1 Script Basics

This chapter explains the conventions your Apple Remote Access modem script must follow and describes the basic tasks it must perform. For detailed descriptions of the modem scripting commands, see Chapter 2, “CCL Commands.”

What is a modem script?

A modem script is a set of instructions that tells a computer how to interact with a modem so that calls can be initiated and received. To establish a connection, a script typically configures and then connects the modem. To terminate a connection, the script disconnects the modem by hanging up and then restores the modem settings that were in effect before the call.

Each type of modem used with Apple Remote Access requires a modem script. Many scripts are provided with Apple Remote Access (see "Available Modem Scripts" in the Preface of this book for more information).

If no script is provided for your modem, you must write one using Communication Command Language (CCL), a programming language that configures and controls your modem. This chapter describes the basic elements and structure of a CCL file.

Script conventions

The following sections describe CCL script conventions for

- instructions
- comments
- capitalization
- labels
- script resources
- string formats
- variable strings (varStrings)
- match strings
- script size

Instructions

Each line of CCL code consists of one instruction that is made up of a command and its parameters, if any. Modem commands are used as parameters of CCL commands. For example, the command `write "ATDT^1\13"` includes the following:

`write`	a CCL command
`ATDT`	a modem command
`^1\13`	a modem command parameter

This command tells the CCL interpreter to send to the modem the modem command `ATDT` followed by variable string 1, and a carriage return (ASCII code 13).

The CCL interpreter reads scripts from left to right and from top to bottom. It reads straight through, from beginning to end, unless you tell it otherwise (for example, by using the `JUMP` command).

Comments

You can insert explanatory comments into your script. To enter a comment, start the line with an exclamation point (`!`).

You may also want to use a blank comment line to make your script more readable; to do so, type an exclamation point with no text and press Return.

Capitalization

The CCL interpreter is not case-sensitive. Therefore,

```
@LABEL 1
@Label 1
@label 1
@laBel 1
```

are all valid instructions.

Labels

Labels are used to mark locations in the script. Other script commands, such as `JUMP`, transfer control to locations in the script marked by the `@LABEL` command. For instance, `JUMP 13` tells the CCL interpreter to jump to label 13 and start executing the commands after the `@LABEL 13` command. A script may use up to 128 labels, numbered 1 to 128.

Script resources

A modem script may contain an optional mlts resource with resource ID 0. This resource contains data that tells the CCL interpreter the characteristics of the modem. An mlts resource contains five unsigned bytes indicating the modem's error correction and command string limitations.

Byte 1 — Indicates the modem's error correction capability:
- 0 — The modem has no built-in error correction.
- 1 — The modem has at least one embedded standard error correction protocol, such as MNP or LAPM. (Do not set to this value if the modem has only proprietary protocols such as PEP or HST.)

Byte 2 — Reserved by Apple.

Byte 3 — Indicates the maximum length of the first dial string fragment (in varString7; for details, see "Variable Strings (varStrings)" later in this chapter) passed to the script by the CCL interpreter. Check your modem documentation to determine the maximum number of characters your modem can accept in an `AT` command.

Byte 4 — Indicates the maximum length of the second dial string fragment (in varString8) passed to the script by the CCL interpreter.

Byte 5 — Indicates the maximum length of the third dial string fragment (in varString9) passed to the script by the CCL interpreter.

If your script has no mlts resource (for example, if it is an Apple Remote Access 1.0 script), the CCL interpreter assumes a maximum length of 30 characters for each of the dial string fragments, and it assumes that the modem cannot negotiate a reliable link.

To create a script resource, follow the instructions in “Creating an mlts Resource” in the *Apple Remote Access Modem WorkShop User’s Guide*.

String formats

To delimit a string in CCL code, you can use single quotation marks (`'`) or double quotation marks (`"`). If you do not start the string with a single or double quotation mark, any of the following characters determines the end of the string: space, return, tab, comma, or semicolon.

CCL strings may include references to variable strings (varStrings). See “Variable Strings (varStrings)” later in this chapter for details.

CCL strings may include nonprinting characters such as null, tab, and return. To specify one of these characters in a string, precede it by an escape character. The CCL interpreter recognizes two escape characters: the backslash (`\`) and the caret (`^`).

The backslash is used to include a character by specifying the decimal representation of the ASCII character (decimal numbers 00 to 99 are valid), or to include the backslash or caret character in a string.

The caret is used to insert a variable string or an ASK string into another string. (For details, see “Variable Strings (varStrings)” later in this chapter.)

Here are some examples of how the backslash and caret are used:

`\13`	inserts a carriage return (0x0D) into the string
`\00`	inserts the null character (0x00) into the string
`\\`	inserts the backslash (\) character (0x5C) into the string
`\^`	inserts the caret (^) character (0x5E) into the string
`^1`	inserts variable string 1 into the string
`^*`	inserts the ASK string into the string

Here are some string examples:

`'this is a test'`	yields	`this is a test`
`thisisatest`	yields	`thisisatest`
`"this is a test"`	yields	`this is a test`
`this is a test`	yields	`this`
`"this \34\73\83\34 a test"`	yields	`this "IS" a test`

If variable string 1 is 555-1212, then

`"ATDT^1"`	yields	`ATDT555-1212`

Variable strings (varStrings)

CCL strings may include references to variable strings (varStrings), which are strings passed to the script as parameters. VarStrings 1– 4 and 6 are defined on the Script Setup card, as described in “Testing a Script” in the *Apple Remote Access Modem WorkShop User’s Guide*. VarStrings 7–9 are defined by the CCL interpreter based on varString1 and bytes 3–5 of the mlts resource. VarString5 is reserved for future use by Apple.

Apple Remote Access uses the following varStrings:

varString1 — the complete dial string (the telephone number to dial, with the necessary prefixes and suffixes)

varString2 — the modem speaker flag
- 0 speaker off
- 1 speaker on

varString3 — the tone/pulse dialing selector
- T tone dialing
- P pulse dialing

varString4 — the modem error correction flag
- 0 script should not try to use modem error correction
- 1 script should try to use modem error correction
- 2 script should try to establish MNP10 error correction

varString6 — the dialing mode flag

- 0 normal dialing
- 1 blind dialing (that is, have the modem begin dialing without confirmation of dial tone detection)
- 2 manual dialing (that is, have the script assume that the user has already picked up the phone and dialed the remote number)

varStrings7–9 — varStrings 7–9 break a long dial string into shorter strings for use with modems that can accept only a limited command string size. They are automatically generated by Apple Remote Access.

varString7 contains the first characters of varString1. Its maximum length is defined by byte 3 of the mlts resource (see “Script Resources” earlier in this chapter).

If varString7 contains all the characters included in varString1, varString8 contains a blank string (ASCII 20 hex). If varString1 includes more characters than can be contained in varString7, varString8 contains some or all of the additional characters. Its maximum length is defined by byte 4 of the mlts resource.

If varString7 and varString8 contain all the characters included in varString1, varString9 contains a blank string. If varString1 includes more characters than can be contained in varString7 and varString8, varString9 contains some or all of the additional characters. Its maximum length is defined by byte 5 of the mlts resource.

Note: If you do not create an mlts resource for your script (as described in “Creating an mlts Resource” in the *Apple Remote Access Modem WorkShop User’s Guide*), the CCL interpreter assumes a maximum length of 30 characters for each of the dial string fragments.

All eight varStrings are passed to the script when it is running in originate mode. The modem speaker flag (varString2) and the error correction flag (varString4) are passed to the script when it is running in answer mode.

Match strings

The CCL interpreter has a buffer that can hold up to 32 strings loaded by the `MATCHSTR` command. The `MATCHCLR` command erases the contents of the buffer. The `MATCHREAD` command reads input from the serial driver and compares the input to the strings currently in the buffer. If a match is found in the specified time, execution continues at the label associated with that match string.

A recommended strategy for sending commands to the modem is as follows:

- Use `MATCHCLR` to clear all match strings.
- Use `MATCHSTR` to load match strings with appropriate responses.
- Use `WRITE` to send commands to the modem.
- Use `MATCHREAD` to check for defined modem responses. If an appropriate response is received, branch as defined by the corresponding `MATCHSTR` command.
- Use `SETTRIES`, `INCTRIES`, and `IFTRIES` to loop a few times.
- If an appropriate response is not received, branch to exit or to alternate code as defined by the `MATCHREAD` command.

Script size

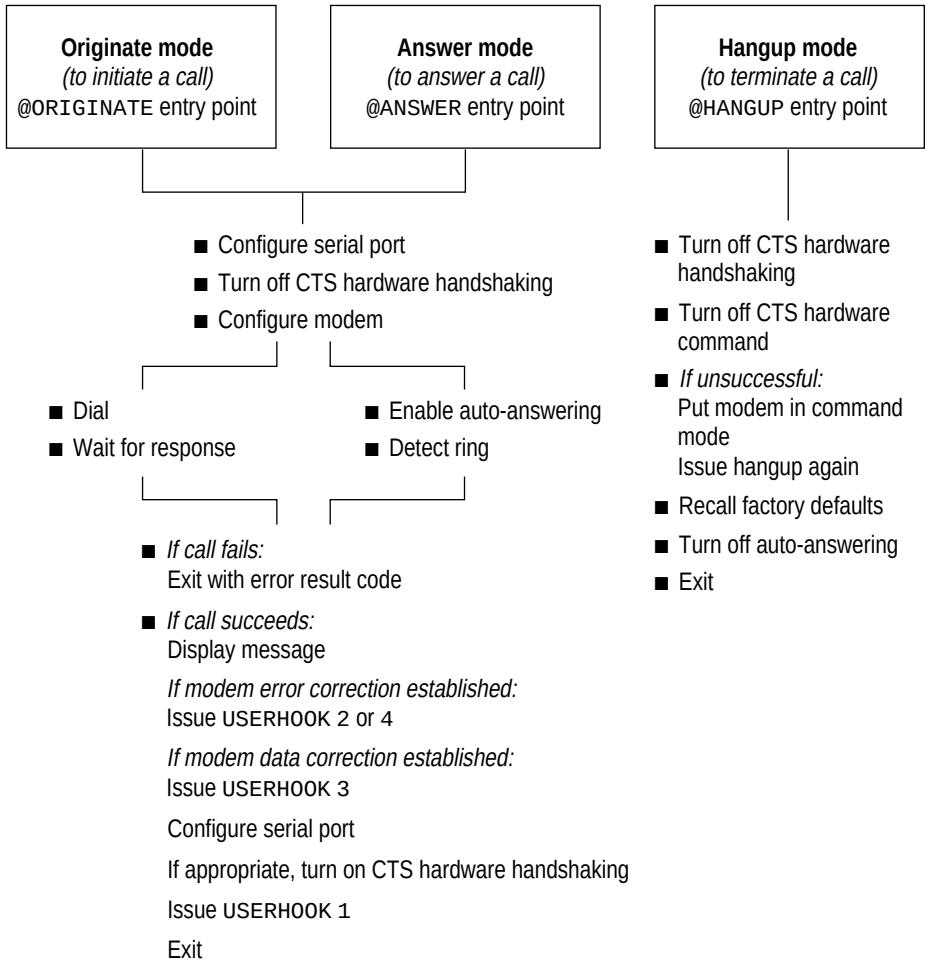
Scripts used in Modem WorkShop may be a maximum of 30 kilobytes. (You can write quite a detailed script in 30 kilobytes.) If your script is too large, you may be able to make it small enough by minimizing the number and length of comments.

Tasks your script must perform

A modem script executes in one of three possible modes, each with a separate entry point. The modes are as follows:

originate	used when a call is initiated
answer	used when call answering is enabled
hangup	used to terminate every connection, whether in originate or answer mode, and whether or not the connection was successful

The following figure provides an overview of the tasks your script must perform in each mode. The remainder of this chapter describes these tasks.



Initiating a call

When you signal the Apple Remote Access program to initiate a call, it plays the script starting at the `@ORIGINATE` entry point. The script must perform the following tasks to initiate a call:

1. **Configure your computer's serial port.**

   Use the `SERRESET` command to reset the serial port's communication parameters. Set the speed of the connection to the maximum speed of the modem. (You may change the serial port speed later in the script.) Set the number of bits to be used for stop, start, and parity.

   Use the `HSRESET` command to turn off the serial port's flow control options. (You will turn on the appropriate options later in the script.)

2. **Configure the modem.**

   Following is a list of standard steps for configuring a modem. Check the documentation that came with your modem to determine whether it requires different steps.

   a. Recall the factory default configuration settings.

   b. If your modem has a speed of 14,400 bps or higher, you need to set up the modem for CTS/RTS handshaking and an appropriate cable must be used, as described in Appendix B, "Cable Requirements." Do not turn on hardware handshaking until the connection has been made.

   c. Configure the modem for DTR usage.

   d. Turn local echo off.

      When local echo is on, the modem sends commands it receives back to the computer.

   e. Set the modem to return detailed result codes including the speed of the connection and the results of error correction and data compression negotiation.

   f. If the modem can do error correction, set error correction according to varString4.

      If varString4 is set to 0, turn error correction off.

      If varString4 is set to 1, have the modem negotiate the best available error correction with the remote modem. If no error correction can be established, have the modems remain connected without error correction.

If varString4 is set to 2, have the modem try to establish MNP10 error correction with the remote modem. If MNP10 negotiation fails, have the modems remain connected without error correction.

Note: If the modem uses a proprietary error correction protocol, make sure that it will try to negotiate standard protocols if it is unable to negotiate the proprietary protocol. If not, disable error correction.

Note: Do not disable Trellis error protocol, which is part of the V.32 standard.

g. Apple Remote Access is generally more efficient than a modem at compressing data. If you believe you have a special situation in which modem data compression is preferable, have the script set up the modem to negotiate data compression.

h. Turn the modem speaker on or off according to the value of varString2.

3 Dial the phone number.

a. Have the script check whether the dial string extends into varString8 (then varString9) by using the `IFSTR` command to check for a blank string. If the entire dial string fits in varString7, have the script issue a single dial command. If the dial string is longer than varString7, have the script issue multiple modem dial commands referencing varStrings 7 and 8, and if necessary, varString9.

b. Set tone or pulse dialing in the modem dial command according to the value of varString3.

c. If varString6 is set to 1, have the modem begin dialing without confirmation of dial tone detection. This is useful when the phone system provides a nonstandard dial tone that can't be recognized by the modem's tone detection circuitry.

If varString6 is set to 2, the user will already have dialed the remote number. Have the script cause the modem to retrain with the remote modem. This is useful when the dialing sequence is too complex or interactive for the modem script to navigate.

d. Display the dialed phone number in the Apple Remote Access Status window and the Activity Log. Use varString1 in log messages rather than a concatenation of varStrings 7–9.

4 Wait for the modem response.

5 **If the call fails, return an error result code indicating what happened.**

For example, use error result code -6022 if the line is busy. (See Appendix A, “Result Codes,” for a complete list.)

6 **If the call is successful, indicate that a connection has been established.**

Display a message such as “Communicating at 9600 bps” in the Apple Remote Access Status window. Also display messages indicating the results of error correction and data compression negotiation, if applicable.

If you were successful in establishing modem error correction, issue `USERHOOK 4` (for MNP10 error correction) or `USERHOOK 2` (for all other error correction) to advise Apple Remote Access that a reliable link was established.

If you were successful in establishing data compression, issue `USERHOOK 3` to advise Apple Remote Access to turn off its built-in data compression.

7 **Configure the serial port based on the modem speed and the connection.**

If your modem has a speed of 14,400 bps or higher and if you are using a CTS/RTS handshaking cable (as described in Appendix B, “Cable Requirements”), use the `HSRESET` command to set flow control for `outputCTS`. Use the `COMMUNICATINGAT` command to tell Apple Remote Access the connection speed so that it can set its timers appropriately.

If your modem has a speed of 9600 bps or slower, use the `SERRESET` command to reset the serial port speed to the speed of the connection.

8 **Exit the script so that the Apple Remote Access program can continue with its internal execution.**

Answering a call

To answer a call, the Apple Remote Access program plays the script starting at the `@ANSWER` entry point. The script must perform the following tasks to answer a call:

1 **Configure your computer's serial port.**

Use the `SERRESET` command to reset the serial port's communication parameters. Set the speed of the connection to the maximum speed of the modem. (You may change the serial port speed later in the script.) Set the number of bits to be used for stop, start, and parity.

Use the `HSRESET` command to turn off the serial port's flow control options. (You will turn on the appropriate options later in the script.)

2 **Configure the modem.**

It will require several modem commands to completely configure the modem. To prevent calls being answered before the configuration is complete, disable auto-answering in the first modem command the script issues. (You will enable it in step 3.)

For additional details about configuring the modem, see step 2 of "Initiating a Call" earlier in this chapter.

a. Recall the factory default configuration settings.

b. If your modem has a speed of 14,400 bps or higher, set up the modem for CTS/RTS handshaking and use an appropriate cable, as described in Appendix B, "Cable Requirements."

c. Configure the modem for DTR usage.

d. Turn local echo off.

e. Set the modem to return detailed result codes including the speed of the connection and the results of error correction and data compression negotiation.

f. If the modem can do error correction, set error correction according to varString4.

g. Apple Remote Access is generally more efficient than a modem at compressing data. If you believe you have a special situation in which modem data compression is preferable, have the script set up the modem to negotiate data compression.

h. Turn the modem speaker on or off according to the value of varString2.

3 **Enable auto-answering and wait for the result.**

On an incoming call, the modem issues a `RING` result code.

4 **If the call fails, return an error result code indicating what happened.**

For example, use error result code -6021 if the modem cannot detect a carrier signal on the phone line. (See Appendix A, “Result Codes,” for a complete list.)

5 **If the call is successful, indicate that a connection has been established.**

Display a message such as “Communicating at 9600 bps” in the Apple Remote Access Status window. Also display messages indicating the results of error correction and data compression negotiation, if applicable.

If you were successful in establishing modem error correction, issue `USERHOOK 4` (for MNP10 error correction) or `USERHOOK 2` (for all other error correction) to advise Apple Remote Access that a reliable link was established.

If you were successful in establishing data compression, issue `USERHOOK 3` to advise Apple Remote Access to turn off its built-in data compression.

6 **Configure the serial port based on the modem speed and the connection.**

Issue the `USERHOOK 1` command. The `USERHOOK 1` command informs Apple Remote Access that the serial port is busy answering a call, which prevents the serial port from being given up to another application.

If your modem has a speed of 14,400 bps or higher and if you are using a CTS/RTS handshaking cable (as described in Appendix B, “Cable Requirements”), use the `HSRESET` command to set flow control for `outputCTS`.

Use the `COMMUNICATINGAT` command to tell Apple Remote Access the connection speed so that it can set its timers appropriately.

If your modem has a speed of 9600 bps or slower, use the `SERRESET` command to reset the serial port speed to the speed of the connection.

7 **Exit the script so that the Apple Remote Access program can continue with its internal execution.**

Terminating a call

To terminate a call, the Apple Remote Access program plays the script starting at the `@HANGUP` entry point.

The hangup part of the script is played to terminate a connection whenever the `@ORIGINATE` or `@ANSWER` parts of the script have been played, regardless of the result. The hangup part of the script is also played when Apple Remote Access terminates answer mode because another application gains control of the serial driver or the system is shut down.

Note: Apple Remote Access has a callback feature (see the *Apple Remote Access Personal Server User's Guide* or the *Apple Remote Access Client User's Guide* for more information). For callback to work, the script must hang up the modem and prepare it for the return call within 15 seconds, which is the minimum time that the answering side will wait before issuing the callback. The maximum time the *calling side* (that is, your computer) will wait for a callback is 80 seconds.

The script must perform the following tasks to terminate a call:

1. **If hardware handshaking is used in the `@ORIGINATE` or `@ANSWER` part of the script, turn off hardware handshaking.**

   Use the `HSRESET` command to turn off hardware handshaking.

2. **If possible, cause the modem to enter command mode.**

   Before you issue a hangup command, you may need to get the attention of the modem by, for example, issuing a short break, a long break, or the attention sequence "+++," or by toggling DTR. Consult your modem documentation for the appropriate method.

3. **Issue a modem hangup command.**

4. **Recall the factory default configuration settings.**

   Since you recalled the default settings at the beginning of the script, this is not necessary if the only communications application you use is Apple Remote Access; however, recalling the default settings at the end of your script is recommended in case the next communications application that you use does not take care of this itself.

5 **Turn off auto-answering.**

This prevents the modem from answering the phone until call answering is enabled.

6 **Exit with an appropriate message.**

If successful, return a result code of 0. If unsuccessful, return the appropriate error result code as listed in Appendix A, “Result Codes.”

2 CCL Commands

This chapter describes the CCL commands interpreted by the Apple Remote Access program. The commands are presented in alphabetical order. Each command section contains a description of the command; the syntax of the command, including any parameters; and an example, if appropriate.

! Comment

To insert a comment or a blank line in the script, start the line with an exclamation point.

Scripts used in Modem WorkShop may be a maximum of 30 kilobytes. If your script is too large, you may be able to make it small enough by minimizing the number and length of comments.

Syntax `! comment`

Examples

```
! Turn echo off
!
```

@ANSWER

`@ANSWER` marks the script entry point when the script plays in answer mode.

Syntax `@ANSWER`

@HANGUP

`@HANGUP` marks the script entry point when the script plays in hangup mode.

Syntax `@HANGUP`

@LABEL

`@LABEL` sets a numeric label in the script that can then be referenced from other script commands, such as `JUMP`, `JSR`, and `IFTRIES`. A script may include up to 128 labels, numbered 1 through 128.

To make debugging easier, assign the labels in ascending sequence. They don't need to be consecutive.

Syntax `@LABEL labelnum`

Parameter *labelnum* a value from 1 through 128 that specifies the label number

Example `@LABEL 30`

@ORIGINATE

`@ORIGINATE` marks the script entry point when the script plays in originate mode (that is, when initiating a call).

Syntax `@ORIGINATE`

ASK

`ASK` causes a dialog box to be displayed to obtain information from the user. The dialog box contains a prompt message, an optional data entry field, a Cancel button, and an OK button. You may need the ASK command if your telephone system uses special telecommunications equipment. This command is typically used in originate mode only.

"String Formats" in Chapter 1 shows how to use the `ASK` string as part of another string. The ASK string is set if the user presses either the OK button or the Cancel button.

Syntax `ASK maskflag "message" [jumpLabel]`

Parameters

maskflag	0 to echo the user's input 1 to mask the user's input with bullets (••••) 2 to not allow user input
message	the string to display in the dialog box as a prompt for the user
jumpLabel	if supplied, the label to jump to, where execution continues when the Cancel button is pressed; if not supplied, or if the OK button is pressed, then execution continues at the next CCL line

Example

```
ASK 1 "Enter your password to access the network."
ASK 2 "When the remote modem answers, click OK, otherwise click Cancel to stop Manual Dialing."
```

CHRDELAY

`CHRDELAY` allows you to specify a delay time between characters for all subsequent `WRITE` commands. This is useful for telecommunications equipment that requires data at a speed slower than the interface speed.

Syntax `CHRDELAY` *x*

Parameter *x* the delay time, in tenths of a second

Example `CHRDELAY 8`

COMMUNICATINGAT

For V.32bis modems with CTS/RTS hardware flow control cables (as described in Appendix B, “Cable Requirements”) only: Use the `COMMUNICATINGAT` command to indicate the speed of the modem connection if the modem speed is different from the serial port speed. This is necessary because the Apple Remote Access application’s internal timers are based on the modem speed.

Syntax `COMMUNICATINGAT` *x*

Parameter *x* the modem speed, in bits per second

Example `COMMUNICATINGAT 4800`

DECTRIES

`DECTRIES` decreases the variable tryCounter by one. The CCL interpreter maintains tryCounter, which you may set to a value and increase or decrease by one. See also the commands `IFTRIES`, `INCTRIES`, and `SETTRIES`.

Syntax `DECTRIES`

DTRCLEAR

`DTRCLEAR` clears (that is, deasserts) the Data Terminal Ready (DTR) signal on the RS-232 interface.

Syntax `DTRCLEAR`

DTRSET

`DTRSET` sets (that is, asserts) the Data Terminal Ready (DTR) signal on the RS-232 interface.

Syntax `DTRSET`

EXIT

`EXIT` terminates execution of the script and returns a result code along with an optional string.

- If the script executes successfully, have it return a result code of 0.
- If the script fails for any reason, have it return the appropriate error result code, as listed in Appendix A, “Result Codes.”
- To give the user a nonstandard error message, use result code -6002 and use the string parameter to pass the nonstandard error message.

Syntax `EXIT result ["string"]`

Parameters

result	one of the CCL result codes listed in Appendix A, “Result Codes”
string	the message displayed to the user when a connection attempt fails; if you include a string for one of the standard result codes, it overrides the one that CCL would normally display

Examples

```
EXIT -6022
EXIT -6002 "unable to communicate
with PBX"
```

FLUSH

`FLUSH` empties all characters from the serial driver input buffer.

Syntax `FLUSH`

HSRESET

`HSRESET` sets the serial port's flow control options. If you are using a standard modem cable, you will turn off flow control and leave it off. If you are using a CTS/RTS handshaking cable such as described in Appendix B, "Cable Requirements," you need only the `outputCTS` parameter. Turn all options off at hangup.

Syntax

`HSRESET` *`outputXON/XOFF outputCTS XON XOFF inputXON/XOFF inputDTR`*

Parameters

`outputXON/XOFF`	XON/XOFF handshaking for output. For Apple Remote Access, it must be off.
`outputCTS`	CTS hardware handshaking for output. For Apple Remote Access, if you are using a CTS/RTS handshaking cable, it should be on for originate and answer modes and off for hangup mode. For more information, see *Inside Macintosh*, volume 2 (no longer in print) or *Inside Macintosh: Devices*, available through the Apple Developer Catalog.
`XON`	specifies the XON character (DO NOT USE for Apple Remote Access)
`XOFF`	specifies the XOFF character (DO NOT USE for Apple Remote Access)
`inputXON/XOFF`	XON/XOFF handshaking for input. For Apple Remote Access, it must be off.
`inputDTR`	DTR hardware handshaking for input. For Apple Remote Access, it should be off. For more information, see *Inside Macintosh*, volume 4 (no longer in print) or *Inside Macintosh: Devices*, available through the Apple Developer Catalog.

Example

```
HSRESET 0 1 0 0 0 0
```

IFANSWER

If the script is playing in answer mode, `IFANSWER` causes execution to continue at the specified label.

Syntax `IFANSWER` *`jumpLabel`*

Parameter *`jumpLabel`* the label to jump to, where execution continues

Example `IFANSWER 30`

IFORIGINATE

If the script is playing in originate mode, `IFORIGINATE` causes execution to continue at the specified label.

Syntax `IFORIGINATE` *`jumpLabel`*

Parameter *`jumpLabel`* the label to jump to, where execution continues

Example `IFORIGINATE 7`

IFSTR

`IFSTR` compares two strings: one of the variable strings (varStrings, described in “Variable Strings (varStrings)” in Chapter 1) and one that you specify in the script. If the strings are equal, the script continues execution at the specified label.

Syntax `IFSTR` *`varStringIndex jumpLabel`* "*`compareString`*"

Parameters

`varStringIndex` the number of the variable string to compare

`jumpLabel` the label to jump to, where execution continues

`compareString` the string to which the variable string is compared

In the following example, if the modem speaker flag (varString 2) is on (1), execution jumps to label 55. Otherwise, the next command is executed.

Example `IFSTR 2 55 "1"`

IFTRIES

`IFTRIES` compares a parameter with the variable tryCounter. If the value of tryCounter is greater than or equal to the parameter, the script continues execution at the specified label. See also the commands `DECTRIES`, `INCTRIES`, and `SETTRIES`.

Syntax `IFTRIES` *`numTries jumpLabel`*

Parameters

`numTries`	the parameter to compare with the variable tryCounter
`jumpLabel`	the label to jump to, where execution continues

The following example checks to see if the value of tryCounter is greater than or equal to 3. If so, execution jumps to label 62 and continues.

Example

```
IFTRIES 3 62
```

INCTRIES

`INCTRIES` increases the variable tryCounter by one. See also the commands `DECTRIES`, `IFTRIES`, and `SETTRIES`.

Syntax `INCTRIES`

JSR

`JSR` causes script execution to jump to the subroutine located at the specific label, saving the address of the line following the `JSR` command. When a `RETURN` command is encountered, execution resumes at the line following the `JSR` command. `JSR` commands can be nested up to 16 deep.

Syntax `JSR` *`jumpLabel`*

Parameter

`jumpLabel`	the label to jump to, where execution continues

Example

```
JSR 50
```

JUMP

`JUMP` causes script execution to continue at the specified label.

Syntax	`JUMP` *`jumpLabel`*	
Parameter	*`jumpLabel`*	the label to jump to, where execution continues
Example	*`JUMP 59`*	

LBREAK

`LBREAK` generates a long break (3.5 seconds) on the transmission line.

Syntax `LBREAK`

MATCHCLR

The CCL interpreter has a buffer that holds up to 32 strings loaded by the `MATCHSTR` command. The `MATCHCLR` command erases all strings in the buffer. Use `MATCHCLR` before loading each new group of strings. See also the commands `MATCHREAD` and `MATCHSTR`.

Syntax `MATCHCLR`

MATCHREAD

The CCL interpreter has a buffer that holds up to 32 strings loaded by the `MATCHSTR` command. `MATCHREAD` reads input from the serial driver and compares the input to the strings currently in the buffer. If a match is found within the specified `MATCHREAD` time, execution continues at the label associated with that match string (as defined by the `MATCHSTR` command that loaded the string). See also the commands `MATCHCLR` and `MATCHSTR`.

Syntax	`MATCHREAD` *`time`*	
Parameter	*`time`*	the time allowed for a match, in tenths of a second

The following example searches for a match within 3 seconds.

Example `MATCHREAD 30`

MATCHSTR

The CCL interpreter has a buffer that holds up to 32 strings. `MATCHSTR` loads a string to the buffer, so that incoming strings can be matched against it by a `MATCHREAD` command. If an incoming string matches the stored string, script execution continues at the label specified in the `MATCHSTR` command. See also the commands `MATCHCLR` and `MATCHREAD`.

Syntax `MATCHSTR matchNum matchLabel "matchStr"`

Parameters

matchNum	a value from 1 through 32 specifying which string in the buffer to replace
matchLabel	the label to jump to, where execution continues when a `MATCHREAD` command detects a matching string
matchStr	a string (1–255 characters) to compare against

The following example loads the string `"OK\13\10"` into the buffer as string 1. If a subsequent MATCHREAD reads a string that matches this one, then execution jumps to label 8.

Example

```
MATCHSTR 1 8 "OK\13\10"
```

NOTE

The `NOTE` command displays status and log information, passing the message string as a parameter. Optionally, you can set the message level to specify where the message should appear.

Note: While in answer mode, `NOTE` does not write to the Activity Log or to the status window.

Syntax `NOTE msgStr [msgLevel]`

Parameters `msgStr` the message to display

`msgLevel` the message level 1, 2, or 3 (default is 3)

1 send the message to the Activity Log only

2 send the message to the Apple Remote Access Status window only

3 send the message to both the Activity Log and the Apple Remote Access Status window

The following examples show important places in which you should use the `NOTE` command. In the first example, the script logs outgoing calls by displaying the dialed phone number in the Apple Remote Access Status window and the Activity Log. In the second example, the script displays the speed of the connection in the Apple Remote Access Status window.

Examples

```
NOTE "DIALING ^1" 3
NOTE "Communicating at 9600 bps." 2
```

PAUSE

`PAUSE` causes script execution to halt for a specified period of time.

Syntax `PAUSE time`

Parameter `time` the time to pause, in tenths of a second

The following example causes script execution to pause for 2 seconds.

Example

```
PAUSE 20
```

RETURN

`RETURN` terminates a subroutine. Script execution continues with the line following the `JSR` command.

Syntax `RETURN`

SBREAK

`SBREAK` generates a short break (.5 seconds) on the transmission line.

Syntax `SBREAK`

SERRESET

`SERRESET` configures the serial port by passing values for baud rate, parity, data bits, and stop bits to the serial driver. Specifying a value other than the values listed below causes the default value to be used. The defaults for each parameter are listed below.

Syntax `SERRESET` *`baudRate, parity, dataBits, stopBits`*

Parameters

`baudRate`	300, 1200, 2400 (the default), 4800, 9600, 14,400, 19,200, 28,800, 38,400, or 57,600
`parity`	1 for odd parity 2 for even parity 0 or 3 for no parity (the default)
`dataBits`	5, 6, 7, or 8 (the default)
`stopBits`	1 for 1 stop bit (the default) 2 for 2 stop bits 3 for 1.5 stop bits

Example `SERRESET 9600, 0, 8, 1`

Note: Apple Remote Access requires no parity, 8 data bits, and 1 stop bit.

SETSPEED

`SETSPEED` sets the asynchronous serial interface speed to the specified speed. Use `SETSPEED` to set speeds other than those allowed in `SERRESET`.

Syntax `SETSPEED` *`interfacespeed`*

Parameter `interfacespeed` the serial interface speed

Example `SETSPEED 24000`

SETTRIES

`SETTRIES` initializes the tryCounter variable to the specified value. See also the commands `DECTRIES`, `IFTRIES`, and `INCTRIES`.

Syntax `SETTRIES` *`tries`*

Parameter *`tries`* the value to assign to the tryCounter variable

Example `SETTRIES 0`

USERHOOK

`USERHOOK` conveys information about the state of the modem to Apple Remote Access.

Syntax `USERHOOK` *`opcode`*

Parameter *`opcode`*

1 to indicate that the script is answering a call and that a ring is indicated by the modem. This prevents other applications from using the serial port until after the call has terminated.

2 to report that the modem is doing error correction (other than MNP10, which is indicated by opcode 4).

3 to request that Apple Remote Access turn off its built-in data compression.

4 to report that the modem is doing MNP10 error correction.

Example `USERHOOK 1`

WRITE

`WRITE` writes the specified string to the serial driver.

Syntax	`WRITE` *`message`*	
Parameter	*`message`*	the message written to the serial driver

The following example sends to the serial driver the modem command `ATDT` followed by variable string 1 and a carriage return. (For more information, see “Variable Strings (varStrings)” in Chapter 1.)

Example

```
WRITE "ATDT^1\13"
```

Appendix A Result Codes

This appendix lists the result codes returned by the Communication Command Language (CCL). Each result code is shown with a description of the error and the message, if any, that is displayed to the user.

If the script executes successfully, have it exit with result code 0.

If the script is unsuccessful for any reason, have it exit with one of the error result codes listed in this appendix. Note that result code -6002 allows you to pass a custom message to the user.

Result code	Description	Message displayed
-6002	Generic CCL error.	[Supplied by the string parameter in the `EXIT` command.]
-6003	Subroutine overflow.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.
-6004	The target label is undefined.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.
-6005	Bad parameter error.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.
-6006	Duplicate label error.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.
-6007	Close error.	[No message is displayed.]
-6008	The script was canceled.	[No message is displayed.]
-6009	The script contains too many lines.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.
-6010	The script contains too many characters.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.
-6011	The CCL has not been initialized.	[No message is displayed.]
-6012	Cancel in progress.	[No message is displayed.]
-6013	Another script is in progress.	[No message is displayed.]
-6014	Exit with no error.	[No message is displayed.]
-6015	A label is out of range.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.

Continued ▶

Result code	Description	Message displayed *(continued)*
-6016	Bad command.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.
-6017	End of script reached; expected `Exit`.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.
-6018	The match string index is out of bounds.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.
-6019	Modem error; the modem is not responding.	Modem not responding. Reset modem, check connections, or check to see that the proper port and modem type were specified in the Remote Access Setup control panel.
-6020	No dial tone.	The modem cannot acquire a dial tone.
-6021	No carrier.	The modems could not connect. Try again.
-6022	The line is busy.	The phone number you are calling is busy.
-6023	No answer.	The phone number you are calling does not answer.
-6024	No `@ORIGINATE` command in the modem script.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.
-6025	No `@ANSWER` command in the modem script.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.
-6026	No `@HANGUP` command in the modem script.	The file for the modem you selected does not work properly. It may be damaged; try replacing the file in the Extensions folder.

Appendix B Cable Requirements

This appendix describes the CTS/RTS handshaking cable that is recommended when using Apple Remote Access with a V.32bis or faster modem and discusses implications of this wiring scheme for other communications applications.

Cable specifications

To make the most efficient use of Apple Remote Access with a V.32bis or faster modem, use a cable with the following specifications:

Computer DIN-8	Modem DB-25	
1 (DTR)	4,20 (RTS, DTR)	
2 (CTS)	5 (CTS)	Normally pin 2 (CTS) is connected to pin 6 (DSR) on other cables.
3 (TxD-)	2 (TD)	
4 (SG)	7 (SG)	
5 (RxD-)	3 (RD)	
6 (TxD+)	Not connected	
7 (GPi)	8 (DCD)	
8 (RxD+)	7 (SG)	

Some manufacturers ship their V.32bis and faster modems with a cable that meets these specifications.

Modem control issues

A cable constructed as specified in the previous section provides the hardware handshaking that high-speed modems require. If your cable does not meet these specifications, the modem may not operate or may not be able to sustain a connection. The cable supports the following handshaking features:

- CTS handshaking allows the modem to signal the computer to stop sending data to the modem.
- RTS handshaking allows the computer to signal the modem to stop sending data to the computer.
- DTR handshaking allows the computer to signal the modem to reset, hangup the call, or go into command mode.

RTS and DTR cannot be used concurrently. If you want to use RTS, you need to force disconnects by other means than DTR, such as +++, SBREAK, or LBREAK.

If you want to use DTR, the computer must be able to accept data at all times. The computer's serial port must be set to a speed equal to or greater than the modem's highest connect speed. The actual connect speed is the modem to modem data rate, rather than the modem's serial port speed.

DSR and DCD handshaking are not available with this cable. Therefore other types of communications software, such as terminal emulation software, cannot use DSR and DCD signals to detect modem readiness or carrier presence with this cable.

Recommended modem control

The following guidelines provide for optimum performance in most instances:

- Set the computer's serial port speed equal to or greater than the modem's highest connect speed.
- Use CTS handshaking to control data flow to the modem.
- Do not use RTS handshaking.
- If possible with your modem type, use DTR control for hanging up and resetting.

Apple Remote Access

Modem Scripting Guide: Version 2.1

Apple Computer, Inc.

© 1996 Apple Computer, Inc. All rights reserved.

Under the copyright laws, this manual may not be copied, in whole or in part, without the written consent of Apple. Your rights to the software are governed by the accompanying software license agreement.

The Apple logo is a trademark of Apple Computer, Inc., registered in the U.S. and other countries. Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

Every effort has been made to ensure that the information in this manual is accurate. Apple is not responsible for printing or clerical errors.

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014-2084
(408) 996-1010

Apple, the Apple logo, HyperCard, LocalTalk, and Mac are trademarks of Apple Computer, Inc., registered in the U.S. and other countries.

Adobe, Acrobat, and PostScript are trademarks of Adobe Systems Incorporated or its subsidiaries and may be registered in certain jurisdictions.

Helvetica and Times are registered trademarks of Linotype-Hell AG and/or its subsidiaries.

Simultaneously published in the United States and Canada.

Mention of third-party products is for informational purposes only and constitutes neither an endorsement nor a recommendation. Apple assumes no responsibility with regard to the performance or use of these products.

Contents

Preface / v

1 Script Basics / 1

What is a modem script? / 2

Script conventions / 2
- Instructions / 3
- Comments / 3
- Capitalization / 3
- Labels / 4
- Script resources / 4
- String formats / 5
- Variable strings (varStrings) / 6
- Match strings / 8
- Script size / 8

Tasks your script must perform / 9
- Initiating a call / 10
- Answering a call / 13
- Terminating a call / 15

2 CCL Commands / 17

`! Comment` / 18
`@ANSWER` / 18
`@HANGUP` / 18
`@LABEL` / 18
`@ORIGINATE` / 19
`ASK` / 19
`CHRDELAY` / 20
`COMMUNICATINGAT` / 20

DECTRIES / 20
DTRCLEAR / 21
DTRSET / 21
EXIT / 21
FLUSH / 21
HSRESET / 22
IFANSWER / 23
IFORIGINATE / 23
IFSTR / 23
IFTRIES / 24
INCTRIES / 24
JSR / 24
JUMP / 25
LBREAK / 25
MATCHCLR / 25
MATCHREAD / 25
MATCHSTR / 26
NOTE / 26
PAUSE / 27
RETURN / 27
SBREAK / 28
SERRESET / 28
SETSPEED / 28
SETTRIES / 29
USERHOOK / 29
WRITE / 30

Appendix A Result Codes / 31

Appendix B Cable Requirements / 34

Cable specifications / 35

Modem control issues / 35

Recommended modem control / 36

Preface

Apple Remote Access is an application program that lets a computer running the Macintosh operating system (Mac OS) communicate over standard telephone lines with another computer running the Mac OS or with an AppleTalk network. Apple Remote Access is supplied with scripts for a variety of Hayes-compatible modems. To use Apple Remote Access with a modem for which a script is not supplied, the user or the modem vendor must write a script to control the modem.

Modem scripts are written using Communication Command Language (CCL). The Apple Remote Access Modem Toolkit, of which this guide is a part, includes a HyperCard stack, the Modem WorkShop, that is used to write and test scripts. Complete instructions for loading and using the WorkShop are provided in a second guide, the *Apple Remote Access Modem WorkShop User's Guide*, also included in the Toolkit.

About this guide

This guide includes instructions for writing scripts and descriptions of all the CCL commands. It is intended for experienced programmers with a good understanding of telecommunications and modem operation.

This guide assumes that you are familiar with the Mac OS desktop as well as with basic Mac OS skills, such as using the mouse and using the Chooser.

The guide is divided into two chapters and two appendixes:

- Chapter 1, “Script Basics,” describes the basic elements and structure of a CCL file and the basic tasks a script must perform.
- Chapter 2, “CCL Commands,” lists the CCL commands, providing for each a definition, syntax, and an example, if appropriate.
- Appendix A, “Result Codes,” lists result codes returned by the CCL, with a description of the error and the accompanying message, if any.
- Appendix B, “Cable Requirements,” discusses requirements for a CTS/RTS handshaking cable, desirable when using Apple Remote Access with a 9600 bps or faster modem.

What you need to get started

To write a script, your computer must be running Macintosh system software version 7.0 or later. You must have Apple Remote Access software installed on your computer. You also need the Apple Remote Access Modem WorkShop, included in the Modem Toolkit with this manual. Modem WorkShop requires HyperCard version 2.0 or later.

IMPORTANT You will also need the documentation that came with your modem; many commands vary from one modem to another.

For more information

You may find it useful to consult a reference manual on telecommunications and modems, such as *The Complete Modem Reference* by Gilbert Held, published by John Wiley & Sons, Inc.

Consult the *Apple Remote Access Personal Server User’s Guide*, the *Apple Remote Access Client User’s Guide*, or the *Apple Remote Access MultiPort Server Administrator’s Guide* for instructions on using Apple Remote Access with scripts you write using Modem WorkShop.

Available modem scripts

A number of modem scripts have already been written for use with Apple Remote Access. They are placed in the Modem Scripts folder, inside the Extensions folder, which is itself inside your System Folder when you install the Apple Remote Access software. If you have one of the modems for which a script has been provided, you don't need to write a script. You can display a list of the provided scripts from within Modem WorkShop, as described in the *Apple Remote Access Modem WorkShop User's Guide.*

If you need to write a script, you may be able to use an existing script as a template. To do so, see "Modifying a Script" in the *Apple Remote Access Modem WorkShop User's Guide.* Be sure to use the Save As command to make a copy of the script you're modifying, so that you don't overwrite the original script.