

Appendix B • CCL Scripts

Contents of this Appendix

- B.1 Modifying CCL Scripts
- B.2 Printing CCL Scripts
- B.3 Writing CCL Scripts
- B.4 Using Strings
- B.5 Using CCL Statements
- B.6 Using Labels
- B.7 Using Comments
- B.8 Using Serial Port Control Commands
- B.9 Using Scripts Flow Control Commands
- B.10 Using Iteration Commands
- B.11 Using User Notification Commands
- B.12 Using Pattern Matching Commands

This section provides a quick overview of CCL. For more detailed information, consult the [AppleTalk Remote Access Modem Developer's Guide](#). This can be purchased from the Apple Programmers and Developers Association by calling 1.800.282.2732 or sending AppleLink email to [APDA](#)

CCL, or Connection Control Language, is the language used to write modem scripts. Each different type of modem requires a modem script to send commands from the Macintosh to the modem. The commands sent to the modem are used to configure and connect the modem to the remote network.

The CCL scripts used with InterPPP are simple text files designed to dial and answer particular modems. The documents can be copied from machine to machine and can be printed.

The CCL scripts execute in one of three possible modes: originate, answer, or hangup. Each mode has a separate entry point. When a call is initiated, the script is run in originate mode. When call answering is enabled, the script is run in answer mode. When terminating a connection, the script is run in hangup mode.

Because InterPPP does not answer calls, the answer mode does not apply to the InterPPP application. Information regarding the answer mode, such as @ANSWER, is provided for technical accuracy and compatibility with AppleTalk Remote Access CCL scripts.

B.1 Modifying CCL Scripts

InterPPP and your modem manufacturer provide default scripts for generic modems. Usually, there should be no need to edit these scripts. If it is necessary to edit a script, use the following procedures as well as the information in [Section B.3, Writing CCL Scripts](#), and in the [AppleTalk Remote Access Modem Developer's Guide](#).

To edit an existing CCL script:

1. Select [Open CCL Script...](#) from the [File](#) menu.
- ✓ Apple's standard [Open File](#) dialog ([Figure B.1](#)) is displayed.

figure B.1 Apple's Standard Open File Dialog

2. Select a CCL script and click **Open**. (Or click **Cancel** to close the dialog without opening a CCL script.)

✓ The selected CCL script opens (Figure B.2).

figure B.2 An Example of an Open CCL Script

3. Edit the text as necessary. Use the **Cut**, **Copy**, and **Paste** commands of the **Edit** menu if needed.

B.2 Printing CCL Scripts

To print a CCL script:

1. Open the CCL script (see [Section B.1, Modifying CCL Scripts](#) , for information).
2. Select [Print...](#) from the [File](#) menu.
- ✓ Apple's standard [Print](#) dialog is displayed.
3. Select your printing options and click [Print](#). (Or click [Cancel](#) to close the dialog without printing.)
- ✓ The CCL script is printed.

B.3 Writing CCL Scripts

If InterPPP or your modem manufacturer does not provide a CCL script for the modem you are using, you can create a script using the information in the following sections, as well as the information in the [AppleTalk Remote Access Modem Developer's Guide](#) .

To create a CCL script:

1. Select [New CCL...](#) from the [File](#) menu.
- ✓ A standard text editing window labeled [Untitled](#) is displayed.
2. Use the information in [Section B .4- B.12](#) to enter the text. Use the [Cut](#), [Copy](#), and [Paste](#) commands of the [Edit](#) menu as necessary.
3. Select [Save](#) from the [File](#) menu to save the CCL script.
- ✓ Apple's standard [Save](#) dialog is displayed.
4. Name the CCL script, select the location where it is to be saved, and

click [Save](#). (Or click [Cancel](#) to close the dialog without saving the script.)

✓ The name of the CCL script is displayed at the top of the window.

To use the script you have created in a connection document, select it from the pop-up menu in the [Modem Configuration](#) dialog. See [Section 5.1, Modem Configuration](#), for more information.

B.4 Using Strings

The strings used in CCL scripts are enclosed in quotes (" "). Defined varStrings are:

[varString 1](#)

the dial string (the phone number)

[varString 2](#)

the modem “speaker on” flag

[varString](#) is a string that contains parameters for modem setup, such as the phone number the modem is calling and whether the modem’s speaker is on or off during dialing.

The following special characters are allowed within strings:

[\13](#)

substitutes a carriage return (hex 0D) into the string.

[\00](#)

substitutes a null character (hex 00) into the string.

[\\](#)

substitutes the \ character (hex 5C) into the string.

[\^](#)

substitutes the ^ character into the string.

`\"` substitutes the " character into the string.

`^1` substitutes the `varString` into the string.

`^*` substitutes the `ASK` string into the string.

The `^` character is used to reference a `varString` to substitute into the string.

B.5 Using CCL Statements

Modem scripts are stored as text files, with one CCL statement per line, either a label, comment, or command.

Labels

Labels begin with an at sign (`@`) and continue until the end of the line. They denote a point in the script that is the target of a jump from another part of the script. There are two kinds of labels, symbolic and numeric. Symbolic labels contain the at sign (`@`) followed by a word. Numeric labels contain the at sign (`@`) followed by a number.

Comments

Comments begin with an exclamation mark (`!`) and continue until the end of the line.

Commands

Commands start with a word (the command name) and are followed by arguments on the rest of the line. Blank lines are ignored.

B.6 Labels

@ANSWER

marks the point in the script where execution begins in answer mode.

Syntax

@ANSWER

@HANGUP

marks the point in the script where execution begins in hangup mode.

Syntax

@HANGUP

@LABEL

sets a numeric label in the script. You can then reference this label from other script commands like JUMP, JSR, and IFTRIES. A script may have up to 128 labels, numbered 1 through 128.

Syntax

@LABEL labelnum

Parameter

labelnum

a value from 1-128 that specifies the label number

Example

@LABEL 30

@ORIGINATE

marks the point in the script where execution begins in originate mode.

Syntax

@ORIGINATE

B.7 Comments

!Comment

designates a comment line in the script. Comments can begin anywhere in a command line and extend to the end of the line.

Syntax

! comment

Example

! This is a comment line.

B.8 Serial Port Control Commands

DTRCLEAR

clears the DTR (Data Terminal Ready) signal on the RS-232 interface.

Syntax

DTRCLEAR

DTRSET

sets the DTR signal on the RS-232 interface.

Syntax

DTRSET

FLUSH

empties all characters from the serial input buffer.

Syntax

FLUSH

HSRESET

sets the serial flow control options.

Syntax

HSRESET outputXON/XOFF outputCTS XON XOFF
inputXON/XOFF inputDTR

The PPP protocol ignores XOn/XOff because it is unable to process these characters. They are provided to ensure backward compatibility with AppleTalk Remote Access CCL scripts.

Parameters

outputXON/XOFF	
XOn/XOff	handshaking for output
outputCTS	
CTS	hardware handshaking for output
XON	
	specifies the XOn
character	
XOFF	
	specifies the XOff
character	
inputXON/XOFF	
XOn/XOff	handshaking for input
inputDTR	
DTR	hardware handshaking for input

Example

HSRESET 0 1 0 0 0 0

LBREAK

generates a long break (3.5 seconds) on the serial output.

Syntax

LBREAK

SBREAK

generates a short break (233 milliseconds) on the serial output.

Syntax

SBREAK

SERRESET

configures the serial driver, providing values for baud rate, parity, databits, and stop bits. Giving a zero value for any of the parameters causes the default value to be used.

Syntax

SERRESET baudRate parity dataBits stopBits

Parameters

baudRate

300, 1200, 2400, 4800, 9600 (default), 19200, 38400, or 57600

parity

1 for odd parity, 2 for even parity, 3 for no parity (default)

dataBits

5, 6, 7, or 8 (default)

stopBits

1 for 1 stop bit (default), 2 for 2 stop bits, 3 for 1.5 stop bits

Example

SERRESET 9600, 3, 8, 1

proper operation of PPP requires 8 databits, no parity, and 1 stop bit.

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

Parameters

interfacespeed
the serial interface speed

Example

```
SETSPEED 14400
```

WRITE

writes the specified message to the serial driver.

Syntax

WRITE message

Parameters

message
the message written to the serial driver

The following example sends the message variable string 1 and carriage return to the serial driver.

Example

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

B.9 Script Flow Control Commands

EXIT

terminates execution of the script and returns a result. If the script completes successfully, the result returned is zero. If the script fails for any reason, the result returned is a CCL error code.

Syntax

EXIT result

Parameters

result
a CCL error code

The example below shows the script exiting and returning the error result code for a busy signal (-6022).

Example

EXIT -6022

JSR

causes script execution to jump to the subroutine specified by the label, saving the address of the line following the JSR command. Upon encountering a RETURN command, execution resumes at the line following the JSR command.

Syntax

JSR jumpLabel

Parameters

jumpLabel
the label to jump to, where execution continues

Example

JSR 50

JUMP

causes script execution to continue at the specified label.

Syntax

JUMP jumpLabel

Parameters

jumpLabel

the label to jump to, where execution continues

Example

JUMP 56

IFANSWER

causes execution to continue at the specified label, if the script is running in answer mode.

Syntax

IFANSWER jumpLabel

Parameters

jumpLabel

the label to jump to, where execution continues

Example

IFANSWER 30

IFORIGINATE

causes execution to continue at the specified label, if the script is running in originate mode.

Syntax

IFORIGINATE jumpLabel

Parameters

jumpLabel

the label to jump to, where execution continues

Example

IFORIGINATE 30

IFSTR

compares two strings. If the strings are equal, the script continues execution at the specified label.

Syntax

```
IFSTR varStringIndex jumpLabel compareString
```

Parameters

varStringIndex

the index of the varString to compare with

jumpLabel

the label to jump to, where execution continues

compareString

the string that the varStringIndex string is compared
against

In the following example, if the “modem speaker on” flag equals 1, execution jumps to label X.

Example

```
IFSTR 2 X "1"
```

PAUSE

causes script execution to stop for a specified period of time.

Syntax

```
PAUSE time
```

Parameters

time

the time to pause, in tenths of a second

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

Example

```
PAUSE 20
```

RETURN

terminates a subroutine. Script execution continues with the line following the last JSR command executed.

Syntax

```
RETURN
```

B.10 Iteration Commands

DECTRIES

decrements the internal try counter by one.

Syntax

```
DECTRIES
```

IFTRIES

compares a parameter with the internal try counter. If they are equal, the script continues execution at the specified label.

Syntax

```
IFTRIES numTries jumpLabel
```

Parameters

numTries

the parameter to compare against the internal try counter

jumpLabel
the label to jump to, where execution continues

The following example checks to see if the internal try counter is equal to 3. If it is, the execution jumps to label 62 and continues.

Example

```
IFTRIES 3 62
```

INCTRIES

increments the internal try counter by one.

Syntax

```
INCTRIES
```

SETTRIES

initializes the internal try counter to the specified value.

Syntax

```
SETTRIES tries
```

Parameters

tries
the value to assign to the internal try counter

Example

```
SETTRIES 0
```

B.11 User Notification Commands

ASK

displays a dialog to obtain information from the user when the script is initialized. You may need the ASK command if your telephone system uses special telecommunications equipment.

Syntax

```
ASK maskflag "message"
```

Parameters

maskflag

1 or true indicates that the user's input is masked with dots
(....)

0 or false indicates that the user's input is displayed as
normal text

message

the prompt displayed in the dialog for the user

Example

```
ASK true "Enter your password"
```

NOTE

sends messages to the internal Activity Log. Optionally, you can set the message level to specify where the message should appear.

Syntax

```
NOTE msgStr msgLevel
```

Parameters

msgStr

the message parameter passed

msgLevel

1 sends the message to the Activity Log only

2 sends the message to the Status window only

(default)

3 sends the message to the Activity Log and Status window

Example

NOTE "Dialing ^1"

USERHOOK

causes the script to perform a special action. USERHOOK passes a parameter to the internal hook, so this hook can perform more than one function.

Syntax

USERHOOK opcode

Parameters

opcode

1 marks a connection as active when a call is answered and a ring is indicated by the modem

2 causes a terminal window to be displayed

Example

USERHOOK 2

B.12 Pattern Matching Commands

MATCHCLR

clears all match strings. The CCL interpreter has an internal buffer that holds up to 16 strings identified with the MATCHSTR command. The buffer holds this information until it is erased with a MATCHCLR command. You should use MATCHCLR at the beginning of every modem script to clear the buffer.

Syntax

MATCHCLR

MATCHREAD

reads input from the serial driver and compares the input to the current match strings. If a match is found in the specified time, execution continues at the label of the matching match string.

Syntax

MATCHREAD time

Parameters

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

specifies a string to match incoming characters against. If a stream of consecutive incoming characters matches the string, script execution continues at the specified label. Sixteen possible match strings exist.

Syntax

MATCHSTR matchNum matchLabel matchStr

Parameters

matchNum

a value from 1-16

matchLabel

the label to jump to, where execution continues

matchStr

a string (1-255 characters) to compare against

The following example matches the string "OK\13\10" with match string one. If the strings match, then execution jumps to label 8.

Example

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