SFX Script Commands

SFX Script Commands are an advanced feature of SFX and are found in the Deluxe edition. Script Commands can be entered into the Command Interface to be executed immediately, or they can be stored and executed through triggers or Script cues in your cue list. A SFX script can be one command or it can be several commands and scripts can even execute other scripts. You can use SFX Scripts to automate tasks, perform functions, or make choices depending on inputs or situations.

Examples of what you can do with SFX script commands:

- Play random cues
- Play cues depending on the specific performance, whether it's a matinee or evening for instance
- Change all or some Wait times depending on the Wait time
- Load productions based on the day of the week
- Store values
- More!

Language Conventions

Each SFX Script command in this document will be displayed in the following format:count

Command Name

Syntax

Description

Example

The syntax shows how the command should be formatted, including what parameters it takes and how they should be formatted. Capital letters indicate literal text while italics indicate a substitution. The vertical pipe symbol | means "or" so that INPUT|OUTPUT means to choose from either INPUT or OUTPUT. When something is enclosed in square brackets [] it means it is optional.

For example,

ASSIGN *variable* = *value*|*statement*

variable = variable name such as @temp
value = number or text to assign to *variable*
statement = another SFX Script command

Example:
ASSIGN @list = "A"
ASSIGN @number = COUNT LISTS

Language Constructs

There are several important constructs about the SFX Script language:

- Variables
- Boolean operations
- Mathematical operations
- Comments
- Data types
- Passing parameters

Variable - a variable is any alphanumeric text name that is used to store values. Variables are preceded by the @ character to differentiate them from the other reserved words of SFX Script. A variable can be of any length, be composed of letters and numbers preceded by the @ character: @temp, @number3, @cueLists. Variables, by default, are global, meaning that a variable defined in one script is visible to other scripts. It is recommended that if you are using a variable name that is used only within a script, you should use the LOCAL ASSIGN command instead of the ASSIGN command. To display a variable outside the script file, use GLOBAL ASSIGN.

Boolean operations - operations that return a True(1) or False(0) result.

== equals. Notice this contains two equals signs. e.g. @list == "Main"
<> not equal
> greater than
< less than
>= greater than or equal to
<= lesser than or equal to

OR and AND. e.g.

```
IF(@list == "A" AND @num < 5) THEN
   // statements go here...
END
```

CONTAINS. e.g.

```
IF (@list CONTAINS "day") THEN // return true if @list has the word "day" in it.
   // statements go here...
END
```

Mathematical operations - add, subtract, multiply, divide, square root

/ divide
* multiply
% modulus (returns the remainder of a division operation)
- subtract
+ add
& concatenate strings. e.g. "ABC" & "DEF" becomes "ABCDEF"
(*expression*) order precedence. e.g. 4 * (3 + @value)

Comments - Comments are text you include in the SFX Script that explain what the script is doing. The computer will ignore comments. The command symbol is "//" and everything to the right of the // is ignored.

```
// get the number of cue lists
ASSIGN @num = COUNT LISTS
ASSIGN @num = @num + 1 // add one
```

You can also comment out more than one line by using the start comment /* and closing it with the */. The following snippet will be ignored because the two statements are within a comment field.

```
/*
   ASSIGN @lists = COUNT LISTS
   DISPLAY @lists
*/
```

Data Types - SFX Script commands return data in one of two formats: Numeric or Text. When something is stored as a Number it must be either an integer or floating point number. Text data is enclosed in double quotation marks. Anything can be stored as Text but when storing numbers as text you may not get the proper results. For example:

```
ASSIGN @num1 = 1
ASSIGN @num2 = "2"
ASSIGN @num1 + @num2 // will return "12" and not "3" because @num2 is storing the number as text.
```

One way to correct this would be to use the CONVERT statement:

```
ASSIGN @num1 = 1
ASSIGN @num2 = "2"
CONVERT @num2 to INT
ASSIGN @num1 + @num2 // will return "3" because both variables are storing numbers.
```

Arrays - Data can can also be stored in an array, which is a variable that can store a set of values. For example,

```
// the @temp variable has three items
ASSIGN @temp[0] = "First"
ASSIGN @temp[1] = "Second"
ASSIGN @temp[2] = "Third"
```

The following example, while it's not practicle, illustrates how arrays are created and accessed:

```
// loop through the array assigning random numbers
FOR @i = 1 to 10
   ASSIGN @rand[@i] = RANDOM 100 TO 200
LOOP
DISPLAY "The 3rd random number picked was: " & @rand[3]
DISPLAY "The 5th random number picked was: " & @rand[5]
```

You can use the the GETARRAYCOUNT statement to determine how many elements are in an array. Array's index start at zero. e.g. @value[0], @valuep[1] etc.

Passing Parameters - SFX Script will accept parameters. The variable @ARGSCOUNT contains the number of passed in variables. Each variable is assigned to a succeeding variable named @ARGS0, @ARGS1, @ARGS2, etc. See the EXEC command to see how to pass parameters. The @ARGS variables are NOT global variables are are only visible to the scripts that they are passed to.

SFX Script Commands

SFX Script commands are composed of several types of statements:

1. Script Language - controls aspects of the script such as assigning values to variables, looping, branching, etc.
    - GETARRAYCOUNT
    - ASSIGN
    - LOCAL ASSIGN
    - GLOBAL ASSIGN
    - CONVERT
    - DISPLAY
    - DEBUG
    - TRACE
    - PROMPT
    - FOR LOOP
    - WHILE DO
    - IF THEN
    - BREAK
    - RETURN
    - RANDOM
    - DELAY
    - GET TIME
    - CALL
    - EXEC
    - LAUNCH
2. Math - statements to peform mathematical functions

        - ABS
        - SQRT
        - ROUND
    3. Text - statements that deal with text properties and manipulation
        - TEXTLENGTH
        - TEXTPOSITION
        - SUBTEXT
    4. Application - deals with aspects of the program such as exiting, opening/closing productions, printing, etc.
    - EXIT
    - NEW
    - OPEN
    - SAVE
    - CLOSE PRODUCTION
    - NEW LIBRARY
    - OPEN LIBRARY
    - SAVE LIBRARY
    - CLOSE LIBRARY
    - NEW CUES
    - REMOVE CUES
    - EXPORT
    - IMPORT
    - PRINT
    - PRINT PREVIEW
    - TOOLBAR
- Cue List - controls and modifies Cue List properties.
    - ACTIVATE [EFFECT] LIST
    - COUNT LISTS
    - COUNT CUES
    - FOREACH...PRODUCTION
    - FOREACH...LIBRARY
    - FOREACH...[EFFECT] LIST
    - FOREACH...EFFECT [EFFECT] LIST
    - GET...PROPERTY
    - SET...PROPERTY
    - CUELIST TRIGGERS
    - CUELIST TOOLBAR
- Cue commands - reads and modifies cue properties as well as other cue specific attributes.
    - UNDO
    - REDO
    - CLEAR UNDO
    - COPY
    - DELETE
    - INSERT
    - RENUMBER
    - LEARN TIMECODE
    - COLLAPSE
    - EXPAND
    - GET STANDBY
    - GET
    - SET
- Transport commands - controls the execution of cues such as playing, pausing, stopping, seeking, etc.
    - FIRE
    - PRECUE
    - GO
    - PLAY
    - PAUSE
    - STOP
    - STOPALL | PANIC
    - SELECT

    - STANDBY
    - GOTO
    - FIRST
    - LAST
    - PREVIOUS
    - NEXT
    - SEEK
    - GET TIMECODE
    - SET TIMECODE
    - START TIMECODE
    - STOP TIMECODE
- Layout commands - deals with the production's Layouts.
    - SWITCH TO LAYOUT
    - CREATE LAYOUT
    - GET LAYOUT
    - SET LAYOUT
- Script and Trigger commands - deals with scripts and triggers
    - COUNT SCRIPTS
    - ADD SCRIPT
    - REMOVE SCRIPT
    - COUNT TRIGGERS
    - ADD TRIGGER
    - DELETE TRIGGER
    - GET TRIGGER
    - SET TRIGGER
- Patch commands - handles the Audio, MIDI, Serial, and Telnet patches.
    - CREATE AUDIOPATCH
    - DESTROY AUDIOPATCH
    - PATCH AUDIO
    - COUNT AUDIOPATCH
    - ADD AUDIOPATCH
    - DELETE AUDIOPATCH
    - GET AUDIOPATCH VOLUME
    - SET AUDIOPATCH VOLUME
    - GET AUDIOPATCH CROSSPOINT
    - SET AUDIOPATCH CROSSPOINT
    - ADD MIDIPATCH
    - GET MIDIPATCH
    - SET MIDIPATCH
    - GET SERIALPATCH
    - ADD SERIALPATCH
    - SET SERIALPATCH
    - ADD TELNETPATCH
    - GET TELNETPATCH
    - SET TELNETPATCH
    - DELETE TELNETPATCH

Script Language Statements

Script language statements deal with assigning variables, looping, branching, prompting for more information, getting system information, etc.

GETARRAYCOUNT

GETARRAYCOUNT *variable*

variable - variable name such as @temp

Returns the size of the array.

Examples:
ASSIGN @size = GETARRAYCOUNT @temp

ASSIGN

ASSIGN *variable* = *value*

variable - variable name such as @temp that will contain text or number data
value - number, text, or statement return value to assign to *variable*

Stores data inside a variable. Variables assigned in this way are visible to other scripts. To create a variable that is only visible to the current script, see LOCAL ASSIGN.

Examples:
ASSIGN @list = "A"
ASSIGN @number = COUNT LISTS
DISPLAY "There are " & @number & " lists."
ASSIGN @part = 2
ASSIGN @cue = "1" & "." & @part
ASSIGN @waitTime = @baseWait

LOCAL ASSIGN

LOCAL ASSIGN *variable* = *value*

variable - variable name such as @temp that will contain text or number data
value - number, text, or statement return value to assign to *variable*

Stores data inside a variable that is only visible to the currently running script.

Examples:
LOCAL ASSIGN @Min = 100

GLOBAL ASSIGN

GLOBAL ASSIGN *variable* = *value*

variable - variable name such as @temp that will contain text or number data
value - number, text, or statement return value to assign to *variable*

Stores data inside a variable. Variables assigned in this way are visible to other scripts. To create a variable that is only visible to the current script, see LOCAL ASSIGN.

Examples:
GLOBAL ASSIGN @list = "A"

CONVERT

CONVERT *variable* TO INT|HEX|DEC|TEXT

variable - variable name such as @temp that will contain text or number data

Converts a variable to the underlying type of Integer (1, 2, 5, 33), Decimal (1.2, 45.6), Hexadecimal (1F, AB, 7F), or Text("3", "Main"). Converting variables is necessary in some circumstances when SFX Script statements return numerical data in text format.

Example:

```
ASSIGN @cueNumber = GET CUE 4 CUENUMBER
// @cueNumber is stored as Text; convert it to a number
CONVERT @cueNumber TO INT
CONVERT @temp TO TEXT
```

DISPLAY

DISPLAY *message*

Creates a dialog box containing the text *message*.

Example:

```
ASSIGN @numLists = COUNT LISTS
DISPLAY "Number of lists: " & @numLists
```

DEBUG

DEBUG ON|OFF [*filename*]

filename - full path name of file. e.g. "C:\temp\SFXDebug.txt"

When placed within a script, sends debug information to a log file named *filename*. ON starts logging, and OFF stops logging.

Example:

```
DEBUG ON "C:\temp\SFXDebug.txt"
PLAY CUE 10
FIRST
DEBUG OFF
```

TRACE

TRACE *message*

message - expression
Outputs a message to the Command Interface logging window when Debug is ON.

Example:

```
DEBUG ON
ASSIGN @numLists = COUNT LISTS
TRACE "Number of lists: " & @numLists
DEBUG OFF
```

PROMPT

PROMPT *message* YESNO|VALUE|OPEN FILENAME|SAVE FILENAME|CHOICE (*valueList...*)|BUTTONS (*valueList...*)

message - a text message or variable
valueList - used only for the CHOICE type. Example: CHOICE ("Main", "Show", "Utility")

Display a dialog box to the user of the specified type with the prompt of *message* and returns as a text string the user's response.

YESNO will create a dialog box with Yes and No buttons. Returns "Yes" or "No" depending on the button pressed.
VALUE will create a dialog with a text input box. Returns the text that that user types into the box.
OPEN FILENAME will display the Open File dialog box and returns the full path the user selected.
SAVE FILENAME will display the Save File dialog box and returns the full path the user selected.
CHOICE will display a dropdown list containing an item for every item in the *valueList.* Returns the item the user selected.
BUTTONS will display each item in *valueList* as its own individual button.

Examples:
```
ASSIGN @result = PROMPT "Continue?" YESNO
IF @Result == "Yes" THEN
    PLAY CUE 10
END

ASSIGN @actor = PROMPT "Who is playing Tom?" CHOICE ("John Adams", "George Washington")
```

FOR LOOP

FOR *variable = start* TO *end*
    statement(s)...
LOOP

variable - variable name such as @temp that will contain the current number of the iteration
start - a number, variable, or statement that produces a number
end - a number, variable, or statement that produces a number
statement(s) - one or more SFX Script commands

The For Loop repeats the statement(s) contained between the FOR and the LOOP. The statement begins by assigning to *variable* the *start* value, executes the commands until the LOOP statement, and then repeats at the beginning of the FOR loop where variable is incremented/decremented by 1.

Example:

```
// this script will iterate through 5 times: from 1 to 5 inclusive
// @cueNum will contain the values of 1, 2, 3, 4, 5 on each iteration
FOR @cueNum = 1 to 5
  PRELOAD Q @cueNum
 // more commands can go here...
LOOP
```

WHILE DO

WHILE *condition* DO
    statement(s)...
LOOP

condition - a statement that resolves to either a True or False answer. For example: @num < 10 or @List = "Main"
statement(s) - one or more SFX Script commands

Loops as long as the expression evaluates to True(1).

Example:

```
ASSIGN @type = "Sound"
ASSIGN @idx = 1
WHILE @type == "Sound" DO
   ASSIGN @type = GET INDEX @idx Type
   ASSIGN @idx = @idx + 1
LOOP
```

IF THEN

IF...THEN...[ELSE IF/ELSE]...END

IF *condition* THEN
   statement(s)...
END

IF *condition* THEN
   statement(s)...
ELSE
   statement(s)...
END

IF *condition* THEN
   statement(s)...
ELSE IF *condition* THEN
   statement(s)...
[ELSE
   statement(s)...]
END

The IF THEN statement is a conditional statement meaning that if the *condition* evaluates as true (1), the statements immediately following the THEN are executed. Optionally, you can include an ELSE clause to execute statements in the event *condition* does not evaluate to true. Also, you can include one or more ELSE IF clauses to test other conditions.

Examples:

```
IF @list == "A" THEN
    START TIMECODE
END
```

```
IF @list == "A" THEN
    PLAY CUE 10
ELSE IF @list == "B" THEN
    PLAY CUE 20
ELSE IF @list == "C" THEN
    PLAY CUE 30
ELSE
    PLAY CUE 40
END
```

BREAK

BREAK

When called from within a FOR LOOP, FOREACH LOOP, or a WHILE loop it will exit from the loop.

Example:

```
WHILE @list == "Main" DO
    // perform operations....
    IF @error > 0 THEN
      BREAK
    END
LOOP
```

RETURN

RETURN [*value]*

value - a text string or a variable that should be returned to the caller

RANDOM

RANDOM *start* TO *end*

Returns a random number between the values of *start* and *end*, inclusive.

Example:
ASSIGN @number = RANDOM 10 TO 20

DELAY

DELAY *seconds*

seconds - number

Delays for *seconds* before continuing with script.

GET TIME

GET YEAR | MONTH | DAY | DAYOFWEEK | TIME | HOUR | HOUR12 | HOUR24 | MINUTE | SECOND | MILLISECOND

Returns the corresponding date or time value.

YEAR returns the full year, e.g. "2008"
MONTH return the two digit month, e.g. "2"
DAY returns the date, e.g. "15"
DAYOFWEEK returns the day of the week, e.g. "Sunday"
TIME returns the full time in 24-hour format in the form "hh:mm:ss.mmm", e.g. 2:45 PM would be "14:45:25.343"
HOUR12 return the time in 12-hour format, e.g. "3"
HOUR and HOUR24 returns the hour on 24-hour format, e.g. "15"
MINUTE returns the minute, e.g. "23"
SECOND returns the seconds, e.g. "45"
MILLISECOND returns the milliseconds, e.g. "325"

Example:

ASSIGN @dayOfWeek = GET DAYOFWEEK

CALL

CALL *variable*

variable - the name of the subroutine to execute.

Executes a subroutine with the name of *variable*.

Example:

```
SUBROUTINE @GetRandom
ASSIGN @temp = RANDOM 1 to 9
ASSIGN @temp = @temp * 10
END SUB

// this is actually where the script will start
ASSIGN @num = CALL @GetRandom // the subroutine returns the last value of the final statement
DISPLAY @num
ASSIGN @num = CALL @GetRandom
DISPLAY @num
```

EXEC

EXEC FILE *filename*
EXEC FILE *filename* (*valueList...*)
EXEC SCRIPT *script*
EXEC SCRIPT *script* (*valueList...*)

filename - full path name of a saved Script file
valueList - comma delimited parameters to pass to script (e.g. "Main", "Show", "Utility")
script - name of script

Executes a script from file *filename* or a script within the production named *script*.

Every script can check the @ARGSCOUNT variable which will return the number of passed in parameters to the last script run. If parameters are passed in, they will fill the variables, @ARGS0, @ARGS1, @ARGS2, etc.

Example:

```
EXEC FILE "C:\temp\fo.scr"
EXEC SCRIPT "f0" ("Main", "Utility", "Shows") // the script "f0" can access the parameters Main, Utility, and Shows through @ARGS0, @ARGS1, and ARGS2 respectively
```

LAUNCH

LAUNCH *pathname*, *initialDirectory*, [PROMPTARGUMENTS] *parameters*

pathname - full path of a executable file
initialDirectory - the directory to execute the *pathName* from
parameters - a text string containing all the parameters

If PROMPTARGUMENTS is specified a dialog box will appear prompt the user to enter in the parameters.

EXAMPLE:
```
LAUNCH "c:\windows\notepad.exe", "C:\temp", "file.txt"
```

Math Statements

ABS

ABS *value*

Returns the absolute value of *value*.

value- variable or number

Example:

```
ASSIGN @temp = "-23.45678"
ASSIGN @newTemp = ABS @temp
```

SQRT

SQRT *value*

Returns the square root of *value*.

value- variable or number

Example:

```
ASSIGN @temp = 35
ASSIGN @newTemp = SQRT @temp
```

ROUND

ROUND *value, numDecimals*

Returns the rounded *value* to *numDecimals*.

value- variable or number

Example:

```
ASSIGN @temp = 35.351
ASSIGN @newTemp = ROUND @temp, 1 // returns 35.4
```

Text Statements

TEXTLENGTH

TEXTLENGTH *text*

Returns the number of characters in the text *text*.

text - variable or text string

Example:

```
ASSIGN @temp = "23.45678"
ASSIGN @length = TEXTLENGTH @temp
```

TEXTPOSITION

TEXTPOSITION *text, findText*

Returns the 0-based index of first character of *findText* in *text*.

text - variable or text string
findText - variable or text string to look for

Example:

```
ASSIGN @temp = "23.45678"
// find the zero-based index of the decimal point
ASSIGN @pos = TEXTPOSITION @temp, "."
// @pos will equal 2
```

SUBTEXT

SUBTEXT *text, startIndex, length*

Returns a substring of *text* starting at the zero-based *startIndex* and continuing for *length* characters.

text - variable or text string
startIndex - variable or number indicating the zero-based index to start getting characters
length - variable or number indicating how many characters to take;

Example:

```
ASSIGN @newValue = SUBTEXT "CUE 1.0", 4, 3
// @newValue will be "1.0"
```

Application Statements

EXIT

EXIT [OVERRIDE | SHUTDOWN | REBOOT]

Causes SFX to exit.

OVERRIDE - will not prompt if changes have been made; just closes.
SHUTDOWN - will not prompt if changes have been made and also shuts down computer
REBOOT - will not prompt if changes have been made and also restarts computer

Example:

```
EXIT SHUTDOWN
```

NEW

NEW [PRODUCTION] [*filename*]

filename - full path name

Creates a new Production with the name of *filename* if supplied. If *filename* is not supplied the New Production dialog will appear.

OPEN

OPEN [PRODUCTION] [*filename*] [SCRIPT *script*]

filename - full path name
script - script name

Opens a Production with specified *filename* and if a *script* is specified, executes that script once open. If no production file name is specified, displays the Open Production dialog box.

SAVE

SAVE [PRODUCTION] [AS *filename*]

Saves the current production. If AS *filename* is specified it will save the production under the new name, and if not specified then SFX will display the Save As dialog box.

CLOSE PRODUCTION

CLOSE PRODUCTION

Closes the production.

NEW LIBRARY

NEW LIBRARY [*filename*]

filename - full path name

Create a new Effects Library with the name of *filename* if supplied. If *filename* is not supplied, the New Effects Library dialog will appear.

OPEN LIBRARY

OPEN LIBRARY [*filename*]

filename - full path name

Opens the Effects Library with the file name of *filename*. If no *filename* is specified, the Open Effect Library dialog box will appear.

SAVE LIBRARY

SAVE LIBRARY [AS *filename*]

Saves the current Effects Library. If AS *filename* is specified it will save the Effects Library under the new name, and if not specified then SFX will display the Save As dialog box.

CLOSE LIBRARY

CLOSE LIBRARY

Closes the Effects Library.

NEW CUES

NEW|ADD CUES [[EFFECT] LIST *listName*]

Creates a new Cue List.

REMOVE CUES

REMOVE|DELETE CUES [[EFFECT] LIST *listName*]

Removes that active Cue List or *listName* if specified.

EXPORT

EXPORT [EFFECT] LIST *listName* [TO *filename*]

Exports a Cue List to an export file. Will prompt for file unless *filename* is specified.

IMPORT

IMPORT [EFFECT] LIST *listName* [FROM *filename*]

Imports a Cue List from a previously exported cue list. Will prompt for file unless *filename* is specified.

PRINT

Opens Print dialog box.

PRINT PREVIEW

Opens Print Preview dialog box.

TOOLBAR

TOOLBAR STATUSTOOLBAR | COMMANDINTERFACE | EFFECTSTOOLBOX | GOBUTTON | TRANSPORT | ACTIVEMATRIX ON|OFF

Shows or hides the specified toolbar.

Cue List Statements

ACTIVATE [EFFECT] LIST

ACTIVATE [[EFFECT] LIST *name]*

name - the friendly name of the cue list

Brings Cue List *name* to the front and makes it active. Returns the name of active list. If no List name is specified, simply returns the name of the current active list.

Example:

```
ACTIVATE [EFFECT] LIST "B"
ASSIGN @active = ACTIVATE
```

COUNT LISTS

COUNT [EFFECT] LISTS

Returns the number of Cue Lists that are in the current production, or, if COUNT EFFECT LISTS is specified, the number of Effect Lists in the Effect Library.

Example:

```
ASSIGN @numLists = COUNT LISTS
```

COUNT CUES

COUNT [[EFFECT] LIST *listName*]

Returns the number of Cues in a *listName.*

Example:

```
ASSIGN @numCues = COUNT LIST "B"
```

COUNT SELECTED CUES

COUNT SELECTED [LIST *listName*]

Returns the number of slected cues (cues that are highlighted orange) in a *listName.*

Example:

```
ASSIGN @numSelectedCues = COUNT SELECTED LIST "B"
```

FOREACH...PRODUCTION

FOREACH *variable* IN PRODUCTION DO
   statement(s)...
LOOP

Loops through each Cue List in the production, assigning the Cue List's name to *variable*

Example:

```
FOREACH @cueList IN PRODUCTION DO
    DISPLAY "Cue ListL: + @CueList
LOOP
```

FOREACH...LIBRARY

FOREACH *variable* IN LIBRARY DO
  statement(s)...
LOOP

Loops through each Effect List in the Effect List Library, assigning the Effect List's name to *variable*

Example:

FOREACH @cueList IN LIBRARY DO
   DISPLAY "Effect List: + @CueList
LOOP

FOREACH...[EFFECT] LIST

FOREACH *variable* IN [EFFECT] LIST *listName* DO
  statement(s)...
LOOP

Loops through each cue in the cue list *listName*, or, if *listName* is empty, then the currently active cue list. *variable* is assigned the index of the cue.

FOREACH SELECTED...LIST

FOREACH SELECTED *variable* IN LIST *listName* DO
  statement(s)...
LOOP

Loops through each selected cue (i.e. cue that is highlighted orage) in the cue list *listName*, or, if *listName* is empty, then the currently active cue list. *variable* is assigned the index of the cue.

FOREACH...EFFECT [EFFECT] LIST

FOREACH *variable* IN EFFECT [EFFECT] LIST *listName* DO
  statement(s)...
LOOP

Loops through each cue in the Effect List *listName*, or, if *listName* is empty, then the currently active effect list. *variable* is assigned the index of the cue.

GET...PROPERTY
SET...PROPERTY

GET [[EFFECT] LIST *listName*] PROPERTY *listProperty*
SET [[EFFECT] LIST *listName*] PROPERTY *listProperty = propertyValue*

PRODUCTION_AUTHOR
PRODUCTION_DESCRIPTION
PRODUCTION_NOTES
PRODUCTION_DATECREATED
PRODUCTION_DATEMODIFIED
PRODUCTION_DEFAULTFADE
PRODUCTION_DEFAULTWAIT
PRODUCTION_FILENAME
PRODUCTION_MSCENABLED
PRODUCTION_REVISION
PRODUCTION_VERSIONCREATED

PRODUCTION_VERSIONMODIFIED
LIST_TITLE
LIST_ID
LIST_AUTHOR
LIST_DATECREATED
LIST_DATEMODIFIED
LIST_VERSIONCREATED
LIST_VERSIONMODIFIED
LIST_TRIGGERSENABLED
LIST_REVISION
LIST_DESCRIPTION
LIST_NOTES
LIST_TC_TIMECODE
LIST_TC_MIDIOUTPUT
LIST_TC_MIDIINPUT
LIST_TC_FREERUN
LIST_TC_STOPONLOSS
LIST_TC_FORMAT

Examples:
ASSIGN @auth = GET LIST "A" PROPERTY LIST_AUTHOR
SET PROPERTY LIST_NOTES = "Example Show"

START TIMECODE [LIST *listName*]
STOP TIMECODE [LIST *listName*]

Starts/stops time code in List *listName*.

CUELIST TRIGGERS

CUELIST [LIST *listName*] TRIGGERS ACTIVE | INACTIVE

listName - friendly name of cue list

Enables/disables Triggers for list *listName*, or the active list if not specified.

Example:
CUELIST TRIGGERS INACTIVE

CUELIST TOOLBAR

CUELIST [LIST *listName*] TOOLBAR TRANSPORT_TOOLBAR|NOTES_TOOLBAR ON|OFF

listName - friendly name of cue list

Displays/hides the specified cuel list.

Example:
CUELIST LIST "C" TOOLBAR NOTES_TOOLBAR OFF
CUELIST TOOLBAR TRANSPORT_TOOLBAR ON

Cue Statements

UNDO

UNDO [[EFFECT] LIST *name*]

name - the friendly name of the cue list

Performs an Undo in the list *name* or the active Cue List if [EFFECT] LIST *name* is not specified.

Example:

UNDO [EFFECT] LIST "A"
UNDO

REDO

REDO [[EFFECT] LIST *name*]

name - the friendly name of the cue list

Performs a Redo in the list *name* or the active Cue List if [EFFECT] LIST *name* is not specified.

Example:

REDO

CLEAR UNDO

CLEAR UNDO [[EFFECT] LIST *name*]

Peforms a Clear Undo in the list *name* or the active Cue List if [EFFECT] LIST *name* is not specified.

Example:

CLEAR UNDO [EFFECT] LIST "B"

COPY

COPY [[EFFECT] LIST *listName1*] [CUE|Q|INDEX *value1*] TO [[EFFECT] LIST *listName2*] [CUE|Q|INDEX *value2*]

Copies the *listName1 value1* cue to the new *listName2 value2* cue.

Example:

COPY INDEX 0 to CUE 5.5
COPY Q 5.5 to Q 10

DELETE

DELETE|REMOVE [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]

Deletes target cue.

Example:

DELETE LIST "A" CUE 4

INSERT

INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] AUTOFOLLOW

INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] FILE|SOUND|MIDISEQUENCE *filename*

Inserts the file with the path of *filename* into the list at *listName value.*

INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] TRACK *track* FILE *filename*

Insertsa file before track number *track* in an already existing sound cue.

INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] TRACK *track* TARGET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]

Insertsinto an already existing Stop of Volumn Change cue another targetted cue.

Insertsa file before track number *track* in an already existing sound cue.

INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] WAIT *seconds*

Insertsa Wait with a time of *seconds.*

INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] MEMO *description*

Insertsa Memo with the specified *description*.

INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] VOLUME [[EFFECT] LISTlistName2] [CUE|Q|INDEX *value2*]
INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] STOP [[EFFECT] LISTlistName2] [CUE|Q|INDEX *value2*]

Insertsa Volume change or Stop that targets listName2 *value2*

INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] MIDICMD *value*

The value should be in the format "time command data data;". For example:

INSERTMIDICMD "00:00:01.000 90 12 34"
or

INSERTMIDICMD "00:00:01.000 90 12 34;00:00:04.000 80 AA BB"

Time can be in format of
00:00:00.000 where the .000 is Milliseconds
or
00:00:00

INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] MSC *value*

Insertsa MIDI Show Control cue where *value* is the MSC values in ASCII format. For example:
INSERTCUE 10 MSC "F0 7F 01 02 01 01 31 00 00 F7"

INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] SCRIPT *scriptname*

Inserta Script cue with a call to the script with the name *scriptname*.

INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] TELNET *port, data*
INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] SERIALCMD *port, data*

Insertsa TELNET or Serial cue that ouputs out of *port* with the data of *data*.

INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] COMMAND *command*
INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] COMMAND *command* TARGET [[EFFECT] LIST *listNameTarget*] [CUE|Q|INDEX *valueTarget*]
INSERT[[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] COMMAND *command* TARGET [[EFFECT] LIST *listNameTarget*] [CUE|Q|INDEX *valueTarget*] TIMECODE = *timevalue*

Insertsa command cue into list. *listName* and *value* are the location to insertthe cue. If the command cue requires a target cue, it is specified as *listNameTarget* and *valueTarget*. If the Command cue requires timecode, it as specified as *timevalue* and is in the form of "00:12:23.99".

RENUMBER

RENUMBER [[EFFECT] LIST *listName*]

listName - friendly name of cue list

Renumbers cue list.

LEARN TIMECODE

LEARN TIMECODE [LIST *listName*] ON|OFF

listName - friendly name of cue list

Toggles the timecode learning for cue list *listName* or the current list if *listName* is not specified.

COLLAPSE

COLLAPSE [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]

listName - friendly name of cue list
value - either cue number or zero-based index of cue

For a cue with multiparts, hides the sub-parts.

EXPAND

EXPAND [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]

listName - friendly name of cue list
value - either cue number or zero-based index of cue

For a cue with multi-parts, shows the subparts.

GET STANDBY

GET STANDBY [[EFFECT] LIST *listName*]

listName - friendly name of cue list
value - either cue number or zero-based index of cue

Returns the index of the cue currently in Standby (i.e. selected with green bar).

GET
SET

GET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] *propertyName*
SET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] *propertyName = newValue*

listName - friendly name of cue list
value - either cue number or zero-based index of cue
propertyName -

EXISTS
TYPE
DESCRIPTION
CUENUMBER
RANK
CUE_INDEX
DURATION
TIME_CODE
ERRORCODE
ERRORMESSAGE
STATUS
DURATIONENTIRE
TIMEELAPSED
TIMEELAPSEDENTIRE
NOTES
ACT
PAGE
SCENE
KEYSTROKE_KEY
e.g. "F12" or "Ctrl+A"
MIDI_CMD
e.g. "C0,2,12,120" // Program Change, Channel 2, Data 12, velocity 120
e.g. "90,1,10,Any" // Note On, Channel 1, Data 10 with any velocity.
TRACKS
RECORDID
Special (see below):
CROSSPOINT
VOLUME
FADECURVE
LOOPCOUNT
FILENAME

GET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] **CROSSPOINT** TRACK *track* BUSS *buss*
SET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] **CROSSPOINT** TRACK *track* BUSS *buss* = ACTIVE|INACTIVE

listName - friendly name of cue list
track - number or variable representing the number of the track
buss - text or variable representing the name the buss

Gets/sets whether the cue is ACTIVE or INACTIVE.

GET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] **VOLUME** TRACK *track* BUSS *buss*
SET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] **VOLUME** TRACK *track* BUSS *buss = level*

listName - friendly name of cue list
track - number or variable representing the number of the track
buss - text or variable representing the name the buss
value - volume level as a number or variable

Gets/sets the volume level of the specified cue.

GET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] **FADECURVE** TRACK *track* BUSS *buss*
SET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] **FADECURVE** TRACK *track* BUSS *buss = curve*

listName - friendly name of cue list
track - number or variable representing the number of the track
buss - text or variable representing the name the buss
curve - curve name

Gets/sets the name of the fade curve of of the specified cue.

GET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] **LOOPCOUNT** TRACK *track*
SET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] **LOOPCOUNT** TRACK *track = value*

listName - friendly name of cue list
track - text or variable representing the number of the track
value - number or variable of the number of loops

Gets/sets the number of loops for the specified cue.

GET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] **FILENAME** TRACK *track*
SET [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] **FILENAME** TRACK *track = value*

listName - friendly name of cue list
track - text or variable representing the number of the track
value - text or variable of the full filename of the sound file

Gets/sets the full file name of a track.

Transport Statements

FIRE

FIRE KEYBOARD|MIDI|SFX_TRANSPORT *data*

data - the name of the trigger

Example:

FIRE KEYBOARD "F9"

Individual Cues:

PRECUE|PRELOAD [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]
GO [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]
PLAY [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]
PAUSE [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]
STOP [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]
STOPALL|**PANIC** [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]
SELECT [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]
STANDBY [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]
GOTO [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]
FIRST [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]
LAST [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]
PREVIOUS [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]

NEXT [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*]
SEEK [[EFFECT] LIST *listName*] [CUE|Q|INDEX *value*] TO *time*

GET TIMECODE [[EFFECT] LIST *listName*]
SET TIMECODE [[EFFECT] LIST *listName*] = *time*

listName - friendly name of cue list
value - either cue number or zero-based index of cue
time - time value like 00:01:23.0

Examples:

GO
GO LIST "C" Q 4
FIRST LIST "A"
STANDBY Q 3
SEEK Q 4 TO 01:23:00.00

Layout Statements

SWITCH TO LAYOUT

SWITCH TO LAYOUT *layout*

layout = name of layout

Switches the layout to layout with the name of *layout*.

Example:

SWITCH TO LAYOUT "Layout 02"

CREATE LAYOUT

CREATE LAYOUT *layout*

layout = name of layout

Creates a layout with the name of *layout*.

GET LAYOUT

GET LAYOUT *layout attribute*

layout - name of layout
attriute - one of the following: THEME, SHORTCUT, FULLSCREEN, HIDETITLES, HIDEMAINMENU, LAYOUTLOCKED, READONLY, PASSWORD, MERGECUESMENU

Returns the specified layout attribute.

Example:

SET LAYOUT

SET LAYOUT *layout attribute = value*

layout - name of layout
attribute - one of the following: THEME, SHORTCUT, FULLSCREEN, HIDETITLES, HIDEMAINMENU, LAYOUTLOCKED, READONLY, PASSWORD, MERGECUESMENU
value -the value of the item

Sets the specified layout attribute.

Example:

Script and Trigger Statements

COUNT SCRIPTS

COUNT SCRIPTS

Returns the number of scripts in the production.

Example:

ASSIGN @num = COUNT SCRIPTS

ADD SCRIPT

ADD SCRIPT *name*, *script*

name - friendly name of script
script - the script commands

Adds a script with a friendly name of name that contains the script commands commands.

Example:

ADD SCRIPT "Script 1", "GO CUE 5"

REMOVE SCRIPT

REMOVE {SCRIPT *name* | INDEX *idx*}

name - friendly name of script.
idx - zero-based index of script

Removes the script with either the friendly name of name, or the index of idx.

Example:

REMOVE SCRIPT "Telnet 04"
REMOVE SCRIPT INDEX 2

COUNT TRIGGERS

COUNT TRIGGERS

Returns the number of triggers in the production.

Example:

ASSIGN @num = COUNT TRIGGERS

ADD TRIGGER

ADD TRIGGER KEYBOARD|MIDI|SFX_TRANSPORT = *trigger* SCRIPT = *scriptName*

trigger - the actual trigger
scriptName - name of script

Adds a trigger of a certain type that will execute *scriptName* when *trigger* is received.

Example:

ADD TRIGGER KEYBOARD = "F2" SCRIPT = "Script 1"

DELETE TRIGGER

DELETE TRIGGER INDEX *idx*

idx = zero-based index of trigger to delete

Example:

DELETE TRIGGER INDEX 1

GET TRIGGER

GET TRIGGERENABLED|TRIGGERTYPE|TRIGGERVALUE|TRIGGERSCRIPT INDEX *idx*

idx - zero-based index of trigger to delete

Example:

ASSIGN @type = GET TRIGGERTYPE INDEX 1

SET TRIGGER

SET TRIGGERENABLED|TRIGGERTYPE|TRIGGERVALUE|TRIGGERSCRIPT = *value* INDEX *idx*

value - value for the parameter
idx - zero-based index of trigger to delete

Example:

SET TRIGGERENABLED = "ACTIVE" INDEX 3
SET TRIGGERVALUE = "90,1,10,Any" // Note On, Channel 1, Data 10 with any velocity.

Patch Statements

CREATE AUDIOPATCH

CREATE AUDIOPATCH *name*

name - the name of the Audio Patch to create

DESTROY AUDIOPATCH

DESTROY AUDIOPATCH *name*

name - the name of the Audio Patch to remove

PATCH AUDIO

PATCH AUDIO TO *name*

name - the name of the Audio Patch to switch to.

COUNT AUDIOPATCH

COUNT AUDIOPATCH OUTPUT ROWS|COLUMNS

Counts the number of ROWS or COLUMNS in the audio patch.

Example:

ASSIGN @numRow = COUNT AUDIOPATCH OUTPUT ROWS
ASSIGN @numColumns = COUNT AUDIOPATCH OUTPUT COLUMNS

ADD AUDIOPATCH

ADD AUDIOPATCH OUTPUT [*buss*]

buss - a variable or text string of the name of the buss

Adds a new Buss to the current audio patch and names it *buss* if *buss* is specified.

DELETE AUDIOPATCH

DELETE AUDIOPATCH Input|Output *buss*

buss - a variable or text string of the name of the buss

Deletes the buss with the name of *buss* from the current audio patch.

GET AUDIOPATCH VOLUME

GET AUDIOPATCH OUTPUT VOLUME *buss*, GAIN|*column*

buss - a variable or text string of the name of the buss
column - column number

Gets the volume level of the the current patch at buss under the column of number <Expression> or GAIN.

Example:

ASSIGN @vol = GET AUDIOPATCH OUTPUT VOLUME "Buss 01", 1

ASSIGN @volGain = GET AUDIOPATCH OUTPUT VOLUME "Buss 02", GAIN

SET AUDIOPATCH VOLUME

SET AUDIOPATCH OUTPUT VOLUME *buss*, GAIN|*column*= *level*

buss - a variable or text string of the name of the buss
column - column number
level - volume level

Sets the volume level of the the current patch at *buss* under the column of number *track* or GAIN with *level*.

Example:

SET AUDIOPATCH OUTPUT VOLUME "Buss 01", 1 = -5
SET AUDIOPATCH OUTPUT VOLUME "Buss 02", GAIN = -10

GET AUDIOPATCH CROSSPOINT

GET AUDIOPATCH OUTPUT CROSSPOINT *buss*, GAIN|*column*

Gets the active or inactive state of the the current patch at <BussName> under the column of number <Expression> or GAIN.

Example:

ASSIGN @vol = GET AUDIOPATCH OUTPUT CROSSPOINT "Buss 01", 1

SET AUDIOPATCH CROSSPOINT

SET AUDIOPATCH OUTPUT CROSSPOINT *buss*, GAIN|*column = active*

buss - a variable or text string of the name of the buss
column - column number
active - ACTIVE or INACTIVE

Sets the volume active or inactive state of the the current patch at *buss* under the column of number *column* or GAIN with ACTIVE or INACTIVE.

Example:

SET AUDIOPATCH OUTPUT CROSSPOINT "Buss 01", 1 = ACTIVE
SET AUDIOPATCH OUTPUT CROSSPOINT "Buss 02", 3 = INACTIVE

ADD MIDIPATCH

ADD MIDIPATCH INPUT|OUTPUT *name*, *device name*, ACTIVE|INACTIVE

name = text or variable that will serve as the name of the patch.
device name = text or variable that represents the physical name of the MIDI patch as reported by the computer.

Example:

ADD MIDIPATCH INPUT "Main MIDI", "AudioFire4 MIDI", ACTIVE

GET MIDIPATCH

GET MIDIPATCH *name*

name = Friendly name as text or variable

Returns whether the MIDI patch with the friendly name of *name* is ACTIVE or INACTIVE. Empty if name does not exist.

Example:

ASSIGN @state = GET MIDIPATCH "MIDI In 1"

SET MIDIPATCH

SET MIDIPATCH *name* = ACTIVE|INACTIVE

name = Friendly name as text or variable

Sets the MIDI patch with the friendly name of name to ACTIVE or INACTIVE.

Example:

SET MIDIPATCH "Midi Out 1" = INACTIVE

GET SERIALPATCH

GET SERIALPATCH *name*

name - Friendly name as text or variable

Returns whether the MIDI patch with the friendly name of *name* is ACTIVE or INACTIVE. Empty if name does not exist.

Example:

ASSIGN @state = GET SERIALPATCH "MIDI In 1"

SET SERIALPATCH

SET SERIALPATCH *name* = ACTIVE|INACTIVE

Sets the MIDI patch with the friendly name of *name* to ACTIVE or INACTIVE.

Example:

SET SERIALPATCH "Midi Out 1" = INACTIVE

ADD TELNETPATCH

ADD TELNETPATCH *name* AS IP *ipAddress*

name - text of variable that will serve as the name of the patch.
ipAddress = text or variable that represents the IP address. e.g. "192.168.1.100:3000"

Example:

ADD TELNETPATCH "Telnet 01" AS 192.168.0.20:123

GET TELNETPATCH

GET TELNETPATCH *name*

name - text of variable that will serve as the name of the patch.

Returns whether the Telnet patch with the friendly name of *name* is ACTIVE or INACTIVE. Empty if name does not exist.

Example:

ASSIGN @state = GET TELNETPATCH "Telnet 1"

SET TELNETPATCH

SET TELNETPATCH *name* = ACTIVE|INACTIVE

name - text of variable that will serve as the name of the patch.

Sets the Telnet patch with the friendly name of *name* to ACTIVE or INACTIVE.

Example:

SET TELNETPATCH "Telnet 1" = ACTIVE

DELETE TELNETPATCH

DELETE TELNETPATCH *name*

name - text of variable that will serve as the name of the patch.

Removes the Telnet patch with friendly name of *name*.

Example:

DELETE TELNETPATCH "Telnet 1"