYNAMIC PAGE COMMANDS: VARIABLES

a computed value, you must SET a variable first then reference it by name in the INSERT command.

```
<HIDE_LOCAL varName operator value1 value2...>  
<HIDE_VARIABLE varName operator value1 value2...>  
<HIDE_GLOBAL varName operator value1 value2...>
```

Any HTML text that follows HIDE_LOCAL, HIDE_VARIABLE, or HIDE_GLOBAL will be hidden from those users when the specified variable satisfies the comparison to any one of the listed values. If the comparison operator is not specified, text is hidden if the variable's value contains any of the listed values.

An example should make this more clear. Assuming that the input form above has been filled out and submitted by the user, we could use:

```
<SHOW>  
<HIDE_VARIABLE YourName CONTAINS "J">  
Your name does not contain a J.  
<SHOW>
```

Here, the variable YourName (what the user entered into the field YourName on the input form) is checked to see if it contains a J. If it does, text will be hidden, and the phrase "Your name does not contain a J" will not be seen. To see if the checkbox was checked, we could use:

```
<SHOW>  
<HIDE_VARIABLE MyCheckBox IS "CHECKED">  
You didn't check the check box!
```

Frequently, it is necessary to check that a variable value is not empty, as when a user is filling out form fields. You can check for non-empty variables using the EXISTS operator, or by simply not specifying a value in the HIDE command. For example:

```
<SHOW>  
<HIDE_VARIABLE EXISTS YourName>  
You didn't enter a name!  
<SHOW>
```

DYNAMIC PAGE COMMANDS: VARIABLES

The text “You didn’t enter a name!” will only appear if the YourName field was left blank in the form.

```
<SHOW_LOCAL varName operator value1 value2... >  
<SHOW_VARIABLE varName operator value1 value2... >  
<SHOW_GLOBAL varName operator value1 value2... >
```

This command operates exactly like HIDE_VARIABLE except that it turns the text stream on instead of turning it off. To continue the example from above, we could also show a message when the checkbox is checked, by using:

```
<HIDE>  
<SHOW_VARIABLE MyCheckBox IS "CHECKED">  
You checked the check box!  
<SHOW>
```

<VARIABLE_LIST> </VARIABLE_LIST>

This is a handy utility command which displays a list of all the defined user variables for the current page. The list of variables is formatted according to the HTML text inserted between the opening and closing VARIABLE_LIST tags. The HTML text is repeated once for each defined variable. Within this HTML text, any occurrence of the special tag VARNAME is replaced with the variable name, and the special tag VARVALUE is replaced with the value of that variable. An example will make things clearer. Suppose a user has submitted a form with three fields: a field named Crust with a value of Deep Dish, a field named Topping with a value of Pepperoni, and a field named Size with a value of Medium. Then the following code:

```
<VARIABLE_LIST>  
<B><VARNAME>:</B> <VARVALUE>  
</VARIABLE_LIST>
```

...will cause the following text to be displayed in the browser:

```
<B>Crust:</B> Deep Dish  
<B>Topping:</B> Pepperoni  
<B>Size:</B> Medium
```

DYNAMIC PAGE COMMANDS: VARIABLES

This command is very handy for debugging purposes, as well as for passing variables from one form to another when the variable names may not be known in advance.

Commands: Cookies

Cookies, like other kinds of variables, are made up of two parts: a name and a value. They are transferred in the HTTP header of both user requests and server responses, and are stored by the browser on the user's computer.

When the user asks for a page, the server is allowed to send a cookie in the header (it won't be seen in the HTML page) along with the response page it would normally send. The browser is expected to store the cookie so that when it makes subsequent requests of the server it can send the cookie name and value.

NetCloak gives you the ability to create cookies, and it even makes them easy. Cookies have lots of uses, and are actually a lot like user variables. In some cases, user variables may be better because they can be tracked even when a browser doesn't support them and the data doesn't need to be constantly resent by the browser. On the other hand, cookies have the capability of sharing information among multiple servers, or between different add-on tools on the same server. Cookies are also more reliable when users may be coming through proxy servers or have dynamic IP addresses, such as America Online users.

Even if you prefer to use user variables, cookies can be used to help NetCloak track the variables. For users dialing in using shared modem pools or proxy servers, cookies give NetCloak a reliable way to track the user variable data.

<SET_COOKIE cookieName = value expirationDate domain path >

NetCloak supports a default cookie, named NetCloakID. When this cookie is found, it will use the cookie's value as a key to the user variable data, instead of the user's IP address. To create a randomized default cookie, use the command:

DYNAMIC PAGE COMMANDS: COOKIES

<SET_COOKIE>

For example, the command SET_COOKIE on your server's home page will generate a string and send it as a cookie named NetCloakID. If accepted by the browser, all subsequent user variable operations (using either SET_VARIABLE or forms submitted by the user) will store the variable data based on this randomized key. If the browser refuses the cookie (either it doesn't support cookies or the user chooses not to accept it), then NetCloak will store user variables using the user's IP address as the key. If you are using User variables on your site, it is recommended that you include a SET_COOKIE command on your home page to improve tracking of users.

Be sure that if you're tracking user variables with the default cookie, you don't overwrite the cookie's value after it is created, or the link between the variables and the browser will be broken. You can do this by always checking that the cookie exists using a HIDE_COOKIE command before a SET_COOKIE command, like this:

```
<HIDE_COOKIE NetCloakID EXISTS>
<SET_COOKIE>
<SHOW>
```

In addition to the default cookie, you can also specify up to sixteen additional cookies per user, named whatever you like. A cookie name may only include alphabetic characters, no spaces, and is case-sensitive. The value is optional; if you specify a value, then that value will be given to the cookie, and if you don't NetCloak will generate a random value for you.

For example, to create a cookie named BeenThere with a value of DoneThat, use the command:

```
<SET_COOKIE BeenThere = "DoneThat">
```

The optional expirationDate parameter allows you to specify the expiration date for the cookie. The user's Web browser will store the cookie between sessions and return it on subsequent visits to the page until the expiration date is reached. This allows you to track users between visits to your

DYNAMIC PAGE COMMANDS: COOKIES

site. If no expiration date is specified, the cookie will expire when the user quits the browser.

The expiration date is specified either as date@time, in the same format as HIDE_DATE and other commands which accept date parameters, or as an offset to the current date and time. A negative offset can be used to delete a cookie from the browser's "cookie jar" before its expiration date.

For example:

```
<SET_COOKIE myCookie "" "1/1/2000@17:00">
```

Sets cookie myCookie to a random value which expires on January 1, 2000 at 5 pm.

```
<SET_COOKIE myCookie = 701253 +1.5>
```

Sets cookie myCookie to 701253 which expires in 90 minutes.

```
<SET_COOKIE myCookie "" -1>
```

Deletes cookie myCookie from the browser's cookie jar.

The domain and path parameters allow you to specify which servers and which specific URLs within a server will receive the cookie when requested. If no domain or path parameters are given, then the cookie is sent in every request to the server from which the cookie originated. You can redirect the cookie to another host by specifying that host in the domain parameter, or you can supply a partial domain name so that the cookie is sent to all servers in the given domain. You can use the path parameter to limit the cookie so that it is only sent in requests to a given subfolder of your site. Examples:

```
<SET_COOKIE myCookie = "12345" +72 "maxum.com">
```

Send the cookie in all requests to any server in the .maxum.com domain for the next 72 hours.

```
<SET_COOKIE myCookie = "12345" ""  
"secure.sales.com">
```

Only send the cookie to the server secure.sales.com until the browser quits.

DYNAMIC PAGE COMMANDS: COOKIES

```
<SET_COOKIE myCookie = "12345" +72 "www.maxum.com"
"/netcloak/">
```

Only send the cookie to the server www.maxum.com if the URL begins with /netcloak/, for the next 72 hours.

A special note if you have lots of visitors using America Online browsers: certain versions of AOL browsers only support persistent cookies, which means that you must supply an expiration date in the SET_COOKIE command or the cookie will be ignored. So, for best compatibility with AOL users, you should use the following code to track user variables with the NetCloakID cookie:

```
<HIDE_COOKIE NetCloakID EXISTS>
<SET_COOKIE NetCloakID = "" +24>
<SHOW>
```

This code will only create the cookie if it does not already exist, giving it a random value and an expiration date 24 hours from the time it is created.

<INSERT_COOKIE cookieName>

This command will place the contents of the specified cookie into the page. For example, if you have set the BeenThere cookie noted above you can display it with:

Your Cookie Is: <INSERT_COOKIE BeenThere>

Inserted cookies will appear as you would expect, with the value of the cookie inserted at the point at which the INSERT command is placed.

<HIDE_COOKIE cookieName operator value1 value2...>

Any HTML text that follows HIDE_COOKIE will be hidden from those users when the specified cookie satisfies the comparison to the specified text. If the comparison operator is omitted, a contains comparison is performed.

An example is included in the SHOW_COOKIE section below.

DYNAMIC PAGE COMMANDS: COOKIES

<SHOW_COOKIE cookieName operator value1 value2...>

This command operates exactly like HIDE_COOKIE except that it turns the text stream on instead of turning it off.

For example, assume you have a planes page, a trains page, and an automobiles page on your site. Each of these pages sets the cookie LastMainPage to either planes, trains, or autos, accordingly. When the user accesses a form requesting more information, you could use the HTML:

```
<HIDE><SHOW_COOKIE LastMainPage IS "planes">
Airplane information can be received...
<HIDE><SHOW_COOKIE LastMainPage IS "trains">
Train information can be received...
<HIDE><SHOW_COOKIE LastMainPage IS "autos">
Car information can be received...
<SHOW>
```

Here, the cookie LastMainPage will always be set to the value defined by the most recently accessed of the three pages.

A simple example of a cookie might be to change the behavior of your server after the user has visited a page or taken some action. For example, you might have a form that asks people which sports car is best... the Porsche Boxter, the Mazda Miata, the BMW Z3, or the Honda DelSol VTEC. You could gather the results using global variables, use local variables to display the relative percentages of each car, and use a cookie to remember that the user has already voted and disallow them from voting again.

Commands: Utility

<INSERT_BEEP>

The INSERT_BEEP command causes the server to play the default system alert sound whenever the command is encountered. Nothing is inserted into the text of the page being served. The only effect is a system beep on the server.

At first glance, this command may not seem to be useful, but there are several cases in which it can be handy. You can use

DYNAMIC PAGE COMMANDS: UTILITY

INSERT_BEEP while you are developing pages with NetCloak commands to verify that NetCloak is processing the page.

Another possible use for this command is in your server's error page. By including the command INSERT_BEEP on the error page, your server will beep any time an incorrect URL is processed by your server. This can give you a warning of broken links on your site as they are being accessed.

<REDIRECT url>

The REDIRECT command is a little different from most other NetCloak commands. When this command is encountered in your cloaked HTML document, all processing of the page will immediately cease, and a redirect header will be sent to the browser. This causes the browser to immediately transfer to the URL specified.

Note that a REDIRECT command causes all other contents of the page to be ignored. The redirect will not cause the redirected page to appear within the text of the original page.

The command will not cause a redirect to occur if it is hidden by another NetCloak command, so redirects can be performed conditionally based on any SHOW and HIDE commands. The redirect command can be used in the server's error file to automatically send clients accessing bad URLs to the correct pages. You might also use the command to redirect an HTML page to a page that might be in a completely different format depending on the client's browser type, address or other information.

Here is an example of using REDIRECT to send a special internal use only page based on the user's TCP/IP address. In the example, users in the domain 192.1. will be sent to the page Protected.html. All other users will receive the public page.

DYNAMIC PAGE COMMANDS: UTILITY

```
<HIDE>
<SHOW_DOMAIN 192.1.>
<REDIRECT "http://my.server/Protected.html">
<SHOW>
<HTML>
<TITLE>Public Home Page</TITLE>
<BODY>
Welcome to our public Home Page!
</BODY>
</HTML>
```

The normal REDIRECT command returns a Moved Temporarily (302) HTTP response code to the requesting browser. The REDIRECT command can also return a Moved Permanently (301) response, by using REDIRECT* instead of REDIRECT, like this:

```
<REDIRECT* "http://my.server/Protected.html">
```

For permanently-moved files, the 301 response notifies proxy servers, search engines and other automated or caching systems that the old URL should be replaced permanently.

<REPEAT var operator value1 value2 ... >
</REPEAT >

The REPEAT command is a very powerful command that allows you to insert a chunk of HTML text into a web page a number of times, based on the value of any NetCloak global, local, user, counter, or cookie variable. The HTML to be repeated is placed between the opening and the closing REPEAT tags. The number of times the text will be inserted depends on the condition defined in the REPEAT tag. The text is repeated until the condition is false. If the condition is always false, then no text is inserted.

Any comparison operator supported by the SHOW and HIDE commands may also be used in the REPEAT command. If the comparison operator is omitted, the REPEAT command will loop as long as the specified variable exists.

It is up to you to make sure an infinite loop is not created. When an infinite loop occurs, NetCloak continues to add the text to the returned page indefinitely and will make the server freeze or crash. In order to avoid an infinite loop, you **must**

DYNAMIC PAGE COMMANDS: UTILITY

update some loop variable between the opening and closing REPEAT tags until the condition in the REPEAT tag eventually becomes false.

Ordinarily, you would use SET_LOCAL commands to increment or decrement a local variable inside the REPEAT tags. For example, the following code will create a simple 2-row by 3-column table:

```
<SET_LOCAL row = 1>
<SET_LOCAL col = 1>
<TABLE>
<REPEAT row LT= 2>
  <TR>
    <REPEAT col LT= 3>
      <TD>Row <INSERT_LOCAL row> Col <INSERT_LOCAL
col></TD>
      <SET_LOCAL col = col + 1>
    </REPEAT>
  </TR>
  <SET_LOCAL row = row + 1>
</REPEAT>
</TABLE>
```

Here is the HTML text that is actually sent to the browser:

```
<TABLE>
<TR>
  <TD>Row 1 Col 1</TD>
  <TD>Row 1 Col 2</TD>
  <TD>Row 1 Col 3</TD>
</TR>
<TR>
  <TD>Row 2 Col 1</TD>
  <TD>Row 2 Col 2</TD>
  <TD>Row 2 Col 3</TD>
</TR>
</TABLE>
```

The variable in the REPEAT command may also be a multivalue form variable (such as a group of checkboxes or a multiple select list box). In that case, you use a local variable to access each instance of the multivalue variable. For instance, let's suppose a user has filled out a form which contains a set of checkboxes to pick pizza toppings. You want to display the selected toppings back to the user so they can confirm their order. The checkboxes are all named Topping, and each checkbox in the form has a different value, like this:

DYNAMIC PAGE COMMANDS: UTILITY

```
<INPUT TYPE="checkbox" NAME="Topping"
VALUE="sausage">
<INPUT TYPE="checkbox" NAME="Topping"
VALUE="pepperoni">
<INPUT TYPE="checkbox" NAME="Topping"
VALUE="mushrooms">
```

If the user chooses more than one topping, then multiple instances of the variable Topping will be created by NetCloak. Each instance can be accessed using square brackets like this: Topping [1], Topping[2], etc.

To loop through all the selected toppings, you can use a local variable between the square brackets, and repeat as long as an instance of the multivalue variable exists. So here is an excerpt from the order confirmation page:

```
<P>Selected Toppings:
<SET_LOCAL index = 1>
<REPEAT Topping[index] EXISTS>
<INSERT_VARIABLE Topping[index]>
<SET_LOCAL index = index + 1>
</REPEAT>
```

If the user checks the pepperoni and mushrooms checkboxes in the form, the confirmation page they get looks like this:

```
<P>Selected Toppings: pepperoni mushrooms
```

<EXEC_CGI CGIpath scriptName beginText endText>

This command allows you to dynamically execute CGIs and insert their response into a page, as well as execute multiple CGIs from a single client request. These can be either other commercial CGIs, or CGIs that you write yourself. For more information on how NetCloak calls other CGIs, see the section “Using NetCloak With Other CGIs” in the User’s Guide.

The EXEC_CGI command accepts up to four parameters, though the second, third and fourth are optional. The CGIpath parameter is simply the path to the CGI application you would like to execute. The path must be the path to the actual application, not an alias or a document created by or for the application that will be run.

DYNAMIC PAGE COMMANDS: UTILITY

The `scriptName` parameter allows you to overwrite the `direct` parameter sent to the CGI. Leaving this parameter blank will simply pass along the `direct` parameter as it originally came from the server, but overwriting this is an important capability. By changing the parameter, you can make the CGI think that the requested URL was different than it really was. This parameter also replaces the Script Name field of the CGI AppleEvent, so that CGIs that function as actions will work properly.

The `filepath` and `scriptName` parameters may be either relative to the Root Folder, or relative to the location of the page being served. They may be specified using either the URL style for file paths (using slash '/' characters to delimit folder names), or the MacOS style (using colons ':' to delimit folder names). If the first character of the file path is a colon or a slash, the file path is assumed to be relative to the Root Folder. Otherwise, it is assumed to be relative to the location of the page being served. (See the `INSERT_MODIFIED` command for more information about `filepath` parameters.)

Maxum has produced a simple, freeware CGI named Responder which is useful for testing or demonstrating the `EXEC_CGI` command. Responder simply echoes the CGI request that is sent to it as an HTML page listing the various parameters separately. Here is a snippet that will include Responder's output in an HTML page served by NetCloak:

```
<EXEC_CGI "Responder.cgi">
```

You can download Responder from the Maxum Web site at <http://www.maxum.com/>.

In many cases, you may want only a portion of the CGI response. In the case of Responder, you might want only the BODY of the complete HTML page that it returns. You can filter out what you don't want by specifying a beginning and an ending phrase. Only text that appears between the phrases will then be inserted into the response page. Notice that you must specify the `scriptName` parameter (even if it is left blank) in order to use the filtering parameters. For example:

```
<EXEC_CGI "Responder.cgi" "" "<BODY>" "</BODY>">
```

DYNAMIC PAGE COMMANDS: UTILITY

This example would include only the parts of Responder's response that are between, but not including, the `<BODY>` and `</BODY>` tags.

Note that the `EXEC_CGI` command does not allow setting query or post arguments. (Query arguments are included in a CGI request URL preceded by a question mark; post arguments are sent as data from an HTML form.) `EXEC_CGI` therefore cannot be used with CGIs that depend on these for input. This typically includes database access CGIs, as well as search tools such as Phantom. If you have a CGI that depends on query or post arguments, consider running it as a back-end to NetCloak instead.

`<EXEC_PIXO serviceName scriptName startText endText>`

This command is only available in the NetCloak plug-in. It operates identically to the `EXEC_CGI` command, but instead of calling an external CGI application via AppleEvents, it calls another WebSTAR API plug-in running on the same web server as NetCloak, using the PIXO (Plug-In Cross Over) standard for inter-plug-in communication jointly developed by Maxum and Clearway Technologies, Inc. Most commercial WebSTAR plug-ins available today support PIXO.

In this case, the first parameter, `serviceName`, is the name of the PIXO service registered by the plug-in you want to call. The PIXO standard requires plug-ins to register a service name which provides other plug-ins access to their functionality. The convention for naming a PIXO service is to append the text `_PIXO` to the plug-in's normal action name. For instance, NetCloak's PIXO service name for processing HTML pages is `CLOAK_PI_PIXO`, and the PIXO service name used by NetCloak Pro to process FDML files is `NFORMS_PI_PIXO`.

Included in the NetCloak package (in the Goodies folder) is a plug-in called FireSite SSI `_PIXO` Adapter which allows you to call the WebSTAR SSI plug-in from cloaked pages. When this plug-in, WebSTAR SSI, and NetCloak are all installed, you can

DYNAMIC PAGE COMMANDS: UTILITY

insert the contents of an SSI-processed page into a NetCloak-processed page like this:

```
<EXEC_PIXO "SSI_PIXO" "ssipage.shtml" "<BODY>"  
"</BODY>">
```

The above command will call WebSTAR SSI to process the file `ssipage.shtml`, and then insert the text between the `<BODY>` and `</BODY>` tags into the current page.

<EXEC_FDML FDMLPath beginText endText>

Using this command in any HTML page, NetCloak can trigger the processing of an FDML file whenever a Web page is accessed by a browser. Multiple instances of this command in a cloaked page can also be used to perform multiple tasks from a single form submission, without using CHAIN directives.

Where this command appears in the page, NetCloak will insert the response page returned from the FDML file.

The `FDMLPath` parameter is a filepath to an FDML file. This parameter conforms to the filepath rules described in the NetCloak Standard User's Guide; if the path begins with a slash or colon character, the path is appended to the Root Folder to locate the file. Otherwise, the path is appended to the path of the folder in which the page containing the `EXEC_FDML` command resides.

The optional `beginText` and `endText` parameters are used to extract only a portion of the response page to be inserted into the HTML page. If omitted, NetCloak will use `<BODY>` and `</BODY>` so that only the content of the response page body is inserted into the cloaked HTML page.

Here are some examples:

```
<EXEC_FDML "SendMsg.fdm1">
```

This command processes the FDML file `SendMsg.fdm1`, located in the same folder as the current page, and inserts the body of the resulting response page into the current page.

DYNAMIC PAGE COMMANDS: UTILITY

```
<EXEC_FDML "/forms/updaterpt.fdm1">  
<EXEC_FDML "/forms/log.fdm1" "</HEAD>" "<BODY>">
```

This example causes two FDML files to be executed; the files named `updaterpt.fdm1` and `log.fdm1`, each located in the `forms` folder in the web server Root Folder. The response from processing `updaterpt.fdm1` will be inserted into the HTML page. Because no text usually occurs between the `</HEAD>` and `<BODY>` tags in an HTML page, no text from the response of the second FDML file will be inserted into the final page.

<MACRO macroName>

The Macro command is used to insert text or HTML from a local file or pre-defined macros. If the specified macro name matches the name of a macro defined in the file `NetCloak.macros`, NetCloak will insert the contents of that macro into the HTML in place of the MACRO command. If the macro name is a pathname for a file, NetCloak will insert the contents of the specified file.

When inserting a file, the `macroName` parameter may be either relative to the Root Folder, or relative to the location of the page being served. It may be specified using either the URL style for file paths (using slash `'/'` characters to delimit folder names), or the MacOS style (using colons `':'` to delimit folder names). If the first character of the file path is a colon or a slash, the file path is assumed to be relative to the Root Folder. Otherwise, it is assumed to be relative to the location of the page being served. (See the `INSERT_MODIFIED` command for more information about filepath parameters.)

When serving multiple virtual domains, macro file paths are resolved relative to the root folder for each virtual domain. This has two implications. First, no macro may access any file outside its virtual domain's root folder, unless you create aliases within the virtual domain's root folder. Second, this lets you provide macros that are customized for each virtual domain.

DYNAMIC PAGE COMMANDS: UTILITY

If NetCloak is unable to insert a specified macro, it will insert an error message enclosed in an HTML comment. View the HTML source of the document after it is served to a Web browser to see the error message.

If the specified macro contains NetCloak SHOW or HIDE commands, the show/hide state (or “mode”) of text following the MACRO command is **not** affected by the final mode of the macro. In other words, the mode of a MACRO command is **not** propagated back to the calling page.

When processing a page, NetCloak also does not process MACRO commands that are hidden by a previous HIDE command.

A trivial example will help explain this. Consider the following macro, defined in the NetCloak.macros file:

```
<BEGIN_MACRO TurnOn>
<SHOW>
<END_MACRO>
```

This macro does nothing when used in a cloaked page:

```
<HIDE>
This text is hidden from all users all the time.
<MACRO TurnOn>
This text is ALSO always hidden! In fact, NetCloak
won't even call the macro.
<SHOW>
```

If you want a macro to affect the mode of the calling page, use the MODEMACRO command (see below) instead.

< MODEMACRO macroName >

MODEMACRO behaves identically to the MACRO command, except for two important differences:

- The show/hide mode of text following a MODEMACRO command matches the final show/hide mode inside the called macro.
- MODEMACRO commands cannot be hidden by a HIDE command. They are *always* processed where ever they occur in a cloaked page.

DYNAMIC PAGE COMMANDS: UTILITY

Using the same example TurnOn macro defined above, the following code works as expected:

```
<HIDE>
This text is hidden from all users all the time.
<MODEMACRO TurnOn>
This text is shown to all users.
```

<BEGIN_MACRO>

The BEGIN_MACRO command is used in the NetCloak.macros file to define the start of a macro. Outside of the NetCloak.macros file it will have no effect.

<END_MACRO>

The END_MACRO command is used in the NetCloak.macros file to define the end of a macro. Outside of the NetCloak.macros file it will have no effect.

See the “Macros” section in the User’s Guide for more information on defining and using macros.

HTML FORM TAGS

The first step in allowing users to publish articles on your web server is to create an HTML form in which authors will enter information. Creating HTML forms that allow user input is relatively straightforward and the process is well documented in many places on the Web. Here is a quick reference for the most common form tags and their attributes.

The FORM Tag

An HTML form is defined by a beginning `<FORM>` tag and an ending `</FORM>` tag. In between these form tags are various `INPUT` tags which define the look and behavior of the input fields in the form.

Every `FORM` tag used with NetCloak must have at least two attributes:

1. An `ACTION` attribute which tells the web server that NetCloak is to process the input data.
2. A `METHOD` attribute which describes how the input data is sent from the browser to the server.

The ACTION Attribute

It is the `ACTION` attribute that tells the browser (which tells the server) how to process the data from the form when it is submitted. This is actually a URL, so it works similarly to an `HREF` tag. NetCloak Pro automatically registers the suffix `.fdml` with the web server, so all you need to do is specify the NetCloak FDML template document to use to process the form data. For example:

```
<FORM ACTION="MyFDML.fdm1" METHOD=POST>
```

In this case, the `ACTION` attribute of the `FORM` tag tells the server to use the FDML template document `MyFDML.fdm1` (located in the same directory with the form) to process the data.

Just like a hyperlink in an `HREF` tag, the `ACTION` can be specified with a relative path when the desired FDML is

located on the same server as the form being accessed. In this case, either a relative or an absolute path may be used.

The form and FDML need not be located on the same server, in which case an absolute URL (including `http://` and the server hostname) must be used. (There is a security option that restricts this by default— see the Configuring NetCloak section.) In either case the path to the FDML file is always relative to the location of the web server or site root folder. The following example uses the same syntax as above but with an absolute path, which would be used if the form were located on another web server:

```
<FORM ACTION="http://www.myserver.com/MyFDML.fdm1"  
METHOD=POST>
```

You can also set up other file name suffixes for FDML templates; these need not be `.fdml`. This requires setting up an additional suffix mapping in your web server and checking the NetCloak security option. See the Configuring NetCloak section of the User's Guide.

The METHOD Attribute

There are two methods of sending form data from a Web browser to a server: GET and POST. NetCloak supports either method.

The GET method encodes the form data and appends it to the URL specified in the ACTION attribute as search arguments. Search arguments follow a question mark (?) character in the URL received by the server. Because URLs are limited to a length of 1024 characters, including the protocol, server name, file path, and search arguments, this limits the length of data that can be sent using the GET method. Ordinarily, HTML forms use the POST method.

Using the POST method, the encoded form data is sent in the content of the message sent from the browser to the server. All this means is that there is no limit to the length of data you can send using the POST method. However, NetCloak has a limit of 32Kb (32,768 characters) of data that can be received using the POST method.

HTML FORM TAGS

In either case, you must specify the method you want to use in the METHOD attribute of the FORM tag, as in these examples:

```
<FORM ACTION="/MyFDML.fdm1" METHOD=GET>
```

```
<FORM ACTION="/MyFDML.fdm1" METHOD=POST>
```

Form Fields

To allow users to enter data into your form, you will also need some entry fields. There are various types of entry fields, most of which correspond to user-interface elements you are already familiar with from the dialog boxes on your computer.

Every form field tag must include a name attribute. Later, when creating FDML documents, fields must be referred to by name. Other attributes are particular to each field type, such as type, value or size. For most field types the type attribute is required, but others may be optional.

Text Box

A Text Box allows the user to enter a single line of text into the form. The optional SIZE attribute can be used to set the length, in characters, of the box that will be displayed; MAXLENGTH is the maximum number of characters that can be entered. The VALUE attribute can optionally be included to set a default value for the field which will appear without the user having entered anything.

```
<INPUT TYPE="text" NAME="myText" SIZE="30"  
MAXLENGTH="30" VALUE="default value">
```

Password Box

A password box is identical to a text box, except that the text entered by the user will appear as bullet characters (•). These are most commonly used for entering passwords or other sensitive information you wish to be obscured as it is entered. The password type allows the same options as the TEXT type.

```
<INPUT TYPE="password" NAME="mySecret">
```

Text Area

A text area field is similar to a text box, but can have multiple rows with scroll bars. The optional rows and cols attributes set the size of the text area in rows and columns. The optional wrap attribute allows you to specify automatic text wrapping. A value of hard or physical will automatically insert return characters when the end of a line is reached; a value of soft will wrap text at the end of the line without inserting return characters; and a value of none will not wrap text automatically. Default text for the text area may be specified by placing the default text in between the opening and closing TEXTAREA tags.

```
<TEXTAREA NAME="myText" ROWS=5 COLS=40>Default  
Text</TEXTAREA>
```

Checkbox

Displays a checkbox on the form. Checkboxes are normally used singly to allow the user to make a yes/no choice, or in a group to allow the user to select one or more of several options.

The required value attribute will be returned as the value of the field if the box is checked, otherwise nothing is returned. If the checked attribute is specified, the box will be checked by default.

```
<INPUT TYPE="checkbox" NAME="myCheck" CHECKED  
VALUE="box1">
```

Radio Buttons

```
<INPUT TYPE="radio" NAME="myButtons" CHECKED  
VALUE="button1">  
<INPUT TYPE="radio" NAME="myButtons"  
VALUE="button2">
```

Displays a radio button into your form. Radio buttons are so named because they operate like the mechanical push buttons on old car radios—pushing one button in selects a

HTML FORM TAGS

particular station, and pushing a different button pops out the previous one. Radio buttons are normally used in a group to allow the user to select one and only one of several options.

The required value attribute will be returned as the value of the field if that button is selected. If none of the buttons are selected, nothing will be returned. To link several radio buttons, assign the same name but different values. If the checked attribute is specified, that button will be selected by default before the user has selected anything.

Pop-up Menu

```
<SELECT NAME="myPopup">
<OPTION VALUE="1" SELECTED>first
<OPTION VALUE="2">second
<OPTION VALUE="3">third
</SELECT>
```

Inserts a pop-up selection menu element into your form. Pop-up menus allow the user to select one and only one of several options, listing each option as text. Popup menus function similarly to radio buttons, but appear quite different.

An option which includes selected in the option tag will be selected by default. A value attribute may be specified inside each option tag. If a value is specified, the value will be returned when that field is selected. Otherwise, the text listed after the option tag will be returned when that option is selected.

Scrolling List

```
<SELECT MULTIPLE SIZE=3 NAME="myList">
<OPTION VALUE="1" SELECTED>first
<OPTION VALUE="2">second
<OPTION VALUE="3">third
<OPTION VALUE="4">fourth
</SELECT>
```

Adds a scrolling selection list to the form. A scrolling list has a vertical scroll bar, and lists each option allowing the user to scroll through and select one or more items. Just as a pop-up

menu is analogous to a group of radio buttons, a scrolling list is analogous to a group of check boxes.

The optional size attribute sets the number of items that will be visible at the same time. In most browsers, the scroll bar will not be displayed correctly unless there are at least four items displayed. An option which includes selected in the option tag will be selected by default. A value attribute may also be specified inside each option tag to insert a value other than the text listed after the option tag.

Hidden Field

```
<INPUT TYPE="hidden" NAME="myInvisible" VALUE="you  
can't see this!">
```

Inserts a hidden field into your form. Hidden fields are submitted as part of the form, but the field and the information in it are not displayed by the browser.

Hidden fields are useful for including information in a form that you don't want the user to see, such as the filename from a previous form, or dynamic information inserted by NetCloak. Note that hidden fields *are* displayed to the user if they use the View Source command in the browser, so they should not be used for security or to hide sensitive information.

Submit Button

```
<INPUT TYPE="submit" NAME="Submit Form">
```

When the user is done entering information or making selections, they will need a button on the form to click to submit the form for processing. The optional name attribute can be used to specify the text on the button.

FORMS PROCESSING COMMANDS

After you have created an HTML input form, you will need to create an **FDML** document. FDML stands for Forms Definition Markup Language and can be thought of as a language similar to HTML that understood only by NetCloak Pro. FDML documents are referred to as “templates”, and contain tags that tell NetCloak how to format the data entered by an author in the input form as well as direct it how to act on the data once it is formatted.

There are three types of NetCloak Pro tags that can be used in FDML files:

- Primary Directives
- Supplemental Directives
- Insertion Commands

It is important to note here that forms processing commands are only supported in the Pro version of NetCloak.

Primary Directives

The commands provided by FDML are used to execute the four **Primary Directives** of NetCloak. There must be always exactly one Primary Directive specified in any FDML document.

The four Primary Directives of NetCloak are designed to perform the following tasks:

- Create new HTML pages. (CREATEDOC)
- Update existing HTML pages. (INSERTFILE)
- Store tab-delimited text data. (TEXTSTORE)
- Send an e-mail message. (SENDMAIL)

The following sections describe these four capabilities, and you are encouraged to refer to the tutorial examples in the User’s Guide to see each of the Primary Directives in action.

FORMS PROCESSING COMMANDS

<CREATEDOC> Creating New HTML Pages

The CREATEDOC directive tells NetCloak to build a new HTML page and save it as a file specified by the Filename parameter, as in the example below. The Filename parameter can be specified as a path relative to the web server root folder or relative to the FDML document. The proper syntax for the CREATEDOC directive is:

```
<CREATEDOC>"Filename"</CREATEDOC>  
New document template...
```

Frequently the directive will contain other FDML insertion commands (like REPLACE, described in the Commands section) to build a dynamic filename so that documents are uniquely named on your server and are not overwritten each time a new user fills out your form.

When any of the insertion commands are used in a filename, you should use the REPLACE_FN (FN stands for File Name) command. This ensures that characters which are not allowed in file or folder names are removed before new files are created. See the REPLACE command for more information.

When the filename you specify contains Finder aliases to folders, the aliases are resolved automatically. Also, if any folders in the path do not exist, they are created automatically. If you use the YEAR or MONTH insertion commands in the Filename parameter, for example, you can have NetCloak automatically create a new folder for every year and every month, like this:

```
<CREATEDOC>"Reports/<YEAR>/<MONTH_FN>/MonthlyReport.html"</CREATEDOC>
```

What follows the closing CREATEDOC in the FDML file is text and/or HTML that serves as a template for the document to be created. This template defines the format of the new document, including the standard HTML, header, and body tags (<HTML> ... </HTML>, <HEAD> ... </HEAD>, and <BODY> ... </BODY>). Throughout the document template, you may use FDML insertion commands to dynamically insert form data into the created document.

FORMS PROCESSING COMMANDS

Note: FDML syntax is similar to HTML, but is different enough to cause problems when using WYSIWYG HTML editing applications like Adobe GoLive or Claris HomePage. If you plan to use such a program to create your document templates, Maxum recommends creating the template using the HTML editor first, until you are satisfied with the page layout. Then open the HTML file using a text editor like SimpleText or BBEdit, and add the FDML directives and insertion commands afterward.

<INSERTFILE> Updating Existing Pages

The INSERTFILE directive allows you to insert text into an existing HTML page instead of creating an entirely new page. The syntax is shown below:

```
<INSERTFILE "MarkerText">"Filename"</INSERTFILE>  
Text to insert...
```

There are two parameters used with the INSERTFILE directive.

The first, located within the opening INSERTFILE tag, is the text which marks the location within the file where the new text should be inserted. While any text may be used to mark the insertion point in the file, an HTML comment is most often used, so that the insertion point text does not appear in the page.

The second parameter is simply the path and name of the file to be updated and appears between the opening and closing tags.

The syntax of an HTML comment is shown in the following example.

```
<BODY>  
<H1>This is a test document</H1>  
<!--EnterText-->  
</BODY>
```

To use the INSERTFILE directive to update this file, simply specify the HTML comment followed by the file path, like this, in your FDML file. For our example, the file is named

FORMS PROCESSING COMMANDS

testdoc.html, and is located in a folder named docs located in the root folder of the web server:

```
<INSERTFILE "<!--EnterText-->">
"/docs/testdoc.html"
</INSERTFILE>
```

Following the closing INSERTFILE tag is the text or HTML to be inserted into the document. As with the CREATEDOC directive, you may use any standard HTML tag and any FDML insertion command in this text to customize it based on the form input data. Since this text will be inserted into an existing HTML document, however, you should not include the HTML, header, and body tags.

Using the basic INSERTFILE directive, new text will be added after the comment, which will cause the new entries to be added to the top of the list of entries, creating a newest first listing. Use the variation INSERTFILE* to have new entries added before the comment, which will cause the list of entries to grow downward, in a newest last format.

<TEXTSTORE> Storing Data

The TEXTSTORE directive will store data submitted from a form in a tab-delimited text file without any other formatting. Data from subsequent form submissions are added as the next line of the file.

There will probably be times when you want to create a form to receive user input and use that information in another application such as a spreadsheet or a database application. You might also want to store data in a simple format for archival or backup purposes, and go on to process the form with other FDML documents.

In either case, the NetCloak directive TEXTSTORE will handle this for you by creating and maintaining a tab-delimited text file. This file can then be imported into the application or database of your choice whenever you need it. What you do with the file is up to you, but virtually all word processing, database, and spreadsheet applications will read or import it.

FORMS PROCESSING COMMANDS

Most personal information managers and other types of software will read the information as well.

The syntax for the TEXTSTORE directive is as follows.

```
<TEXTSTORE>"TextFileName"</TEXTSTORE>  
Record format...
```

The *TextFileName* parameter is simply the name of the file that NetCloak will create and maintain. As with any directive, you may use FDML insertion commands within the parameter to allow files to be created and maintained based on user input. Note that since the created file is not an HTML file, it should *not* have the .html file suffix. Instead, it should have a suffix of .text or .txt, to denote that it is a text file.

Following the TEXTSTORE directive in the FDML file is the list of fields you would like stored in each record in the text file. Records are separated in the text file by a single carriage return character, and fields within a record are separated by a tab character.

In the FDML file, each field name is placed by itself on a line of the FDML file in the order they are to appear in the file. The data specified on each line of the FDML will be separated by a tab and all of the lines of the FDML will be combined into a single line in the text file.

For example, if you have a form with fields for Name and E-mail, the syntax for a very simple FDML document to store the data in a file would be:

```
<TEXTSTORE>"Signup.text"</TEXTSTORE>  
Name  
E-mail
```

This would result in a text file named Signup.text (located in the same folder as the FDML file) which might contain the following text after a few users had used the form page:

Chris Patterson	chris@maxum.com
John O'Fallon	john@maxum.com
Steve Jobs	steve@apple.com

You can also specify text constants to be stored in the text file by beginning a line of the FDML file with a plus sign (+).

FORMS PROCESSING COMMANDS

This tells NetCloak to insert the text exactly as it appears, without attempting to find user-entered data for that field. Inserting constant text by itself isn't terribly useful, but when used in conjunction with other NetCloak commands, the possibilities grow considerably. For example, we can add a time and date stamp to our signup example above, as in the following example:

```
<TEXTSTORE>"Signup.text"</TEXTSTORE>
+Date: <DATE> Time: <TIME>
E-mail
```

Which would result in a text file that looked like this:

```
Date: 12/14/97 Time: 14:25:27 chris@maxum.com
Date: 12/15/97 Time: 08:42:18 john@maxum.com
Date: 12/16/97 Time: 18:02:44 steve@apple.com
```

The E-mail field could also be labeled, as in the following example:

```
<TEXTSTORE>"Signup.text"</TEXTSTORE>
+Date: <DATE> Time: <TIME>
+E-Mail: <REPLACE Email>
```

Resulting in:

```
Date: 12/14/97 Time: 14:25:27      E-Mail: chris@maxum.com
Date: 12/15/97 Time: 08:42:18      E-Mail: john@maxum.com
Date: 12/16/97 Time: 18:02:44      E-Mail: steve@apple.com
```

TEXTSTORE FDML files are completely parsed, so you can use all FDML insertion commands, including IF commands, within TEXTSTORE documents. The IF command can be used effectively within TEXTSTORE documents to selectively insert fields based on the contents of other fields. In addition, you can specify text to be inserted into the data file as-is.

<SENDMAIL> Sending E-mail

NetCloak is capable of sending e-mail messages using Simple Mail Transfer Protocol (SMTP).

When a form is submitted, NetCloak can build an e-mail message and send it to anyone you like. As the author of the HTML entry form, you control which user-entered fields are

FORMS PROCESSING COMMANDS

included in the message and the format of the message. The SENDMAIL syntax is:

```
<SENDMAIL "server" "to" "from" "cc">  
subject  
</SENDMAIL>  
Message to send...
```

Within the SENDMAIL tag, there are four parameters, described below.

- **Server**

This is the address of the SMTP mail server that NetCloak will use to send the message. Normally this is the address of your mail server or your ISP's mail server, not the server of the person receiving the message. While a message for ed@somewhere.net may be sent directly to the mail server somewhere.net, you would normally tell NetCloak to connect to your local mail server. The server that NetCloak gives the message to will then forward the e-mail to the appropriate server for the recipient.

To reduce the chances of failure, you should use the closest (and most stable) e-mail server available to you.

If you aren't sure of the address of your e-mail server, check your Internet e-mail client software configuration or ask your ISP or network administrator.

- **To**

This parameter contains the e-mail addresses of the person or persons to whom the message will be sent. For example, bob@mycompany.com or bob@mail.mycompany.com.

You may specify multiple e-mail recipients in this parameter by separating the e-mail addresses with a comma, like this: "bob@company.com, fred@company.com, jill@company.com".

For each recipient, you may also include the user's real name, like this: "Bob Smith bob@company.com, Fred Jones fred@company.com, Jill Johnson jill@company.com".

FORMS PROCESSING COMMANDS

Finally, for each e-mail address, you may use double quote characters to mark text that is to be ignored by the mail server when parsing the address, such as the real name. But because double quote characters are also used to delimit the to address parameter, you must use two double quotes around the embedded double-quote characters within the SENDMAIL tag. For example:

```
<SENDMAIL ... """"Chris Patterson""  
chris@maxum.com" ... >
```

NetCloak insertion commands can be used to dynamically set the to parameter.

- **From**

This parameter is required, and specifies an address from which the message has been sent. The from address must be a valid e-mail address, otherwise the message may be rejected by the mail server. If there is a problem with the message, most mail servers will first attempt to return it to the sender.

- **CC**

This is an optional parameter, which lists additional email addresses of carbon-copy recipients of the message. As with the to parameter, you may specify multiple carbon-copy recipients, with or without real names, and may use NetCloak insertion commands to dynamically cc recipients.

Immediately following the SENDMAIL tag is the message subject. This is followed by the closing SENDMAIL tag. As with any directive parameter, REPLACE and other insertion commands can be used to dynamically create a subject line.

Anything after the </SENDMAIL> will be sent in the body of the message. A combination of plain text and insertion commands may be used to customize the body of the message from the input form data.

The content type of the mail message is normally plain text, which is the default content type, and is supported by all e-mail client software. However, you can send a message with

FORMS PROCESSING COMMANDS

HTML-formatted content. NetCloak will set the content type of the mail message to text/html if the very first text of the message, other than space characters, is the HTML tag. This allows HTML-formatted mail messages to be displayed properly by e-mail clients, such as Netscape Communicator, that can handle them.

Using NetCloak to send e-mail is very simple and is almost always set up in much the same way, so an example will make things clearer and is always a good place to start. For this example, we have an HTML form that accepts the following fields from the user: Email, Subject, UserName, Phone, and Message. The FDML would look like this:

```
<RESPONSE>"/MailDemo/Response.html"</RESPONSE>
<SENDMAIL
    "mail.yourorg.com"
    "John Smith john@mail.yourorg.com"
    "<REPLACE UserName> <REPLACE Email>">
<REPLACE Subject>
</SENDMAIL>

Name: <REPLACE UserName>
E-Mail Address: <REPLACE Email>
Phone Number: <REPLACE Phone>
Message:
<REPLACE Message>
```

Here we've used a RESPONSE directive prior to the SENDMAIL directive to allow a predetermined HTML page to be returned to a user after the SENDMAIL form is submitted. The SENDMAIL directive then specifies exactly how to send the message, including SMTP host, sender, recipient (obtained from the person filling in the form), and subject (also obtained from the form). In this case, NetCloak will use the server mail.yourorg.com to send a message to John Smith. The message will be from the person that filled out the form, so when John uses his e-mail client to reply to the message, the reply will be automatically sent to the correct address.

Note that using REPLACE commands within the SENDMAIL directive is perfectly legal, and sometimes essential. Finally, following the SENDMAIL directive is the body of the message

FORMS PROCESSING COMMANDS

(which is also obtained from the form that has been completed).

How NetCloak Manages SENDMAIL Messages

When a mail message is created using the SENDMAIL directive, NetCloak stores the information about the message in a uniquely-named text file inside the folder named NetCloak Mail Queue inside the NetCloak Files folder. At regular intervals, it looks for files in this folder, and attempts to send them using the SMTP protocol.

If the message is sent successfully, the mail file is deleted.

If the message cannot be sent, the mail file is updated with a retry count and an error message that explains the reason the message could not be sent. Common reasons that mail messages fail to be sent are:

- missing or invalid from addresses,
- malformed to or cc addresses
- incorrectly specified mail server name
- mail server denies access to unknown from addresses

NetCloak will try to send mail messages up to five times. After the fifth failed attempt, the corresponding mail file is moved from the NetForms Mail Queue folder into another folder named NetForms Failed Mail inside the NetCloak Files folder. This is the electronic equivalent of the dead-letter bin at your post office. Once moved, these files are ignored by NetCloak.

Periodically, the NetCloak administrator should check the contents of the NetForms Failed Mail folder. Some failed mail messages can be easily fixed by editing the mail file (they are simple TEXT files) and re-queuing them by moving them back to the NetForms Mail Queue folder. Other failed mail files may be simply dragged to the Trash to reclaim disk space.

Hot tip: Because NetCloak looks for text files in the NetForms Mail Queue folder to send email messages, *any* application or AppleScript that can create text files can also use NetCloak to send email messages! The text file created

FORMS PROCESSING COMMANDS

need only conform to the format of NetCloak-generated mail files in order for NetCloak to be able to send a message.

Supplemental Directives

There are ten Supplemental Directives and many Insertion Commands available for use in an FDML document. Both of these types of commands are used to augment the functionality of the four Primary Directives of NetCloak.

Supplemental Directives are put at the very beginning of an FDML document, typically before the Primary Directive, and allow you to specify operations to be performed by NetCloak in addition to that specified by the Primary Directive.

Insertion Commands are used to place the contents of fields (entered by the author, or generated by the system) into the text of the finished HTML page, e-mail message, or text file created by NetCloak.

Within each FDML document you may use each Supplemental Directive only once, but Insertion Commands may be used repeatedly.

The NetCloak Supplemental Directives will allow you to:

AUTOLINK	Automatically replace text entered into articles with text you specify
CHAIN	Connect more FDML documents to process data from the same form
COPY	Save a second copy of an article
MENUDOC	Automatically insert a link to a new article or other text into an existing document
RECENTLIST	Add the new article to an internal list of recently submitted articles

FORMS PROCESSING COMMANDS

RESPONSE	Send a customized response file to the user after the form data is processed
VERIFY	Verify that specified fields in the form are not empty
RETURN_PAGE	Send a customized response file and abort form processing (used with the IF command).
EXPIRE	Automatically delete the created document or inserted text, and menu document links, after a specified period of time has passed.
SET_NETCLOAK	Temporarily modify NetCloak configuration settings while processing a form submission.

<AUTOLINK> Replace text entered into articles

`<AUTOLINK>"Filename"</AUTOLINK>`

One of the most popular features of NetCloak is AUTOLINK. The AUTOLINK directive allows you to specify a file that will be used to automatically insert hypertext links into new articles and inserted text. The filename may be specified as a full path from the web server root folder, or as a relative path from the FDML file, depending on your NetCloak configuration.

AUTOLINK is not limited to just building hypertext links. It can also be used to replace any sequence of characters found in a page or e-mail message. This is handy for censoring foul language within submitted text, for instance.

The AutoLink file specified is a simple text file which lists the links to be generated. This is explained in detail in the section "Creating an AutoLink File".

FORMS PROCESSING COMMANDS

Since the AUTOLINK directive can appear in each FDML file, you may specify a different AutoLink file for each form, if you wish.

<CHAIN> Process the same entry with another FDML

```
<CHAIN>"Filename"</CHAIN>
```

The CHAIN directive is used within any FDML document, and allows you to specify additional FDML documents used to process the same input form data. In other words, if you would like for a single form submission to generate two or more results, you can chain two FDML documents together. For example, you may want to have NetCloak create a new Web page, and then have an e-mail message with the text of the new page e-mailed to the Webmaster for content verification.

You may chain as many FDML documents as you wish.

The filename may be specified as a full path from the web server root folder, or as a relative path from the FDML file.

<COPY> Save a second copy of an article

Syntax:

```
<COPY>"Filename"</COPY>
```

Requires: a CREATEDOC directive.

The COPY directive is used to save a second copy of an article file created by the CREATEDOC directive. The filename is specified exactly as it is in the CREATEDOC directive described above, and the file created is otherwise identical to the document created by CREATEDOC.

<MENUDOC> Link an existing document to a new article

Syntax:

```
<MENUDOC "MarkerText">"Filename"  
LinkText  
</MENUDOC>
```

FORMS PROCESSING COMMANDS

Requires: a CREATEDOC directive

Once you have created an article, you'll probably want to have the article included on an HTML page with a list of links to all the articles available for browsing. With MENUDOC, you specify the HTML page to which you want the link added, complete with the HTML text for building the link. A simple MENUDOC example follows:

```
<MENUDOC>"/Recipes/RecipesMenu.html"  
<LI> <A HREF="<HTMLFILENAME>">  
<REPLACE RecipeName></A>  
</MENUDOC>
```

The LI tag will be added to an unordered list within the file RecipesMenu.html, which will become the list of articles.

If no MarkerText is specified, NetCloak inserts the link text just after every unordered list (UL) tag in the specified menu file. If there is more than one list in the menu document, the link will be added to all lists. The link is added to the top of the list, so that the most recent articles will be displayed first.

If you wish to maintain a menu page with multiple unordered lists of different items, or to insert the menu item into some structure other than an unordered list, you may specify the place within the menu page for the new link to be added. To use this feature, you will need to add text to your menu document at the point where you would like new links to be inserted. While the marker text may consist of anything, it is most useful to use an HTML comment, so that the marker text does not appear in the browser. Then, add the text marking the insertion point to the opening MENUDOC tag in your FDML file. For example, if the menu document contained the comment `<!--INSERT_LINK_HERE-->`, then the MENUDOC directive would be:

```
<MENUDOC "<!--INSERT_LINK_HERE-->"> ... </MENUDOC>
```

Finally, if you would prefer that links to articles in your menu document be added to the bottom of the list instead of the top, add an asterisk to the MENUDOC command, like this:

```
<MENUDOC* ...> ... </MENUDOC>
```

FORMS PROCESSING COMMANDS

The articles will then appear in chronological order (oldest first) instead of reverse-chronological order (newest first)

<RECENTLIST> Add to the list of recently submitted articles

```
<RECENTLIST ListID>  
LinkText  
</RECENTLIST>
```

Requires: a CREATEDOC directive

NetCloak will automatically maintain a recent articles list that allows users to retrieve a list of articles that have been recently added to your server. This command is very much like the MENUDOC directive, except that you do not need to specify a file name to be updated since the article list is maintained internally, and the web page displaying this list is built on-the-fly by NetCloak. For example, to keep a list of recently added recipes, you might use the following example:

```
<RECENTLIST>  
<LI><A HREF="<HTMLFILENAME>">  
<REPLACE RecipeName></A>  
</RECENTLIST>
```

The RECENTLIST directive adds the link to an internal list, which will be automatically time-stamped.

When users ask for recent articles, the internal list is searched and the appropriate links are returned. Recent list requests can be for all articles or only those within a specified time period. Up to 256 links can be stored in the internal list.

NetCloak maintains up to 26 such internal lists, each designated by a letter A-Z. The RECENTLIST directive can include the list to add the articles to as below:

```
<RECENTLIST B>  
<LI><A HREF="<HTMLFILENAME>">  
<REPLACE RecipeName></A>  
</RECENTLIST>
```

The ListID parameter is optional. If no list is specified, the first list (list A) is assumed.

FORMS PROCESSING COMMANDS

For more information on creating and accessing multiple Recent Article lists, see the section entitled “Creating & Accessing Recent Article Lists” in the User’s Guide.

<RESPONSE> Return an HTML file after processing

```
<RESPONSE>"Filename"</RESPONSE>
```

The RESPONSE directive allows you to specify an HTML document that will be returned to the author after they have submitted a form. An example follows:

```
<RESPONSE>"/Feedback/Response.html"</RESPONSE>
```

Note that a filename, not an URL, is specified in the RESPONSE directive.

If you do not use RESPONSE to specify a file, a very simple default Thank You response will be sent back to the author. See “Creating a Response File” in the User’s Guide for more details.

<VERIFY> Verify that fields have been filled out

```
<VERIFY FieldList>"Filename"</VERIFY>
```

Unlike all other supplemental directives, you may use as many VERIFY directives as you like within a single FDML file. Like other directives, VERIFY is best used at the top of the FDML file, before the primary directive.

VERIFY allows you to check one or more input fields to make sure the user entered data into the field(s). If one or more of the fields were left blank by the user, NetCloak will not process the form and instead display the specified error response page, as in the following example:

```
<VERIFY Email UserName Message>  
"/Folder/IncompleteForm.html"  
</VERIFY>
```

You may specify as many fields as you like within the VERIFY tag, and each will be checked. If any of the fields are blank,

FORMS PROCESSING COMMANDS

the file specified will be returned and the form will not be processed.

Note: For more extensive error detection and field validation, use the RETURN_PAGE directive along with the IF command.

<RETURN_PAGE> Return an error page & abort

```
<RETURN_PAGE>"Filename"</RETURN_PAGE>
```

The RETURN_PAGE directive works like the RESPONSE directive, with one important difference. As with the RESPONSE directive, the command specifies a page to be returned to the user in the Filename parameter. However, RETURN_PAGE causes NetCloak to abort all processing of the input form data and simply return the specified page.

There are two cases where RETURN_PAGE can be used.

The first is when all you want to do in response to a form submission is return a page of data. In this case, you can create an FDML file containing nothing but the RETURN_PAGE directive specifying the document to be returned. This might be useful in a simple FAQ document retrieval system, for example, like this:

```
<RETURN_PAGE>
"/FAQs/<REPLACE FAQNumber>.html"
</RETURN_PAGE>
```

The second case, which is far more common, is when you need to perform some kind of input validation on the form data beyond simply checking whether a field is blank. To do this, use RETURN_PAGE in combination with an IF command so that an error page is returned when some condition, specified in the IF command, is met. When the condition is not met the primary directive will perform its task. Here is a common example of this usage of the RETURN_PAGE directive:

```
<IF EmailAddress #CONTAINS "@" THEN
  "<RETURN_PAGE>
  ""InvalidEmail.html""
  </RETURN_PAGE>"
>
```

FORMS PROCESSING COMMANDS

This FDML snippet tells NetCloak “If the form field named 'EmailAddress' does not contain an '@' character, then abort processing and return the page named 'InvalidEmail.html' instead.”

Note that when the RETURN_PAGE directive is used in an IF command, you must use two double-quote characters to delimit the file path to prevent NetCloak from interpreting them as the end of the THEN clause.

<EXPIRES> Automatically delete links & pages

<EXPIRES n units>

Requires: a CREATEDOC directive and a MENUDOC directive, or an INSERTFILE directive.

After a while, a popular forum maintained automatically by NetCloak can generate dozens, even hundreds, of entries in a menu document or INSERTFILE target file. Cleaning up old links and their corresponding files can be a tedious chore for the web server administrator. NetCloak provides the EXPIRES tag to automatically delete links and pages after a configurable period of time has passed.

The period of time is specified with the two parameters to the EXPIRES tag. N represents any numeric value, and the units parameter may be any one of the following: WEEKS, DAYS, HOURS, or MINUTES.

For example, to automatically remove menu document entries and delete the corresponding created files after 90 days, you would put the following directive into the FDML file:

<EXPIRES 90 DAYS>

The EXPIRES directive can be used with a CREATEDOC primary directive and a MENUDOC directive; in that case, it deletes the link in the specified menu document, and also deletes the corresponding file from the web server.

FORMS PROCESSING COMMANDS

It may also be used with the INSERTFILE primary directive, so that it deletes the text inserted into the target HTML file after the period of time has elapsed.

How it works: When the EXPIRES directive is found in an FDML file, NetCloak adds special HTML comment tags around the menu item or the inserted text. These comments encode the date/time that the text is to be removed from the page. When NetCloak serves the menu document, it treats these special tags just like HIDE_DATE commands. Then, when NetCloak inserts a new item into a menu document or INSERTFILE target, it permanently deletes everything between these tags if the expiration date is passed. When it deletes an item from a menu document, it also looks for the corresponding article file and deletes it as well.

<SET_NETCLOAK> Set NetCloak configuration options

```
<SET_NETCLOAK Setting = Value> or  
<NETCLOAK Setting = Value>
```

These NetCloak configuration commands work exactly the same way as they do in cloaked pages (see the section “Commands: NetCloak Configuration” above and “NetCloak Configuration” in the User's Guide); they change the NetCloak configuration setting specified while the FDML file is processed. Like VERIFY directives, you can include as many SET_NETCLOAK or NETCLOAK directives in the page as you want to. The most common use for SET_NETCLOAK directives is to specify the link text used in LINKNEXT/LINKPREVIOUS tags in created articles, like this:

```
<SET_NETCLOAK LinkNextText = "Next Recipe">  
<SET_NETCLOAK LinkPrevText = "Previous Recipe">
```

Thus, you can customize the link text for the kinds of articles the FDML file is creating.

FORMS PROCESSING COMMANDS

Insertion Commands

Insertion commands are used to insert user-entered information into the body of the template contained in the FDML document. The commands used to insert text are as follows:

<REPLACE>	<DATE>
<BULLET>	<YEAR>
<UNIQUE>	<MONTH>
<IF>	<DAY>
<LINKPREVIOUS>	<WEEKDAY>
<LINKNEXT>	<TIME>
<HTMLFILENAME>	<HOUR>
<MENUFILENAME>	<MINUTE>
<VARIABLE_LIST>	<SECOND>
<REPLACE_OFF>	<REPLACE_ON>

<REPLACE>

```
<REPLACE VarName>  
<REPLACE_FN VarName>  
<REPLACE_RAW VarName>  
<REPLACE_URL VarName>
```

REPLACE inserts the contents of the form field variable, global variable, cookie, or counter named VarName. In the case of text entry form fields, this will be the text entered by the user. In the case of radio buttons, checkboxes and other fields which allow the user to select from pre-defined options, it will be the value specified for the field that the user selected.

When NetCloak inserts the variable value, it will replace single line breaks with BR or pairs of line breaks with P tags. If the Convert Brackets configuration setting is enabled, NetCloak will also convert angle bracket characters in the value (< and >) to equivalent HTML entities (< and > respectively).

The three extensions to the standard REPLACE command modify the format of the data when it is inserted.

The REPLACE_FN command will remove or convert characters that are not allowed in file or folder names. This extension must be used within the Filename parameter in any

FORMS PROCESSING COMMANDS

NetCloak directive so that legal folder and file names will be created. For example, a recipe system might start like this:

```
<CREATEDOC>  
"/Recipes/<REPLACE_FN RecipeName>.html"  
</CREATEDOC>
```

In this example, a new HTML page will be created in the Recipes folder and will be named with the title of the recipe the author entered into the submitted form.

The REPLACE_FN command is very strict about the characters it inserts. The only characters that will be inserted will be upper and lower case alphabetic characters, numbers, underscores (_), periods (.), and dashes (-). Colons (:) and slashes (/) entered into the field will be converted to periods. All other characters will be removed. This is to ensure that the result will be both a legal Macintosh file name and a valid URL.

In cases where you would like to insert variable values without any conversion at all by NetCloak, use the REPLACE_RAW command. This suppresses conversion of line breaks and angle brackets to their HTML equivalents normally done by the REPLACE command.

Finally, if you want to insert a variable value into a URL, say when building a link to a database query document or a search page, use the REPLACE_URL extension. Using REPLACE_URL will URL-encode the value before inserting it. URL-encoding converts any non-alphanumeric character in the data to a three-character sequence of the form %XX, where XX is the ASCII value of the character in hexadecimal. This is the format used when a web browser sends form data to a web server in an HTTP GET request.

<BULLET>

```
<BULLET VarName>
```

The BULLET command will create an HTML bullet list from the contents of the specified NetCloak form variable, global variable, cookie, or counter, complete with and

FORMS PROCESSING COMMANDS

 tags. Each line of the specified variable separated by a return character becomes a bullet point. The command

```
<H3>Ingredients -</H3>
<BULLET Ingredients>
```

would place the ingredients entered into a bullet list; the resulting HTML might be something like this:

```
<H3>Ingredients -</H3>
<UL BULLET>
<LI>Malt
<LI>Barley
<LI>Hops
</UL>
```

And the HTML would be rendered like this in a browser:

Ingredients—

- Malt
- Barley
- Hops

This command is almost always used to process a scrolling text field, but any field type is supported.

<UNIQUE>

```
<UNIQUE>
```

The UNIQUE command will insert a string of up to six alphanumeric characters that is guaranteed to be unique each time the FDML containing the command is used to process a form. The text is safe for use in filenames and URLs. This can be used to guarantee a unique filename for created documents, or to generate a unique identifier for each record in a TEXTSTORE file.

Adding an asterisk to the command (UNIQUE*) will insert the same unique value set by a previous UNIQUE command. This is useful for re-using the unique value when producing links or other references to a new file whose name was created with UNIQUE.

FORMS PROCESSING COMMANDS

The original unique value is preserved for an entire submission, so UNIQUE* can be used across multiple chained FDMLs and in the response page.

<IF>

```
<IF VarName operator comparisonValue THEN  
  "TrueText" ELSE "FalseText">
```

The IF command compares the value of a form variable, global variable, cookie, or counter to a constant value or another variable, and inserts a specified string of text into the page if the comparison evaluates TRUE, and inserts another (optional) string of text if the comparison evaluates FALSE. This allows you to insert different text or execute different commands based on the contents of a field.

After the name of the variable to be tested comes one of the NetCloak comparison operators, which determines how the variable value will be compared to the constant. The supported operators are covered in detail in the section General Rules of NetCloak Syntax. The list of operators supported is shown below:

<u>Text Comparisons</u>	<u>Number Comparisons</u>
CONTAINS	== (Equals)
BEGINS (With)	LT (Less Than)
ENDS (With)	LT= (Less Than or Equal)
BEFORE (Alphabetic)	GT (Greater Than)
AFTER (Alphabetic)	GT= (Greater Than or Equal)
IS (Exact Match)	IN

The comparison operator is optional. If it is not present, then an IS text comparison is performed, requiring an exact match between the named form field value and the text parameter.

As in NetCloak HTML commands, you can combine the comparison operator with a '#' character to negate the result of the comparison.

Note: For backward compatibility with older versions, the '=' text operator is still supported, operating identically to the IS operator. The IS operator is preferred.

FORMS PROCESSING COMMANDS

The next parameter is the comparison value. This can be a variable or field name, or a text string. Text strings should be enclosed in double-quotes or else NetCloak will attempt to interpret the text as a variable name first. If the comparison value is not enclosed in quotes and there is no variable with that name, NetCloak will use the specified value as text.

Comparison values can be form fields or NetCloak global or user variables. NetCloak will use variables in that order if there are field values, global variables or user variables with duplicate names.

The keywords THEN and, if used ELSE, are required. However, they do not have to be upper case. Uppercase is recommended because it enhances the readability of the command.

To check to see if a field was left blank by the author, use two double-quotes with nothing between them . For example, a common IF usage is:

```
<IF MyTextField IS "" THEN "The text was blank">
```

You may use any FDML directive or command within either replacement string.

Unlike NetCloak commands, the FDML IF command requires double-quote characters around the text values following the comparison operator.

If you need to use a double-quote within the text value, the THEN clause, or the ELSE clause, then use two double-quotes in the text of the parameter. For example, the IF statement above could have read:

```
<IF MyTextField IS "" THEN "Clarus the dogcow said  
""Moof!""">
```

If MyTextField were left blank, the inserted text would be:

```
Clarus the dogcow said "Moof!"
```

The double-quote pair can also be used in the comparison constant.

FORMS PROCESSING COMMANDS

The IF command can accept multiple conditional statements using AND and OR conjunctions. For example:

```
<IF Name IS "John" AND EMail CONTAINS "@" THEN  
  "John entered his e-mail address.">
```

You can also use OR, as in:

```
<IF Name IS "John" OR Name IS "Bob" OR Name IS  
"Bill" THEN "Hi <REPLACE Name>!">
```

The IF command also supports an ELSE clause. If the condition is true, the first replacement string will be used. If the condition is false, the second replacement string, following the ELSE, will be used. For example:

```
<IF Name IS "John" OR Name IS "Bob" OR Name IS  
"Bill" THEN  
  "Hi <REPLACE Name>!"  
ELSE  
  "Sorry, <REPLACE Name>, I don't know you...">
```

The IF command above will replace the value of the field Name if it contains either John, Bob, or Bill. Otherwise, it will insert "I don't know you...".

Note, finally, that there is frequently more than one correct way to compare field values using an IF command. For instance, the example given above, using 'IS' and 'OR' operators, could be rewritten using the 'IN' operator, like this:

```
<IF Name IN "John,Bob,Bill" THEN  
  "Hi <REPLACE Name>!"  
ELSE  
  "Sorry, <REPLACE Name>, I don't know you..."  
>
```

<DATE>

```
<DATE>  
<DATE_FN>  
<DATE_URL>
```

The DATE command inserts the system date of the machine running NetCloak into the text, in the short date format

FORMS PROCESSING COMMANDS

specified in the Date & Time control panel. You might use this to date when your articles are posted, for example:

This article was submitted on <DATE>.

If you want to use the DATE command as part of a Filename parameter in other FDML directives, use the extension DATE_FN. For example:

```
<COPY>"/FolderName/<REPLACE_FN  
TitleName><DATE_FN>.html"</COPY>
```

The above insertion would provide a date-stamped title to each article submitted. NetCloak will replace the slashes normally found in the date into periods so that they are not misinterpreted by the server as path delimiters.

Use the DATE_URL extension to URL-encode the current date before it is inserted, as you would need to do when inserting the date into a hypertext link URL. See the REPLACE command for a description of URL-encoding.

<YEAR>

<YEAR format>

The YEAR command inserts the current year, according to the system date, into the text.

The optional format parameter can be either LONG or SHORT, and defaults to LONG if no format is specified. If SHORT format is specified, the year will be returned as a two-digit number without the century.

<MONTH>

```
<MONTH>  
<MONTH# format>  
<MONTH_FN>  
<MONTH_URL>
```

The MONTH command inserts the current month name, according to the system date, into the text. The month name is given in the language set in the Date & Time control panel.

FORMS PROCESSING COMMANDS

The `MONTH#` extension merely inserts the value (1–12) corresponding to the current month instead of the month name. The optional format parameter can be either `LONG` or `SHORT`, and defaults to `SHORT` if no format is specified. If `LONG` format is specified, single-digit numbers will include a leading zero. The format parameter may be included in other forms of the command, but has no effect.

Because in some languages, month names contain non-alphanumeric characters, you can use the `MONTH_FN` extension to include the month name in `Filename` parameters in other `FDML` directives. Similarly, you can use the `MONTH_URL` extension to insert an URL encoded month name into URL text. See the `REPLACE` command for a description of URL-encoding.

<DAY>

<DAY format>

The `DAY` command inserts the value for the day of the month into the text. This value ranges from 1 to 31. The optional format parameter can be either `LONG` or `SHORT`, and defaults to `SHORT` if no format is specified. If `LONG` format is specified, the day will include a leading zero for single-digit numbers.

<WEEKDAY>

<WEEKDAY>
<WEEKDAY#>
<WEEKDAY_FN>
<WEEKDAY_URL>

The `WEEKDAY` command inserts the current day of the week, according to the system date, into the text. The name of the day is given in the language set in the `Date & Time` control panel.

The `WEEKDAY #` extension merely inserts the value (1–7) corresponding to the weekday instead of the name of the day.

Because in some languages, weekday names contain non-alphanumeric characters, you can use the `WEEKDAY_FN` extension to include the month name in `filename` parameters

FORMS PROCESSING COMMANDS

of other FDML directives. Similarly, you can use the WEEKDAY_URL extension to insert an URL encoded weekday name into URL text. See the REPLACE command for a description of URL-encoding.

<TIME>

<TIME>
<TIME_FN>
<TIME_URL>

The TIME command inserts the current time of the web server running NetCloak in the same way the date is inserted using the DATE command.

If you want to use the TIME command in a filename parameter in other FDML directives, use the extension TIME_FN (where FN stands for File Name). This command will replace the colons normally found in the expression of time into periods so that they are not misinterpreted by the server.

Similarly, if you want to use the current time in a URL embedded in the text, use the TIME_URL command. See the REPLACE command for a description of URL-encoding.

See the DATE command for an example of using the _FN variation of the command.

<HOUR>

<HOUR format>

The HOUR command inserts the hour of the current time into the text. This value ranges from 0 to 23.

The optional format parameter can be either LONG or SHORT, and defaults to SHORT if no format is specified. If LONG format is specified, single-digit numbers will include a leading zero.

<MINUTE>

<MINUTE format>

FORMS PROCESSING COMMANDS

The MINUTE command inserts the minute of the current time into the text. This value ranges from 0 to 59.

The optional format parameter can be either LONG or SHORT, and defaults to SHORT if no format is specified. If LONG format is specified, single-digit numbers will include a leading zero.

<SECOND>

`<SECOND format>`

The SECOND command inserts the second of the current time into the text. This value ranges from 0 to 59.

The optional format parameter can be either LONG or SHORT, and defaults to SHORT if no format is specified. If LONG format is specified, single-digit numbers will include a leading zero.

<HTMLFILENAME>

`<HTMLFILENAME>`

The HTMLFILENAME command is used to specify the path and filename of a newly created HTML article so that links can be created to it. It works with the CREATEDOC and INSERTFILE primary directives.

Used in the text of a response file, you can allow authors to review their finished articles. In the MENUDOC directive, HTMLFILENAME is especially useful to provide a link to the new article. Simply place the HTML link command in the link text parameter of the MENUDOC command.

`<A HREF="<HTMLFILENAME>">`

In a response page, you could use HTMLFILENAME command to allow users to see their finished article:

`<A HREF="<HTMLFILENAME>">Review Your Article</A>`

Note that if several FDMLs are chained together with the CHAIN directive, the value of HTMLFILENAME will only be valid within CREATEDOC FDMLs. If the last FDML in the

FORMS PROCESSING COMMANDS

chain uses CREATEDOC, HTMLFILENAME will also work in the response file.

<MENUFILENAME>

<MENUFILENAME>

The MENUFILENAME command is used to specify insertion of a URL to the menu document of a newly created article. In most cases, this command is used to provide a hypertext link back to the new article's menu. Obviously, this command can only be used when a MENUDOC directive has been specified.

<LINKPREVIOUS>

<LINKPREVIOUS>

The LINKPREVIOUS command allows you to add a hypertext link between the document currently being processed and the last document that was added to the same menu (in the MENUDOC directive) of an HTML page. Users can navigate between articles that are on the same menu without having to constantly be moving back and forth between the menu document and the individual articles. Simply insert the command into the text template of your FDML file, like this:

<LINKPREVIOUS>

By default, the hypertext link will be labeled with the text, "Previous Article". You can change this phrase to anything you like in the NetCloak configuration settings, or using a SET_NETCLOAK command in your FDML file.

<LINKNEXT>

<LINKNEXT>

The LINKNEXT command is identical in its implementation to the LINKPREVIOUS command except that the hypertext link inserted is labeled "Next Article" by default, and when the next article is submitted, NetCloak automatically adds a link to the new document in place of "Next Article". As with LINKPREVIOUS, the phrase used in the hypertext link can be

FORMS PROCESSING COMMANDS

customized in the NetCloak configuration settings, or with a SET_NETCLOAK command.

<VARIABLE_LIST>

```
<VARIABLE_LIST>  
HTML<VARNAME>HTML<VARVALUE>HTML  
</VARIABLE_LIST>
```

This command behaves the same way in an FDML file that it does in cloaked HTML pages – it repeats the text in between the opening and closing VARIABLE_LIST tags once for each form field in the current form submission. Within this repeating text, each occurrence of the tag VARNAME is replaced with the variable name, and each occurrence of the tag VARVALUE is replaced with the corresponding variable value. For more information, see the VARIABLE_LIST dynamic page command.

<REPLACE_OFF>

<REPLACE_ON>

```
<REPLACE_OFF>  
<REPLACE_ON>
```

Sometimes, you may not want FDML insertion commands in an FDML file to be processed immediately. This most often occurs when you want to use an FDML file to actually create another FDML file.

In these cases, you can use the special commands REPLACE_OFF to tell NetCloak to *not process* FDML insertion commands (including REPLACE, IF, BULLET, VARIABLE_LIST, date/time commands, etc.) until the next REPLACE_ON command is encountered in the FDML file.

These commands also come in handy with the VARIABLE_LIST command, since it is supported in both cloaked pages and in FDML file templates. To avoid replacing the VARIABLE_LIST tags when the FDML processes the form, you could do this, for instance:

FORMS PROCESSING COMMANDS

```
<PRE>
<REPLACE_OFF>
<VARIABLE_LIST>
<B><VARNAME>:</B> <VARVALUE>
</VARIABLE_LIST>
<REPLACE_ON>
</PRE>
```

Using Multiple Instances of a Field

When a field name is used multiple times within the same HTML form (as in a series of checkboxes), you can access instances other than the first. For example, the first three items chosen from a group of checkboxes named MyOptions could be added to a page using...

```
<REPLACE MyOptions[1]><BR>
<REPLACE MyOptions[2]><BR>
<REPLACE MyOptions[3]><BR>
```

The square brackets are used to specify the number of the instance to be used. If the brackets are left off and only the variable name is given, then the first instance is used. The IF command can be used to test to see if a particular instance exists. For example, to include Option: before each selected item in the example above, you could use...

```
<IF MyOptions IS "" THEN "No Options
Selected!<BR>">
<IF MyOptions[1] IS# "" THEN "Option: <REPLACE
MyOptions[1]><BR>">
<IF MyOptions[2] IS# "" THEN "Option: <REPLACE
MyOptions[2]><BR>">
<IF MyOptions[3] IS# "" THEN "Option: <REPLACE
MyOptions[3]><BR>">
<IF MyOptions[4] IS# "" THEN "More than 3 options
selected!<BR>">
```

ADJUSTING INTERNAL SETTINGS

This section is for advanced users whose system exceeds the limitations imposed by NetCloak on back-end CGI response time-outs or the maximum number of counters available; and who have experience using a resource editor. In the vast majority of cases, you will never need these instructions.

NetCloak stores the default values for two operational parameters inside 'MaxP' resources in the resource fork of the ACGI and the plug-in. Refer to the table below for details on how NetCloak uses these resources.

If you need to edit these resources, please take the usual precautions when using a resource editor like ResEdit: always make a duplicate copy of the file to be edited, edit the copy of the file, save it and test your changes *before* throwing away the original and renaming the copy.

ID	Name Description	Min	Max	Default
202	CGI Time-out back-end CGI time-out, in ticks (60 ticks = 1 second).	60	36000	3600
203	Max Counters Determines size of counter database.	32	16384	512

You might need to change the CGI Time-out value if you have set up back-end CGIs or use EXEC_CGI commands, and your users often see the message "NetCloak Error: The back-end CGI did not respond to the request from NetCloak" returned to the browser. This might mean that the CGI is taking longer to return its results than NetCloak expects.

If you find that you need more than 512 counters, you can increase the Max Counters value in 'MaxP' resource ID 203. Or if you are sure you will never need that many counters, and you want to conserve memory, you can decrease this value.

When you change the Max Counters value, you should also change the memory requirements of the web server application (if using the plug-in) or the ACGI using the Finder's Get Info command. For each counter you

ADJUSTING INTERNAL SETTINGS

add/remove, you should increase/decrease the memory requirements by 40 bytes. For example, if you want to increase from 512 to 1024 counters, you should increase the memory requirements by $40 \times 512 = 20480$ bytes = 21K.

ACTION field	66
ASCII.....	6, 56, 60, 71, 115
BEEP.....	16, 78, 79
Begins	5
BEGIN_MACRO.....	87, 88
CGI Timeout.....	127
Checkbox	92
CLIENT.....	9, 11, 13, 15, 18, 39, 40, 41
COOKIE	7, 10, 16, 65, 74, 75, 76, 77, 78
COUNT	8, 9, 11, 15, 26, 27, 28, 29, 30
COUNTDOWN.....	11, 15, 22, 25, 26
Counter.....	30
COUNTERDATE.....	15, 29, 30
COUNTERTIME.....	15, 30
DATE.....	10, 15, 20, 21, 22, 23, 24, 26, 30, 45, 46, 52, 76, 100, 113, 114, 119, 120, 122
Date & Time control panel.	18, 20, 21, 24, 29, 46, 64, 120, 121
DATE command.....	10, 20, 23, 24, 26, 30, 45, 46, 113, 119, 120, 122
Date format.....	20, 30, 64
DAY	3, 10, 15, 23, 24, 36, 114, 121
DelSol	78
DOMAIN.....	15, 30, 31, 32, 33, 80
END_MACRO.....	87, 88
EXEC_CGI.....	12, 16, 59, 82, 83, 84, 127
Failed Mail	104
Feedback.....	110
Filename	96, 97, 106, 107, 110, 111, 120
FILESIZE.....	16, 46, 47
FLIPMODE.....	15, 17, 18, 51, 54, 55, 59, 64
Footer.....	59
Form	3, 10, 67, 91, 94
FORM command	66
GET	58, 90, 91, 115
GLOBAL	4, 5, 6, 7, 12, 16, 66, 68, 70, 72, 73
HEADER.....	16, 49, 50, 51, 52
Hidden field.....	94

INDEX

HOST	16, 47, 48, 49
Host field	47, 48
HTML entities.....	63, 114
HTML form.....	54, 58, 66, 84, 89, 90, 103, 126
HTTP header.....	47, 48, 49, 50, 51, 61, 74
IF/THEN/ELSE	18
Internet Explorer.....	39, 40
ISO-Latin	60
LOCAL.....	12, 14, 16, 52, 68, 69, 70, 71, 72, 73, 81, 82
MACRO	12, 16, 49, 59, 86, 87, 88
Mail Queue.....	104
Max Counters.....	127
MaxP.....	127
MODIFIED	16, 44, 45, 46, 83, 86
Mozilla	9, 40, 41
NETCLOAK.....	16, 60, 61, 63, 64, 106, 113, 124, 125
NetCloak Files folder.....	39, 104
Password.....	35, 38, 91
Path.....	55, 56
PIXO.....	16, 84, 85
POST.....	38, 58, 66, 67, 89, 90, 91
Query arguments.....	84
Radio button	92
Random	16, 41
Recipe	113
REDIRECT	16, 49, 55, 79, 80
REFERRER.....	16, 52, 53, 54
REPLACE_FN.....	96, 114, 115
REQUEST_PASSWORD.....	15, 33, 35, 36, 37, 38
ResEdit.....	127
Resets, counter	29
Responder	83, 84
SAMERAND.....	11, 16, 42, 43, 44
SEARCH.....	16, 58, 59
Search arguments.....	59, 90
SET_COOKIE	7, 16, 65, 74, 75, 76, 77
SET_GLOBAL.....	7, 16, 66, 68
SET_LOCAL.....	14, 16, 52, 68, 69, 70, 71, 81, 82
SET_VARIABLE	12, 13, 16, 68, 75

Short Date	20, 30
SMTP	100, 101, 103, 104
Submit	67, 94
Suffix, file	99
THISURL.....	16, 54, 55
TIME.....	15, 18, 19, 20, 24, 30, 100, 114, 122
URL encoded	56, 121, 122
USERNAME	15, 33, 34, 36
VARIABLE.....	9, 12, 13, 14, 16, 67, 68, 70, 71, 72, 73, 75, 82, 114, 125, 126
WebSTAR.....	32, 84, 85
Wildcards.....	11, 12
YEAR command.....	120



Maxum Development Corporation

Post Office Box 315

Crystal Lake, IL 60039-0315

Printed in the USA

Copyright ©1996-2000, Maxum Development Corporation. All Rights Reserved