d dataton

TRAX

TRUE MULTIMEDIA

3.6

ADDENDUM

Table of Contents

1 Overview 5
SMARTPAX QC 5
Built-in Devices 5
Revised Cues 5
Scripting 5
Named Expressions 6
Tasks 7
Timeline 8
Expression Menu 10
Miscellaneous 11

2 Devices 13
Device Window 13
Port Assignment 14
Device Standby 17
Internal CD-ROM Audio Playback 18
Hard Disk Audio Playback 21

3 Panel Design 23
Using the Monitor as a Panel Device 23
Button Item News 26
Displaying Live Video 27
Picture Item News 28
Display Item News 28

4 Cues 29
Locate 29
Set/Fade 30
Trigger 32

5 Scripting 33
Scripting Options 35
AppleScript 37
Serial Port Scripting 47
TCP/IP Network Port Scripting 54
Network Configurations 64
Scripting Language Overview 65
Object Reference 73
Commands 78
Replies 86
Scripting Errors 87
Scripting Examples 89
Scripting with Macromedia Director 103
Scripting with Microsoft Windows 110

6 Named Expressions 114
Using Expressions 116
Examples 117

7 SMARTPAX QC 124
Front Panel 125
Rear Panel 129

8 TIMECODE SMARTLINK 132
Reading Timecode 132
Control Signal Recording 133
Generating Timecode 135
Technical Specifications 136

Index 137

Document number: 3947

1 OVERVIEW

This chapter gives an overview of new features that have been added to TRAX since the TRAX 3 handbook was printed.

SMARTPAX QC

This version of TRAX fully supports the new SMARTPAX QC control unit, and its simplified user interface. Please refer to Chapter 7 beginning on page 124 for full details.

Built-in Devices

TRAX takes full advantage of the media capabilities built into your MacOS computer including:

- Sound playback from the built-in CD-ROM drive (page 18).
- Sound playback from the hard disk (page 21).
- Use of the computer monitor as a control panel, with capabilities similar to the TOUCHLINK touch panel (page 23).

Revised Cues

The Locate, Set/Fade and Trigger cues have been enhanced. Besides new features, they also support the use of named expressions. Please see the description of these cues beginning on page 29.

Scripting

The new scripting capabilities provide control of TRAX itself, as well as the presentation devices it commands, from other programs and computers. Through scripting, you can control many functions in more or less the same way as using the mouse and keyboard or by executing cues or timelines.

Scripting commands consist of short sentences sent to TRAX from another program, called the client. The client can run on the same computer as TRAX or on another computer. It communicates with TRAX either using messages sent directly between the programs, over a computer network, or via a serial port.

TRAX has a small but powerful vocabulary, proving commands to manipulate devices, cues, timelines and even TRAX itself (eg, to load a new show file). Commands are also provided to query TRAX for information. This allows you to determine, for example, if a timeline is running, or to query the current level of a slider or other device.

Please refer to Chapter 5 beginning on page 33 in this addendum for a complete description of the scripting features.

Named Expressions

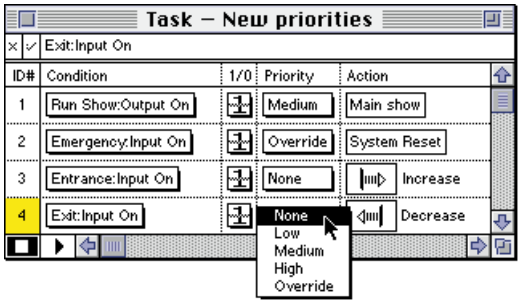
A named expression is a power-user feature that provides increased flexibility and modularity. It can be used to

- Create a numeric keypad for user entry of values, for example to locate chapters on a laserdisc or to enter the starting time of a presentation.
- Link a single slider or analog input to multiple outputs, for example to fade multiple lighting channels from a single input.
- Add the equivalent of program variables to cues and tasks.
- Provide named constants and sub-expressions, making the program easier to maintain.

Please refer to Chapter 6 beginning on page 114 in this addendum for a complete description of named expressions.

Tasks

Two new priority levels and a new starting condition variant have been added to the tasks in the Task window.



None

The "None" priority level is lower than "Low". Its primary advantage is that it doesn't claim ownership of devices, thereby avoiding device ownership conflicts. This is particularly useful for handling events that must access the same device. The example above shows two tasks increasing and decreasing the same visitor counter when triggered by entrance and exit door sensors.

The "None" priority level can only be used for single-cue tasks, not for timelines. This priority level is equivalent to the priority of panel buttons.

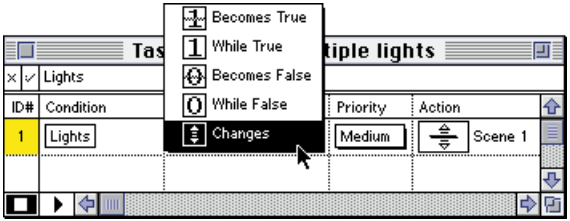
Override

The "Override" priority level is higher than "High". Furthermore, a task set to "Override" will *always* be granted access to all devices it needs, regardless of the priority of their current owner. As this, in effect, bypasses the priority mechanism, use it only when absolutely necessary. In most cases, conflicts should be avoided or handled in other, more intelligent, ways.

The "Override" priority level can be used both with single-cue tasks and timelines.

New Condition Variant: Changes

A new starting condition variant has been added, named "Changes". If this is selected, the task is started when the value of the expression changes.



This variant is particularly useful in conjunction with the new named expressions, and the way these can be used in lieu of values in cues. See the example under "Fading Multiple Devices" on page 120.

◆ **NOTE:** When the task list is started, the initial value of all expressions is considered to be zero. Thus, any initial value that's different from zero will be considered as "changed", and will cause the task to be started.

Timeline

Some new features have been added to timeline windows:

- Pressing the Shift key while selecting "Select to End" on the Edit menu selects all cues ahead on the current track only.

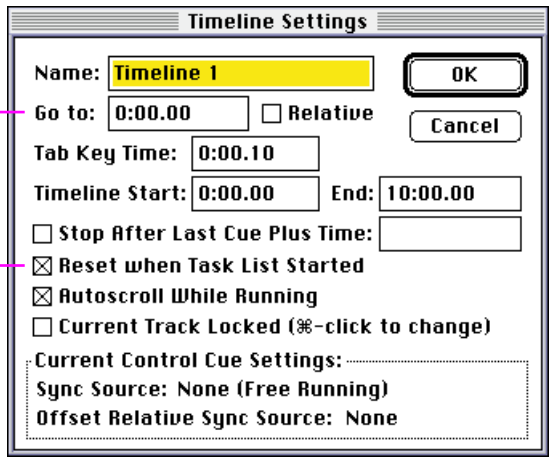
✍ **SHORTCUT:** Press Shift-Command-E.

- The spacebar and arrow keys can be used in a timeline window without causing the task list to stop (ie, TRAX remains in run mode). This is useful in mixed canned/speaker support presentations, where you may want to control the pace of one timeline manually while other timelines continue to run in the background.

- A checkbox specifies if the timeline should be stopped and reset automatically when the Task list is started. This checkbox is selected by default. Unchecking this checkbox also allows you to restart a stopped timeline from its current position using a Control cue.

Press the equal sign key while in a timeline window to open the Timeline Settings dialog box and jump straight to this field.

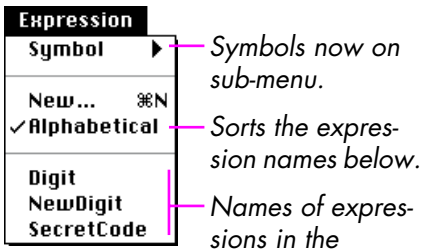
Uncheck this checkbox to make the timeline stay put when starting the task list (eg, by pressing Command-Spacebar).



- Pressing the equal sign "=" brings up the Timeline Settings dialog box with the "Go To" field selected. (This is similar to pressing the "+" or "–" keys, except that it allows you to jump to a time position on the timeline in an absolute rather than relative way.)
- Pressing Option-Tab jumps to the point ahead on a timeline where currently selected devices will become ready. This can be used with devices that can play to frame (ie, laserdiscs), as well as slide projectors (ie, to find the calculated end of the tray cycle). If certain devices are selected then the com-

mand will apply to those. If no particular devices are selected then all devices owned by the timeline will be considered. You can also press Command-Option-Tab to move selected cues to the ready-point.

Expression Menu



A new Expression menu replaces the old Symbol menu. The Symbol menu now appears as a sub-menu on the Expression menu.

The bottom of the Expression menu contains the names of all expressions defined in the Expression window. This allows you to enter these names into cues or other expressions.

- **SHORTCUT:** As an alternative to choosing an expression name from the menu, you can type it using the keyboard.

The Expression menu will become active while inside a field in a cue that supports named expressions, such as the "Scale Factor" field of the Set/Fade cue. See "Named Expressions" on page 114 for more details on how to use expressions.

The list of expressions at the bottom of the menu can be sorted as they appear in the Expression window or alphabetically. To sort them alphabetically, select "Alphabetical" on the Expression menu. This changes the order on the menu only, and doesn't affect the Expression window.

- **SHORTCUT:** The "New…" command on the Expression menu provides a keyboard shortcut for creating new expressions in the Expression window.

Miscellaneous

- Numeric values in dialog boxes can be nudged by holding down the Command key while pressing the arrow keys. Holding down the Shift key as well nudges the value in larger steps.
- Pressing the Command-Left and Command-Right arrow keys in fields that provide frame-step buttons (such as in the status window for a video disc player, or the time fields in a Locate cue) is equivalent to clicking those buttons.
- The arrow keys can be used to select objects in Device and Panel windows.
- A system run-mode indicator has been added, providing the same information as the play symbol in the lower left corner of the Task window. This indicator appears in the menu bar while running the Task window. This is particularly helpful when the Task window is hidden.

☞ **SHORTCUT:** Press Command-Spacebar to start running the Task window. This works even if the Task window is hidden.

- The demonstration and full versions of TRAX have been integrated. The program can now be used in demonstration mode until a valid serial number is entered, which then transforms it into the full version. The separate demonstration and runtime versions of TRAX have been discontinued.

▼ **IMPORTANT:** TRAX is not public domain, freeware or shareware. You must obtain a personal serial number in order to use TRAX, except in its demo mode. You may not distribute or give away a copy of TRAX with a serial number. The serial number can be obtained free of charge by faxing or mailing the registration form (displayed when you start the demo version of TRAX). Alternatively, you can register online at www.dataton.com under the "Free Software" heading.

PowerPC Native

TRAX is now PowerPC native, meaning that it takes full advantage of the PowerPC processor used in newer MacOS computers. This results in a six-fold speed increase of processor-bound functions, such as screen redraws, devices status calculations and scanning of the Task list.

The TRAX application file contains both PowerPC and 68xxx code, allowing it to run on any MacOS computer regardless of processor type.

System Requirements

68xxx computers. System 7.1 and 4 MB RAM for basic functionality. 7.6.1 or later and 8 MB RAM recommended (required for some functions).

PowerPC computers. System 7.5.5 and 8 MB RAM for basic functionality. 7.6.1 or later and 16 MB RAM recommended (required for some functions).

MacOS 8 Compatibility

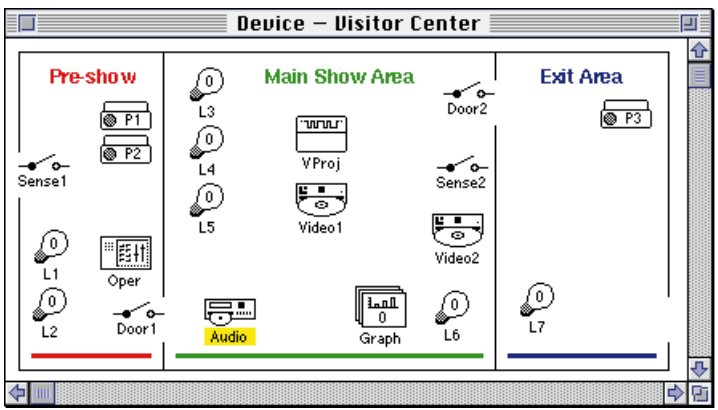
To maintain the consistency of the user interface in TRAX, you should turn off the "System-wide platinum appearance" checkbox in the Appearance control panel (found under "Control Panels" on the Apple menu).

2 DEVICES

This chapter describes new features added to devices, such as enhanced port assignment and support for audio playback from the internal CD-ROM or hard disk.

Device Window

A background picture can be pasted into the Device window. Create the picture using a drawing or painting program (eg, ClarisWorks). Copy it, and paste it into the Device window. To cut or copy the picture from the Device window, first make sure that no devices are selected.



A picture, created in Claris Works, has been pasted into the Device window as a background.

Arrow Key Selection

You can use the arrow keys to select devices in the device window, in a way similar to the MacOS Finder. The same applies to items in panel windows.

Port Assignment

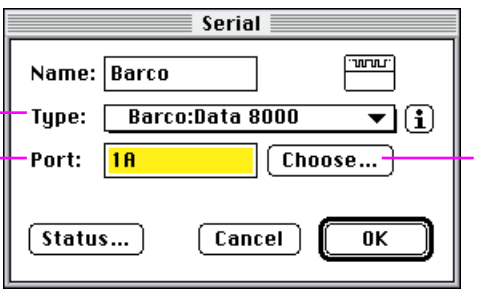
A new method of assigning ports to devices has been added to support the SMARTPAX QC control unit model. SMARTPAX QC uses a simplified method where a single, rotary selector on the front panel selects a unit ID in the range 1 through 15 (see the illustration on page 125). In addition to this unit number, each of the four control ports is designated by a letter A through D (see page 129).

SMARTPAX and PAX continue to use numeric addresses (10 through 77) to designate each output port.

SMARTPAX QC

When using SMARTPAX QC, state the unit ID followed by the port letter to specify the output port in the device configuration dialog box.

Device configuration dialog box.



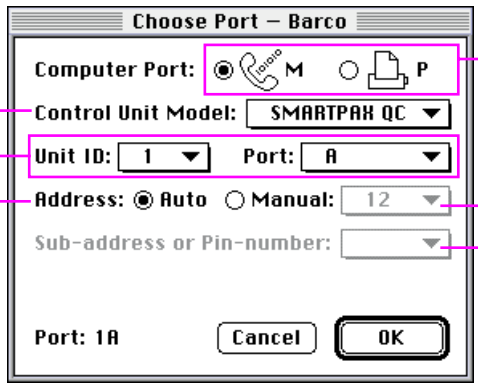
First specify the type of device…

…then enter the port assignment in the appropriate format…

…or click this button to set the port assignment interactively.

As an alternative to typing the port assignment, you can click the "Choose" button to bring up a secondary dialog box, allowing you to specify the same information interactively.

Choose Port dialog box.



Select type of control unit.

Specify SMARTPAX QC ID and Port.

Choose "Auto" and TRAX will assign an address to the SMARTPAX QC. See "Specifying a SMARTPAX QC Address Manually" on page 16 regarding "Manual".

Applies only if you've selected "Both" serial ports in the Preferences dialog box.

Specify address (applies only if "Manual" is selected).

Sub-addresses are used for some types of devices (see page 118 in the TRAX 3 handbook).

First choose the kind of control unit being used from the "Control Unit Model" pop-up menu. The choices available depend on the type of device being controlled, as specified on the Type pop-up menu in the device configuration dialog box. For example, the PAX control unit model is only available for Projector and Switch devices.

If you choose PAX or SMARTPAX you must choose an address on the "Manual" pop-up menu. Addresses already in use are indicated by a dash on the pop-up menu. If you have activated "Gangs" in the Preferences dialog box, the maximum address available will be lower than 77 depending on the number of gangs activated.

If you choose SMARTPAX QC on the Control Unit Model pop-up menu, the Unit ID and Port pop-up menus become available. Choose the unit ID corresponding to the setting of the ID selector on the front panel of the SMARTPAX QC. Choose the port, A through D, to which the device is connected (see picture on page 129). Unit IDs and ports already in use are indicated by dashes on the pop-up menus.

In most cases you can leave "Address" set to "Auto", allowing TRAX to allocate an address to the port automatically.

When you have finished configuring all devices, you must update the system using the "Download" button in the "Device Support" window.

▼ **IMPORTANT:** You must update the system even if you only make a small change to the configuration, such as altering a unit ID number or address.

Specifying a SMARTPAX QC Address Manually

In some cases you may need to specify the address manually even when using SMARTPAX QC. This applies, for example, if you want to run a show from a control track on tape. In this case, it is important that the system is configured with the same physical addresses that were in use when the tape was recorded. These addresses are typically indicated on the slide projector trays or other documentation accompanying the show.

To run such a presentation using SMARTPAX QC, choose "Manual" and specify both the ID/Port and the 10 through 77 address number for each device used in the show. You must then update the system configuration using the "Download" button in the "Device Support" window.

Device Standby

The Disable Device command on the Object menu also allows you to put selected devices on standby. By holding down the Shift key when selecting this command (or clicking the Disable checkbox in the device's status window), the device will be set to standby as part of being disabled.

Standby behavior varies according to the device:

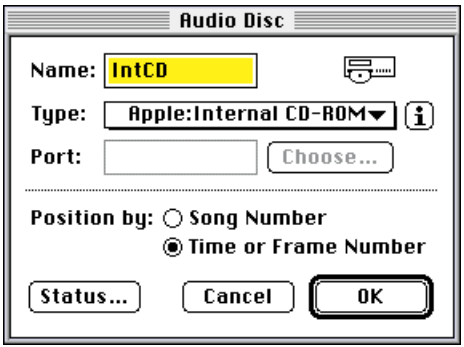
Device	*Standby behavior*
Slide Projector	Turn off the projector lamp
Tape	Stop playback
Audio Disc	Stop playback
Video Visc	Stop playback
Lamp	Turn off the lamp
Level	Set the level to zero
Others	No effect

The Disable and Enable Device commands have keyboard shortcuts ("/" and "*" respectively, available in the numeric area of most keyboards). Thus, you can set a device to standby and disable it by selecting it in the Device window and pressing Shift-Command-/.

- **NOTE:** Do not confuse this standby feature with any standby capabilities built into particular device models. Such standby capabilities are not activated by this command. In most cases, device-specific standby features can be controlled using a Trigger cue.

Internal CD-ROM Audio Playback

You can play audio CDs using the CD-ROM player built into most desktop MacOS computers. To use this feature, add an Audio Disc device to the Device window, and choose "Apple:Internal CD-ROM" on the Type pop-up menu.



You program the internal CD in the same way as an external device connected through a SMARTPAX; using a Locate cue to position the CD, a Trigger cue to Play it, etc.

Positioning Options

Choose "Position by: Time" for best positioning accuracy. In this mode, the disk will be positioned automatically during editing when controlled from a timeline. It is also possible to synchronize a timeline to the CD using a Control cue (see "Synchronizing to an Intelligent Device", page 222, TRAX 3 handbook).

Choose "Position by: Song Number" if you want to be able to access songs numerically, for example from a TOUCHLINK panel. This, however, does not provide accurate tracking within songs when controlled from a timeline.

Volume Control

You can control the volume of the CD using a fader on a panel or a Level cue on a timeline. By specifying a rate in the Level cue, you can fade the volume gradually, or make cross-fades between audio from the CD and the hard disk.

System Requirements

You can only control a single CD-ROM device in this way. The computer must have the following capabilities and features:

- A built-in Apple CD-ROM drive and the following support files in your Extensions folder (located in the System folder): Apple CD-ROM version 5.1.7 or later, Foreign File Access 5.1 or later, Audio CD Access 5.1 or later.
- System 7.6.1 or later.
- The computer's hardware must support "Asynchronous SCSI". This is supported by all 68040 and PowerPC based desktop models.
- You must have updated the drivers on your computer's hard disk(s), as prompted during the installation/updating procedure.

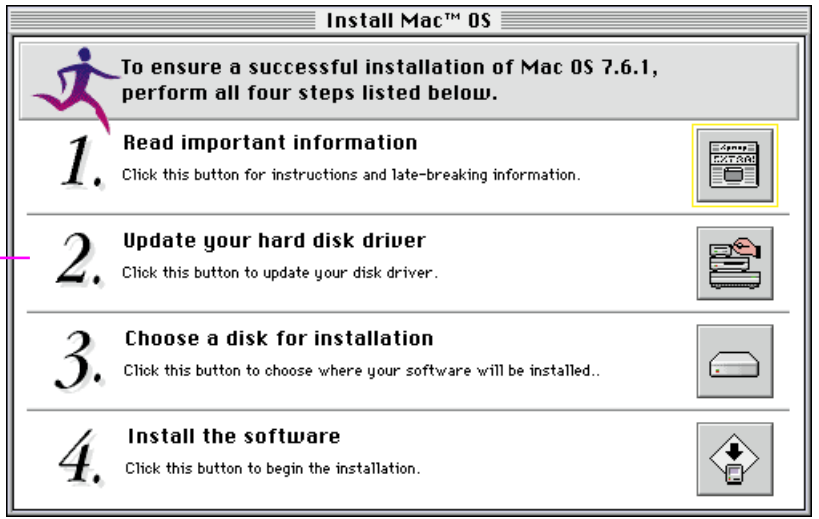
Upgrading the Driver on the Hard Disk

All hard disks connected to your computer have driver software embedded. This driver software is automatically loaded by the operating system when you boot your computer or mount the disk. When using the built-in CD or hard disk audio playback features of TRAX, it is important that all hard disks connected to your computer are updated with drivers compatible with "Asynchronous SCSI" (also known as "SCSI Manager 4.3").

▼ **IMPORTANT**: It is imperative that you follow these steps to update the drivers for *all* hard disk and CD-ROM drives connected to your computer in order to be compatible with "Asynchronous SCSI"/"SCSI Manager 4.3". Not doing so will result in system crashes when playing audio from the hard disk or the CD-ROM.

If you use original Apple hard disk(s) only, then the drivers on those disks are updated automatically as part of the system installation/upgrading procedure. This is step two of the installation procedure for System 7.6.1, as shown below.

Click this button to update your hard disk driver.



If you use external hard disks (including Syquests, ZIP Drives, etc), those must be updated as well. For all non-Apple disk drives, you must contact the driver manufacturer for details on how to update the driver. This applies even if they are not actively used to play back audio.

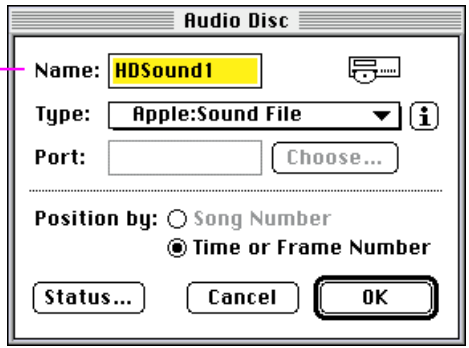
External CD-ROM drives usually don't store the driver on the medium. Instead, the driver is installed by a system extension. If you have an external, non Apple CD-ROM drive connected to your computer, this may need to be updated too.

Hard Disk Audio Playback

You can play back audio from the computer's hard disk. All modern desktop models have very good audio playback capabilities, usually providing full CD-quality audio.

To use this feature, add an Audio Disc device to the Device window, and choose "Apple:Sound File" on the Type pop-up menu.

Type the name of the sound file.



The sound file must be of type AIFF (a standard audio file format supported by all major sound editing applications). Store the sound file (or an alias) in the same folder as the show file. Type the name of the sound file, or its alias, into the "Name" field in the device's configuration dialog box.

Although you can not programmatically change the name of the sound file associated with an audio disc device, you can have multiple audio disc devices, each associated with its own file. By starting and stopping these devices, you can access their individual sound files. This even allows you to

play several sound files simultaneously (depending on computer hardware capabilities).

You program the hard disk audio in the same way as an external device connected through a SMARTPAX, using a Locate cue to position the sound, a Trigger cue to Play it, etc.

Volume Control

You can control the volume of the sound file using a fader on a panel or a Level cue on a timeline. By specifying a rate in the Level cue, you can fade the volume gradually, or make cross-fades between audio from the hard disk and the built-in CD-ROM player.

System Requirements

In order to play audio files from the hard disk, the computer must have the following capabilities and features:

- System 7.6.1 or later.
- The computer hardware must support "Asynchronous SCSI". This is supported by all 68040 and PowerPC desktop models.
- You must have updated the drivers on your computer's hard disk(s), as prompted during the installation procedure (see page 19).

▼ **IMPORTANT:** You *must* update the drivers of all hard disks and CD-ROM drives attached to your computer, as described under "Upgrading the Driver on the Hard Disk" on page 19. If you don't, you will suffer system crashes when playing audio from the hard disk or the CD-ROM.

3 PANEL DESIGN

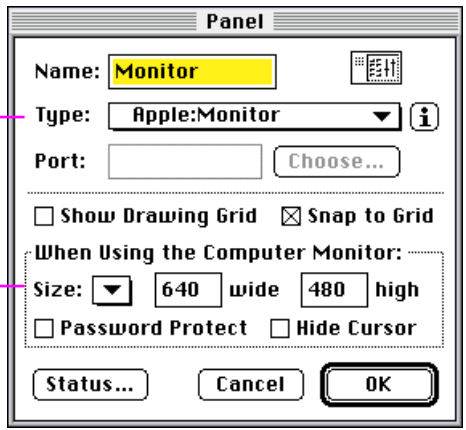
The Panel device has been enhanced to allow the computer's monitor to be used as a panel. The Button and Display items have also been enhanced.

Using the Monitor as a Panel Device

To use your computer's monitor as a panel device, choose "Apple:Monitor" on the Type pop-up menu in the panel's configuration dialog box.

Choose "Apple:Monitor".

Specify the target monitor's size.



Monitor Size

Specify the size of the target monitor using the "Size" pop-up menu, or by typing the dimensions into the fields. This allows you to design the panel for another monitor size than the current one, if desired. Scrollbars will appear in the panel's status window in order to manage larger panel sizes.

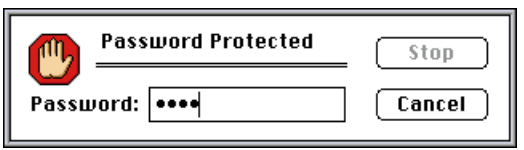
You can change the size of a panel afterwards, if desired. However, you must manually re-arrange the items on the panel's pages to fit the new size.

- **NOTE:** When running the panel on the monitor by pressing Command-Spacebar, the panel will always zoom out to cover the entire monitor. The size set in the configuration dialog box is only a design aid, causing the outline for the specified size to be indicated while editing the panel.

Password Protection

When using the panel in its full screen mode, you can choose to protect the system using a password. This uses the password specified in the "Security Options" dialog box (see page 113 in the TRAX 3 handbook).

When activated, a dialog box will appear when you attempt to use the keyboard while the panel is in its full screen mode. This dialog box allows you to stop the system by entering the correct password. The password dialog box disappears automatically after a few seconds.



Using a Touch Overlay

By attaching a touch overlay to the monitor, you can use it as a touch panel. Such touch overlays typically emulate the computer's mouse, and connect to the computer through its mouse (ADB) port or through a serial port.

Hand Cursor: 

When using a touch overlay, you may prefer to hide the hand cursor normally shown when operating a panel. This is accomplished by selecting the "Hide Cursor" checkbox.

Adding Pictures

When you paste a picture onto a monitor panel, it will be rendered using 256 colors instead of the 8 colors available to TOUCHLINK panels. This results in better color fidelity.

Using the Monitor Panel

To use the monitor panel in its full screen mode, first make sure that the panel's status window is currently selected, then press Command-Spacebar to start the system. This causes the panel to zoom up and cover the entire monitor.

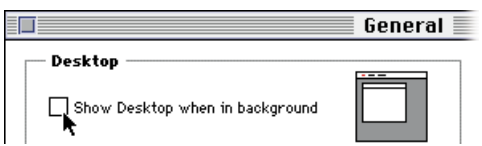
- **HINT:** If Command-Spacebar doesn't work on your computer, check if you have the "SCSI Probe" control panel installed. If you do, open it, click the Options button, and change the "Mount key" from Command-Spacebar to some other key combination.

Press any key to exit the system run mode, causing the panel to zoom back to its editing position. If you've activated the password feature of the panel, you must enter the password in order to exit the system run mode.

Using Multiple Monitors

If you have more than one monitor connected to your computer, the panel will zoom to the main monitor (the one with the menu bar). You can use the other monitors for other purposes (eg, to display device status windows).

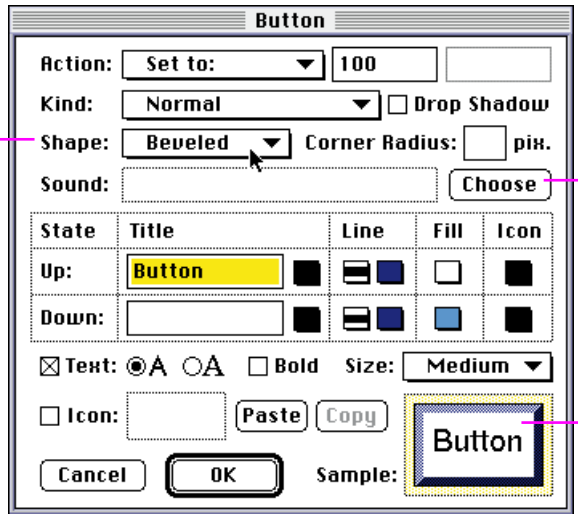
Clicking the mouse in a window that belongs to another application, the Finder, or the desktop may cause TRAX to exit its run mode even if the panel is password protected. This can be prevented by deselecting "Show Desktop when in background" in the General control panel.



Button Item News

Beveled "3D" shaped buttons are available in addition to the rectangle and oval shapes described in the TRAX 3 handbook. A button can also have a sound, which is played when the button is pressed.

Button shape can be Rectangle, Oval or Beveled.



Click here to add a sound to the button (see "Sound" on page 169 in the TRAX 3 handbook for details).

Example of the beveled button shape (also available for sliders).

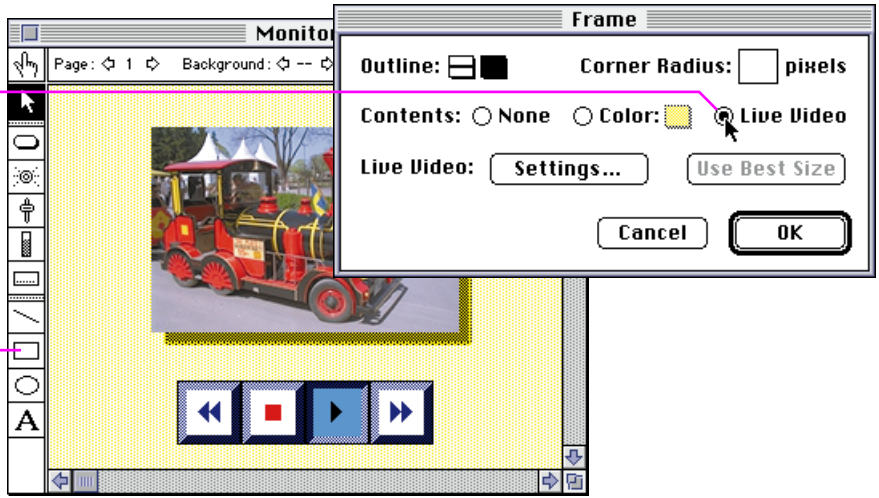
When a button is assigned to a named state of a device (such as the "Play" or "Rewind" states of the transport mode of a tape device), you can choose "Momentary" on the "Action" pop-up menu in the button's configuration dialog box. This allows you to make a button that springs back to its default state when released. This is particularly useful in conjunction with IR drivers, where you often need such momentary functions to, for example, increase or decrease the volume for as long as a button is held down.

Displaying Live Video

The Frame item allows display of live video when using the "Apple Monitor" type of panel on a computer with a suitable video input.

The "Live Video" checkbox is only available if your computer and system software supports a video input.

Draw a Frame, double-click it and choose "Live Video".



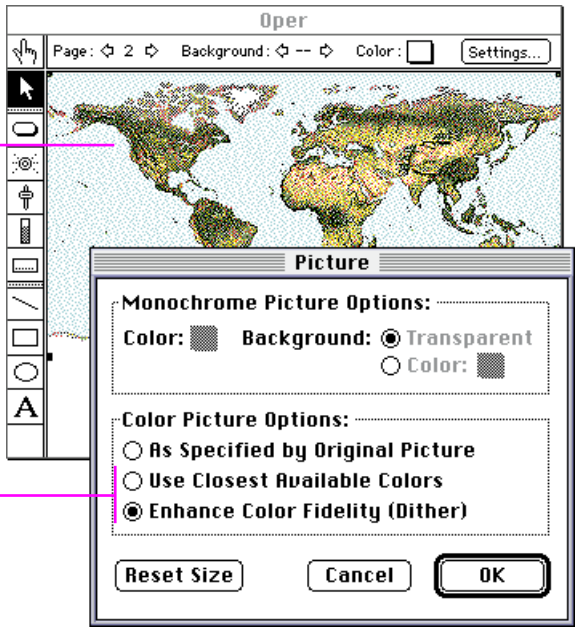
After choosing "Live Video" in the dialog box, click "Settings…" to adjust the video image. In particular, make sure you select the correct video standard (PAL/SECAM/NTSC). Finally, click the "Use Best Size" button if you want to make sure that the aspect ratio of the enclosing frame matches the video image.

- **NOTE:** This feature is only supported on Apple Macintosh computers with built-in video input. It may not work with third party video input cards. Live video can not be used on TOUCHLINK panels.

Picture Item News

Using the "Color Picture Options", you can choose between color fidelity or smoothness. In particular, when using pictures on TOUCHLINK panels, you may need to adjust this option to obtain the best possible picture display.

When choosing "Enhance Color Fidelity", colors may appear to be closer to the original. However, the image may also look grainier.



Use these options to improve the looks of color pictures.

Display Item News

Support for the AM/PM time format has been added to the display item. This is specified inside the configuration dialog box for the display item.

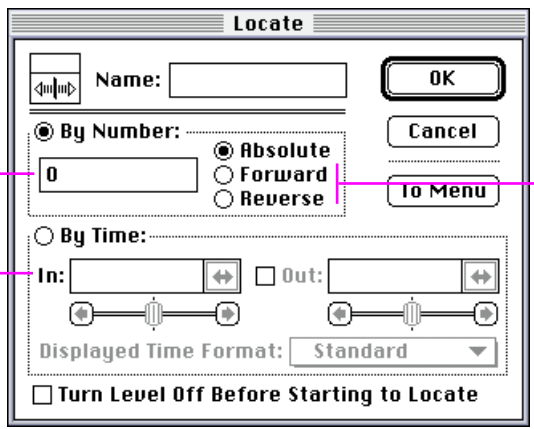
4 CUES

The Set/Fade, Locate and Trigger cues have been enhanced. These cues also support the use of named expressions.

Locate

The Locate cue is enhanced with the ability to move forward or backward by a specified number of steps. You can use a named expression instead of a numeric constant in the "By Number" and "By Time" fields (see the example under "Creating a Numeric Keypad" on page 122).

You can use the name of an expression instead of a numeric constant in these fields.



Select "Forward" or "Reverse" to move in that direction by the specified number of steps.

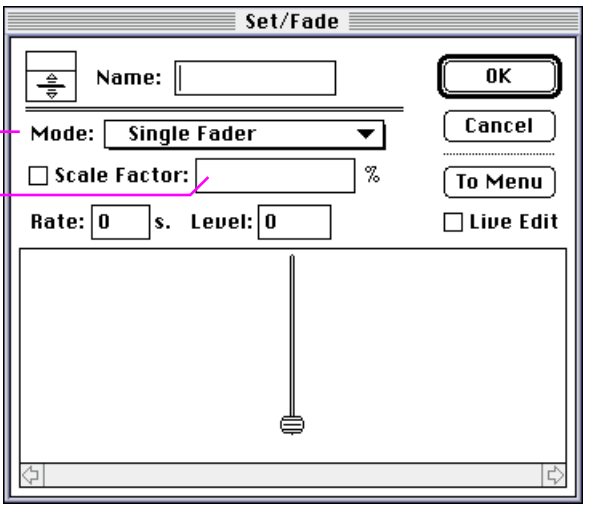
- **NOTE:** In order to maintain compatibility with the PAX slide projector control unit, Forward and Reverse by a value greater than one can not be used with slide projectors. If you do, they will still advance only a single position.

Set/Fade

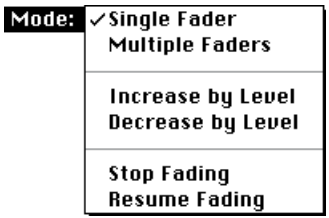
The Set/Fade cue is enhanced with the ability to increase or decrease the level of a device by a specified amount. The various modes of the Set/Fade cue have been consolidated into the Mode pop-up menu. The new Scale Factor field allows you to adjust the overall level by a specified percentage. This is particularly useful when using the cue's Multiple Faders mode. The scaling can also be performed dynamically by using a named expression.

Specifies the mode of the cue.

You can use the name of an expression instead of a numeric constant in this field.



Mode



Single Fader. Sets or fades the level of all assigned devices to the specified level. All devices will be set to the same level.

Multiple Faders. Provides a separate fader for each device assigned to the cue (up to 32). You can use the Scale Factor to adjust the overall level, if desired.

Increase by Level. Increases the levels of all devices assigned to the cue by the amount specified in the "Level" field. For example, assume you have three Lamp devices assigned to the cue, currently at 15, 25 and 35 percent brightness. If you the choose "Increase by Level", and set Level to 10, the levels will be increased to 25, 35 and 45 percent. Levels will be clipped when they reach 100 percent.

Decrease by level. Same as "Increase by Level", but decreases the levels of the assigned devices instead.

Stop Fading. Stops any fading or dissolve in progress on the assigned devices. This can be used as an alternative to fading to a specific percentage, for example to keep fading until a button is pressed.

Resume Fading. Resumes a fading interrupted by a previous "Stop Fading" cue.

Scale Factor

When selected, the device levels set by the cue will be scaled by the percentage specified in the "Scale Factor" field. A value of 100 percent is equivalent to no scaling and a value of 0 percent is equivalent to setting the levels to zero.

This field is particularly useful when using the "Multiple Faders" mode of the cue, as it allows you to adjust the overall look of, for example, a lighting scene, without adjusting each individual fader.

Using the Scale Factor with a Named Expression

By specifying a name of an expression in the Scale Factor field instead of a numeric constant, you can make the behavior of the cue more dynamic since the value can then be derived from a device. This can be used, for example, to fade multiple lighting channels using a single fader on a panel or a single Level input device (eg, a MIDI continuous controller input).

See "Fading Multiple Devices" on page 120 for an example.

Rate

Specifies the fade rate, in seconds. You can specify tenths of seconds for increased accuracy. Set this to 0 to set the level instantly.

Level

The level of the currently selected fader, as a percentage between 0 and 100. When using the "Single Fader" mode of the cue, this is the level to which all devices will be set/faded. Select "Live Edit" to see the levels of assigned devices change as you edit the cue.

- **NOTE:** Some devices support fractional percentages for extra precision.

Trigger

The trigger cue has no new visible features. However, you can use the name of an expression in the "Device Specific Mode" value field, if desired. See the example under "Named Constants" on page 118.

5 SCRIPTING

Scripting adds an alternative way to control TRAX, in addition to the program's graphical user interface and external control from devices such as AIRLINK and TOUCHLINK.

Through scripting, you can control most aspects of TRAX, such as:

- TRAX itself: Opening and saving show files, starting and stopping the task list, etc.
- Timelines: Creating, deleting, positioning, starting, stopping, querying the configuration, status and contents.
- Cues: Creating, performing, deleting, querying the settings.
- Devices: Controlling their status either directly or by performing cues, querying device configuration and status.

Client/Server

The scripting commands always originate from another program, called the client. The client sends a scripting command to TRAX, which performs it and returns any information requested to the client. Thus, TRAX acts as a server for the scripting client.

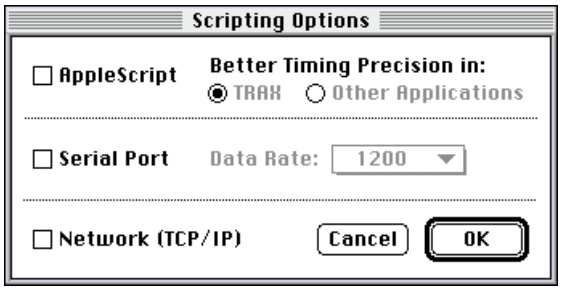
- **NOTE:** TRAX can only act as the server. It can not send scripting commands to itself, nor to other programs.

The client program can run on the same computer as TRAX; on another computer connected via the network; or on an external device connected through a serial port. The scripting language remains the same in all these cases, allowing you to develop your script on one platform and deploy it on another.

You can have multiple clients talking to one TRAX server at the same time. It is also possible to have multiple TRAX servers on the same network, each talking to its own set of devices.

Scripting Options

To activate the scripting capabilities, open the Scripting Options dialog box by clicking the Scripting Options button in the Preferences dialog box.



The three checkboxes to the left allow you to activate the various scripting ports. Each scripting port represents a particular way by which scripting commands can be sent to TRAX.

AppleScript

This is the standard way for MacOS applications to talk to each other. The client application can run either on the same computer or on another computer across a network.

Better timing precision in: TRAX. When selected, TRAX will attempt to maintain the best possible timing precision, while still giving other applications on the same computer the opportunity to perform their functions whenever possible. However, due to the cooperative nature of the MacOS, TRAX has no control over how much time the processor spends attending to other applications. Consequently, the timing precision in TRAX may suffer.

Better timing precision in: Other Applications. When selected, TRAX will yield extensively to other applications running on the same computer. This may improve the timing precision in those applications at the cost of even further reduced timing precision in TRAX.

Serial Port

This option allows you to send scripting commands to TRAX through the computer's Printer or Modem port. This is particularly useful when interfacing TRAX to other control systems, which may not support networking capabilities.

Data Rate. Specifies the communication bit rate (sometimes referred to as "baudrate") between TRAX and its client, in bits per second. Using higher data rates results in better timing precision and less overhead for the scripting client.

TCP/IP

When selected, clients can connect to TRAX via a network that uses the TCP/IP protocol, supported by virtualy all computers and operating systems.

Most desktop MacOS computers come with built-in Ethernet capabilities both in hardware and software. Ethernet can also be added by means of expansion cards to most desktop models and portables.

AppleScript

The AppleScript scripting port allows you to send commands to TRAX from other MacOS applications that provide this capability. Some applications provide extensive capabilities for creating, executing and debugging scripts, while others merely allow you to run a "canned" script.

These are some examples of applications that can talk to TRAX using AppleScript:

- Script Editor (Apple – included with MacOS).
- HyperCard (Claris/Apple).
- FileMaker Pro (Claris).
- QuicKeys (CE Software).
- Excel (Microsoft).
- Frontier (UserLand) .
- FaceSpan (Digital Technology International).

The major advantages of AppleScript are:

- It's the standard method for MacOS applications to communicate with each other.
- It allows you to run both the client and TRAX on the same computer.

Running TRAX and the client application on the same computer will, in many cases, result in unacceptably slow performance of both TRAX and the client application. This is due to limitations in the multitasking capabilities of MacOS, as well as the slow performance of AppleScript itself. Thus, this is primarily useful for various "off line" purposes, such as:

- Developing and debugging scripts, using Apple's Script Editor or other similar application.
- "Batch processing," such as copying the text from all Note cues along a timeline into a word processor, or building timelines from a database or an EDL (edit decision list).

You can improve the timing somewhat by running the client application on another computer via a network. In this case, most of the AppleScript overhead is handled by the client computer, and the performance of TRAX is greatly improved.

Prerequisites

The basic software you need to use AppleScript is included with MacOS. Depending on which version of MacOS you use, it may be implemented by one or more system extensions, located in the Extensions folder:

- AppleScript™
- AppleScriptLib
- ObjectSupportLib

To write and test scripts you'll also need one of the following applications (or equivalent):

- Script Editor (Apple – included free with MacOS).
- ScriptWizard (Full Moon Software).
- ScriptDebugger (Late Night Software).
- Scripter (Main Event).

While there's some scripting documentation included with MacOS as well as

the script editors mentioned above, you may want to obtain additional documentation, such as:

- AppleScript Language Guide, English Dialect. Apple Computer/Addison-Wesley, ISBN 0-201-40910-0, 1994, 80 pages.
- The Tao of AppleScript, 2nd Edition, Derrick Schneider/Hayden Books, ISBN 1-56830-115-4, 1994, 387 pages.
- AppleScript for Dummies, Tom Trinko/IDG Books, ISBN 1-56884-975-3, 1995, 396 pages.
- Applied Mac Scripting, Tom Trinko/M&T Books, ISBN 1-55828-330-7, 1995, 877 pages.
- Danny Goodman's AppleScript Handbook, Danny Goodman/Random House, ISBN 0-679-75806-2, 1994, 554 pages.

Activating AppleScript

Although AppleScript is available in TRAX by default, TRAX normally disables AppleScript while running timelines or the task list. Thus, you may use many AppleScript functions without activating AppleScript.

By explicitly activating AppleScript in the Scripting Options dialog box (page 35), you tell TRAX to listen for AppleScript commands even while running potentially time critical functions, such as timelines or the task list. The additional AppleScript options in the Scripting Options dialog box give you some degree of control over the trade-off between TRAX, AppleScript and other applications running on the same computer.

▼ **IMPORTANT:** Do not activate AppleScript if your presentation requires the best possible timing precision. The TCP/IP and Serial scripting ports don't noticeably affect the timing precision of TRAX, however.

Creating and Editing Scripts

Apple's Script Editor.

The most straightforward way to create scripts is using Apple's Script Editor. This application comes free with the MacOS system software.

Please refer to the documentation included with the Script Editor, or any of the books mentioned on page 39, for full details on how to use AppleScript.

While Apple's Script Editor provides all the basic functions needed to create, run and debug scripts, you may prefer one of the other commercially available script editors/debuggers (see the list on page 38). These generally offer more features than Apple's standard editor. Contact your Apple dealer to obtain these products.

If you have access to the Internet, another good source for information is

http://www.scriptweb.com

Accessing the TRAX Scripting Dictionary

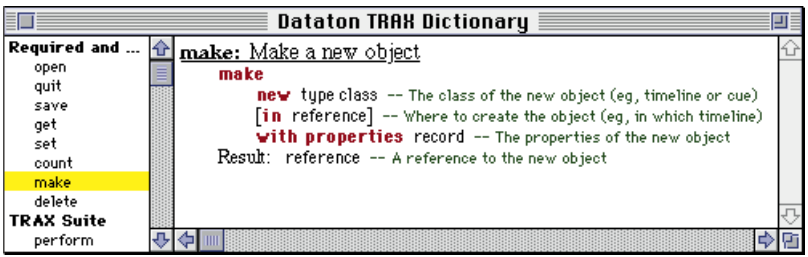
When creating scripts using Apple's Script Editor, you basically deal with two entities:

- System software and AppleScript specific constructs (including variables, control structures, etc).
- Application specific constructs (ie, the various commands that can be sent to an application, the objects managed by that application and their properties).

The documentation included with AppleScript and the books listed on page 39 mainly deal with the first of these points (although they sometimes also cover some popular applications, such as QuarkXpress or Microsoft Excel).

The commands that deal with specific applications have to be more or less tailored to the application at hand. For example, TRAX deals with devices, cues and timelines where a spreadsheet deals with cells, figures and formulas.

In order to help you script a particular application, Apple's Script Editor (as well as the other scripting products mentioned on page 38) provide access to the *scripting dictionary* of that application. This acts as a quick reference to the commands, objects (sometimes called "elements") and properties that can be accessed.

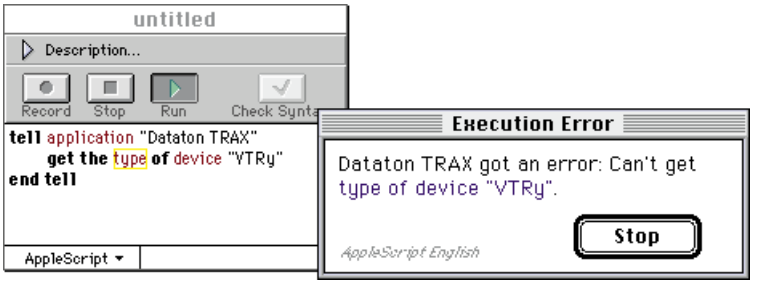


Scripting dictionary in TRAX, as seen from within Apple's Script Editor.

To access the scripting dictionary of TRAX from Apple's Script Editor, either drag the TRAX application icon onto the Script Editor's icon in the finder, or choose "Open Dictionary" on the File menu inside the Script Editor. Other script editors provide similar functionality. Once you've learned AppleScript, you'll find this quick reference handy in case you've forgotten some detail in the application's scripting language.

Testing and Debugging

AppleScript provides excellent tools to test and debug your scripts. As scripts are entered, the basic syntax is checked to conform to the AppleScript language itself as well as the TRAX dictionary. When you run your script, additional checking is performed both by AppleScript and by TRAX. If an error occurs, an error message is displayed and the offending line is highlighted.



While running the script, an Event Log window can be opened to show all commands sent to TRAX, as well as any replies or error messages.

Other commercial alternatives to Apple's Script Editor provide even more sophisticated debugging features, such as the ability to single-step scripts and display variables.

All this makes AppleScript a good starting point for learning how to script TRAX. Remember that the scripting commands sent to TRAX remain the same regardless of whether they're coming from AppleScript or through the TCP/IP or Serial ports. Thus, you can use AppleScript to write and debug your commands, and then simply copy/paste them into the desired context.

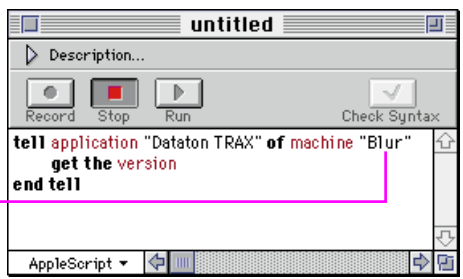
Using AppleScript over a Network

Although TRAX has its own built-in network scripting capabilities (using the TCP/IP scripting port), it is also possible to use AppleScript via a network. In this case, you must activate AppleScript in the Scripting Options dialog box. You don't need to activate the TCP/IP scripting port in order to use AppleScript over a network.

- **NOTE:** Running AppleScript over a network is only possible when the client computer runs MacOS and the client application supports AppleScript. The built-in TCP/IP scripting doesn't have this limitation.

Running the client on a separate computer results in better performance in TRAX. However, the overall speed of AppleScript itself remains more or less the same, so you won't see any improvements in speed in the client software.

When using AppleScript over a network, both computers must be on an AppleTalk network (eg, LocalTalk or Ethernet). AppleTalk must be configured for the network hardware being used. You must also activate Program Linking in the Sharing Setup control panel. Please refer to Apple's documentation for full details on how to run AppleScript via a network.



The name of the computer on the network running TRAX.

To run the script from the client computer over the network, you must also modify the script so it knows how to find TRAX. This is done by adding the name of the computer that runs TRAX as well as the zone of that computer on the network (not required if your network isn't divided into zones). The name of the computer is specified in the Sharing Setup control panel.

Capabilities Specific to AppleScript

As AppleScript is a system-wide scripting language, it can handle many functions not directly related to TRAX. This is different from the TCP/IP and Serial scripting ports, which are specific to TRAX, and only handle TRAX-related commands.

AppleScript can talk to other scriptable MacOS applications, using the commands in their dictionaries. Often, these commands are similar in spirit and syntax to the TRAX commands, with the main difference being the kinds of objects on which they operate. For example, you can give the command "count the paragraphs" to a word processor, which is similar to how you tell TRAX to "count the devices".

In addition to such application-specific commands, AppleScript also has a rich set of built-in functions, operators, control structures and variables. All those are built into AppleScript itself, and are not handled by TRAX or other scriptable applications.

For example, in the script below, only the two bolded commands are directly related to TRAX:

```
tell application "Dataton TRAX"
    get the time of cue -1 of timeline "Main Show" as integer
end tell
copy the result  to cueTime
repeat with paraNum from 1 to 5
    tell application "Scriptable Text Editor"
        get the text of paragraph paraNum of document "Story"
    end tell
    copy the result to cueText
    copy cueTime + 200 to cueTime
    tell application "Dataton TRAX"
        make new cue in timeline "Main Show" with properties ¬
            {time:cueTime, contents:"Note; Text " & cueText}
    end tell
end repeat
```

The second line obtains the time of the last cue of the timeline named "Main Show" (note how you can index a cue from the end of a timeline using a negative number). The line beginning with "make new cue" creates a new cue on the same timeline. The other commands in this script are not related to TRAX, but handled by AppleScript or the "Scriptable Text Editor" (available from Apple as part of their scripting software).

Looking through the example above, you can see how AppleScript is used to:

- Get the time of the last cue on the timeline named "Main Show". This command is directed to TRAX, as specified by the first line in the script.
- Copy the result of the "get" command into an AppleScript variable named cueTime.

- Repeat all commands up until the matching “end repeat” five times.
- Get the text of each paragraph in turn from the "Scriptable Text Editor" application, storing it into the variable named cueText.
- Add 200 to the time in cueTime. Since the time was obtained from TRAX “as integer”, TRAX returned it in centiseconds (hundredths). Thus, by adding 200 to cueTime, you move it forward by 2 seconds.
- Make a new cue at the time specified in the cueTime variable, with the note text specified in the cueText variable. (Although the “make” command is written on two lines, the “¬” character tells AppleScript to logically treat these as a single line.)

This example uses a command to another application (the "Scriptable Text Editor") together with some variables and a repeat loop to add five new cues to a timeline in TRAX.

TRAX only sees the two bolded commands beginning with “get the time…” and “make new cue…”. Furthermore, TRAX never sees the variables named “cueTime” and “cueText”, as these are replaced by their content before being sent to TRAX. The “&” operator following “Note; Text ” causes AppleScript to join that text with the contents of the cueText variable, making it appear as a single piece of text to TRAX .

Scripting Other Applications

Due to the slow performance of AppleScript, it is not suitable for control that requires precision timing. Instead, it excels in moving data from other applications into TRAX, or vice versa. This can be used to automate the creation of cues and timelines, or to extract information from TRAX for use in other applications.

When writing such scripts, you must be familiar not only with the scripting language in TRAX and AppleScript, but also with the scripting language of the other application. The dictionary built into all scriptable applications, as mentioned under "Accessing the TRAX Scripting Dictionary" on page 40, is often a good help here, once you've got a basic understanding of AppleScript. In some cases, additional information is provided in the program's manual or in on-line documentation supplied with the program.

Note that not all MacOS applications are scriptable. Some scriptable applications have only very rudimentary support for scripting, which can make it difficult, or sometimes impossible, to access the data you need.

If you find that an application you need to interface isn't scriptable, you may still be able to access the data by first transferring it into another, similar application with sufficient AppleScript support. For example, while Microsoft Word version 5 isn't scriptable, version 6 is. So by upgrading to a newer version, you may sometimes be able to access your data. Likewise, most applications support export/import functions, which may allow data to be moved between applications even if they're not from the same manufacturer.

Serial Port Scripting

You can use one of the serial ports on the computer running TRAX to send scripting commands to TRAX. These commands are sent as ASCII character strings, using the same language as for the AppleScript and TCP/IP scripting ports. Thus, it is often advantageous to use AppleScript to develop and test the commands, and then transfer them into the desired context. Once you're familiar with the scripting language, you can of course type the commands straight into the client program.

The primary advantages of the Serial scripting port are:

- It can connect to virtually any computer or control system capable of transmitting characters through a serial port.
- It is reasonably fast, particularly at higher data rates.
- It has negligible impact on the timing precision of TRAX.

A disadvantage compared to the AppleScript and TCP/IP scripting ports is that the Serial port is only point-to-point, meaning that there can only be a single client talking to a single TRAX server through its serial scripting port.

Activating the Serial Port

To activate the serial scripting port, select the "Serial Port" checkbox in the Scripting Options dialog box, accessed through the "Scripting Options" button in the Preferences dialog box (see page 35). TRAX will then use the remaining free serial port to receive scripting commands.

As one of the serial ports is typically used by TRAX to talk to SMARTPAX and other control units, you must make sure that the other serial port is available before you can use it for scripting commands. This may include turning off AppleTalk in the Chooser or disabling background communications programs (eg, fax programs, ARA, etc) using the Extension Manager.

◆ **NOTE:** It is not possible to use the Serial scripting port if "Both" is selected in the "Serial Port" section of the Preferences dialog box.

Specifying the Data Rate

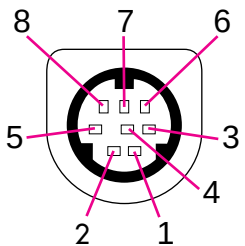
Choose the desired data rate on the pop-up menu next to the "Serial Port" checkbox (see page 35). For best performance, choose the highest rate your client device supports. The scripting port uses 1 stop bit and no parity. All commands and replies are sent as ASCII strings.

Serial Port Wiring

Macintosh computers use serial ports based on the RS-422 standard. This is a balanced signal, which typically makes it more reliable than the more common RS-232C. However, an RS-422 port can usually talk to an RS-232C port using the proper wiring.

To connect a Macintosh serial port to another Macintosh computer, use an "ImageWriter Cable". This has an 8-pin Mini-DIN connector at each end.

To connect a Macintosh serial port to a PC, you can usually use a Macintosh modem cable and a null modem adaptor. Alternatively, make a cable wired as follows:



Macintosh serial port wiring.

Macintosh pin	*Macintosh signal*	*9-pin PC pin*	*25-pin PC pin*
1	Handshake Out >	6, 8	5, 6
2	Handshake In <	7	4
3	Transmit – >	2	3
4	Ground	5	7
5	Receive – <	3	2
6	Transmit + >	N.C.	N.C.
7	N.C./GPi <	N.C.	N.C.
8	Receive + <	5	7
Chassis	Shield	Chassis	Chassis

◆ **NOTE:** N.C. in the table above stands for No Connection.

Handshaking

The serial scripting port supports hardware input and output handshaking. This prevents serial buffer overflow in TRAX or the client device if either end sends data faster than the other end can receive it. This is particularly useful at higher data rates.

- **NOTE:** Some Macintosh models will not transmit unless pin 2 is properly connected to a handshaking signal. If the client device doesn't support hardware handshaking, you can instead loop pin 2 to pin 1 in the Macintosh end. In this case, do not connect anything else to these pins.

Line Endings

Commands sent through the serial scripting port must be terminated by a line feed (hex 0A), carriage return (hex 0D) or null (00) character, or any combination of those. Replies sent back from TRAX are terminated by a null character.

Sending Commands

Send a command to TRAX by transmitting the characters through the serial port at the selected data rate. Terminate the command as specified in the previous paragraph.

Example:

```
set the run mode of timeline 1 to play
```

If no error occurs and no transaction ID was used (see "Queries and Transaction IDs" on page 51), most commands perform silently without returning anything to the client.

Sending Queries

When you send a query to TRAX, such as the "count" or "get" commands, it returns an equal-sign followed by the reply. If an error occurs while processing the command, an error message will be returned instead.

Example (reply shown in *italics*):

```
count devices
=12
```

Queries and Transaction IDs

When sending queries to TRAX through its serial scripting port, it is sometimes advantageous to send multiple queries together, and then wait for all the replies to come back. This can be faster than waiting for the reply to each query before sending the next.

In order to help you associate each answer with its corresponding query, you can tag the query with a transaction ID. This is typically a unique number, but may be any string of printable ASCII characters. This transaction ID is sent immediately before the query, within square brackets:

```
[4135]count devices
```

When TRAX sends the reply to this query, it will be preceded with the transaction ID in the same way:

[4135]=12

When TRAX replies to a successful query, it sends an equal sign followed by the reply. If the query had a transaction ID, this is returned, within square brackets, at the beginning of the reply, as shown above.

Using Transaction IDs as Acknowledge

You can use a transaction ID with any command – not just with queries. When a transaction ID appears at the head of a command, TRAX will always return that transaction ID when the command has been completed, even if it doesn't call for any data to be returned. This can be used as a positive acknowledge that the command has been received and executed without any error.

Example:

```
[4136]set the value of status "Output On" of device "Screen Down" to true
[4136]
```

This is particularly useful if an error occurs, since it allows you to know which command caused the error, even if multiple commands are sent rapidly:

```
[4137]set the value of status "Output On" of device "Screen Ut" to false
[4138]set the value of status "Output On" of device "Screen Down" to true
[4137]!-1728
[4138]
```

In this case, the first command (with transaction ID [4137]) failed because the name of the device wasn't spelled correctly. The second command was OK. Without transaction IDs, you probably wouldn't be able to tell which command that failed if they were sent back to back.

Errors

If an error occurs while processing a script command, TRAX returns an exclamation point immediately followed by a negative error code. If the command that caused the error had a transaction ID, then that ID will appear at the head of the error message.

Examples:

count divorces
!–1723

[4139]get the name of device 9581
[4139]!-1719

An error message is always returned if an error occurs, even if the command would normally execute silently. See page 88 for a list of error codes.

TCP/IP Network Port Scripting

You can use one of the network ports (LocalTalk or Ethernet) on the computer running TRAX to send scripting commands to TRAX. These commands are sent as ASCII character strings, using the same language as for the AppleScript and Serial scripting ports. Thus, it is often advantageous to use AppleScript to develop and test the commands, and then transfer them into the desired context. Once you're familiar with the scripting language, you can of course type the commands straight into the client program.

The primary advantages of the TCP/IP scripting port are:

- It allows multiple servers and clients on the same network wire.
- It uses standard networking protocols, allowing it to run over virtually any kind of network – including most networks already installed in offices and buildings.
- It's usually faster than the serial scripting port (performance may vary with network type and load).
- It has negligible impact on the timing precision of TRAX.

Configuring Open Transport

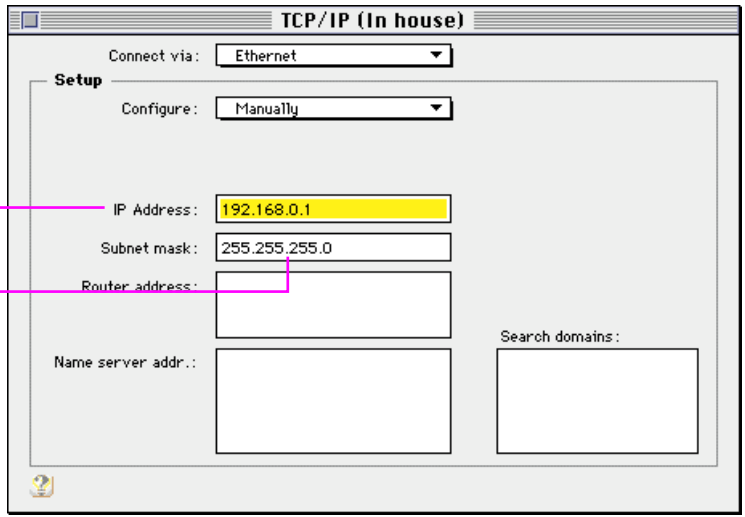
As its name indicates, the TCP/IP scripting port uses the TCP/IP (Transmission Control Protocol / Internet Protocol) network standard to send commands and responses across a network. Under MacOS, the TCP/IP protocol is manager by a system component called Open Transport, which is included with MacOS version 7.5 or later (version 7.6.1 or later recommended).

In order to talk to a computer over a TCP/IP network, it must have a network address, also called an IP number. When using TCP/IP to access the Internet as a client using a modem, this IP number is typically assigned automatically by your Internet Service Provider. However, when using TCP/IP to provide a

server function, such as when accessing TRAX from other clients on the network, the server must have a known address. To accomplish this, type the address into the TCP/IP control panel, as shown below.

IP number to be used as an address for the TRAX server.

The Subnet mask varies with the address range being used.



Choose "Ethernet" or "AppleTalk (MacIP)" on the "Connect via" pop-up menu depending on the physical wiring being used. If you use LocalTalk (ie, connect to the network through the Printer port), choose "AppleTalk (MacIP)". If you're using Ethernet, you should normally choose "Ethernet". Alternatively, choose "AppleTalk (MacIP)" and specify "Ethernet" as the physical wiring in the AppleTalk control panel.

When using a local area network in an office or a building, you can pretty much choose whatever IP numbers you want, as long as each computer has a

unique number. However, if you want to connect your computer or network to the worldwide Internet, you must apply for your globally unique IP number with the proper national authorities.

Network Wiring

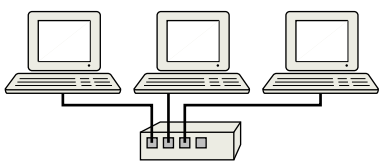
You can run TCP/IP over virtually any kind of network wiring. However, most desktop MacOS computers come with LocalTalk and Ethernet built in, making these natural choices. While it is possible to use LocalTalk, Ethernet is recommended for best performance.

LocalTalk

LocalTalk is wired as a bus, with a "dongle" connected to the Printer port of each computer. The primary advantage of LocalTalk is that it is available in all MacOS compatible computers, and it is very inexpensive.

Coaxial Ethernet

Ethernet can be wired either as a bus or as a star. For bus configurations you typically use a coaxial cable with BNC connectors that runs through all the computers. This means that if the cable is broken at any point along its way, the network will stop working. Furthermore, the coaxial cable must be properly terminated at both ends.



10Base-T Ethernet

If you want Ethernet wired in a star configuration, you need a hub (junction box) to which all the wires are connected. The most common wiring standard for this configuration is called 10Base-T, and uses a small, snap-locking connector (RJ-45). This cabling is sometimes considered easier to manage and more reliable since it doesn't matter if a cable is unplugged, and there's no termination to worry about. However, it may require more wiring, and it adds the cost of a hub.

- **NOTE:** It is possible to use 10Base-T between two computers without the need for a hub. All you need is a crossed 10Base-T cable, available in most computer shops. This is also useful as a troubleshooting aid.

Often a combination of star and bus configurations are used for larger networks. A hub is used for each local cluster of computers. Those computers are wired to the hub using 10Base-T or similar in a star configuration. The hubs are then tied together using a coaxial cable, often referred to as a "network backbone".

The computer or network card may come with any of the following connectors:

Coaxial (BNC). Use a thin Ethernet coaxial cable, which is connected using a T-connector. You need a terminating plug for the "open" end of the T-connector if this is the last computer along the bus.

RJ-45 (telephone style). Connects to a hub using a 10Base-T cable, or directly to another computer using a crossed 10Base-T cable.

AUI (15-pin D-sub). The AUI (Attachment Unit Interface) connector is used on some cards and computers where the manufacturer wanted to leave connection alternaives open. An external network adaptor (transciever) is then needed to interface to the wiring standard of your choice.

AAUI (14-pin mini-D). The AAUI (Apple Attachment Unit Interface) connector is used on some Apple cards and computers. It has the same purpose as the above mentioned AUI connector, but requires a different adaptor due to the smaller connector.

- **CAUTION:** Always turn off the computer/device before plugging in or unplugging an AUI/AAUI transciever.

Activating the TCP/IP Scripting Port

To activate the TCP/IP scripting port, select the "TCP/IP" checkbox in the Scripting Options dialog box, accessed through the "Scripting Options" button in the Preferences dialog box (see page 35).

- **NOTE:** You must configure Open Transport for TCP/IP communication, as shown under "Network Configurations" on page 64, before you can activate TCP/IP in the Scripting Options dialog box. TRAX will display an error message if your computer isn't properly configured.

Configuring a MacOS Client

When using a MacOS computer as a client to talk to TRAX via the network, it needs to be configured in the same way as the TRAX computer (see page 64), but using its own, unique IP number.

Configuring a Windows 95 Client

Follow these steps to configure a computer running Microsoft Windows 95 in order to communicate with TRAX.

1. Install a network communications card into the computer. Most cards support plug-and-play, meaning that Windows 95 automatically recognizes the card when you start the computer.

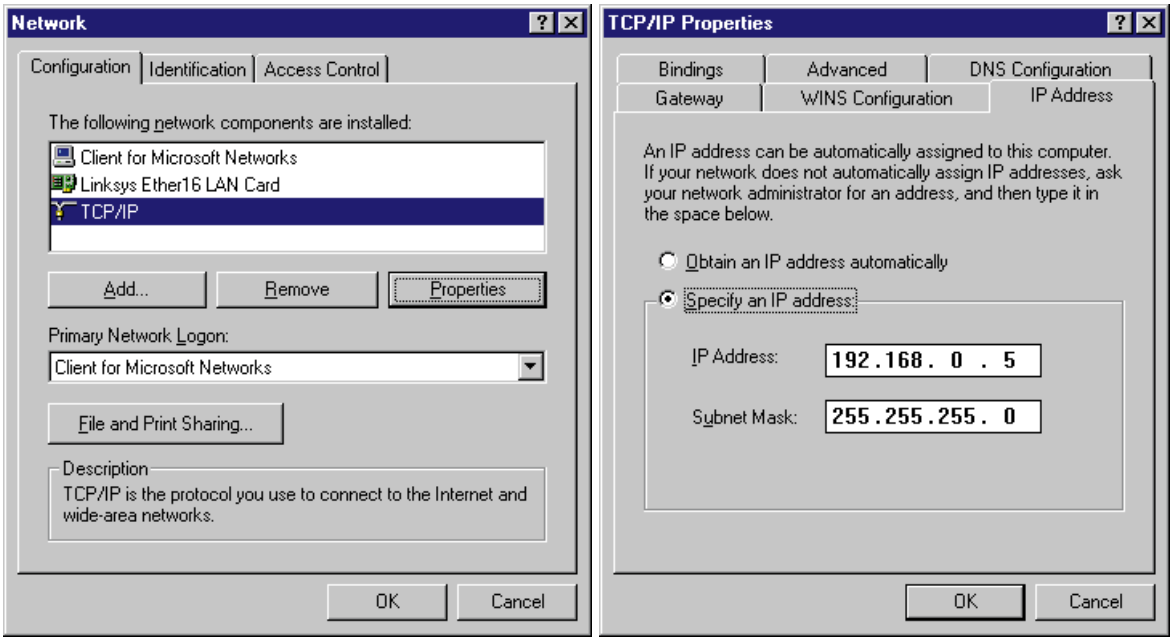
2. Start Windows 95, click the Start button and select Settings: Control Panel. Open the Network control panel.

3. If TCP/IP appears in the list, then skip ahead to point 6.

4. Click Add, select "Protocol" and click Add. Choose "Microsoft" in the Manufacturer list and "TCP/IP" in the Protocol list and click OK.

5. Back in the Network control panel, click Add again, select "Client" and click Add. Choose "Microsoft" in the Manufacturer list and "Client for Microsoft Networks" in the Network Client list. Click OK.

6. Select "TCP/IP" in the Network control panel and click the Properties button. Select the "IP Address" tab, enter the IP number and the subnet mask, and click OK.



Windows 95 Network control panel and TCP/IP protocol properties.

- **NOTE:** Depending on other possible hardware or software issues, you may have to configure additional settings or install more hardware/software. Please consult your computer's documentation for full details on how to configure it for TCP/IP communication.

Configuring a Windows NT 4.0 Workstation Client

Follow these steps to configure a computer running Microsoft Windows NT 4.0 Workstation in order to communicate with TRAX.

1. Install a suitable network communications card into the computer. Most cards support plug-and-play, meaning that Windows automatically recognizes the card the next time you start the computer. If not, you must configure the card and install its driver as instructed in its accompanying documentation.

2. Start Windows NT, click the Start button and select Settings: Control Panel. Open the Network control panel.

3. Select the "Protocols" tab in the Network control panel. If "TCP/IP Protocol" appears in the list, then skip ahead to point 5.

4. Click Add, select "TCP/IP Protocol" in the list and click OK.

5. Select "TCP/IP Protocol" in the list in the Network control panel and click the Properties button. Select the "IP Address" tab, enter the IP number and the subnet mask, and click OK.

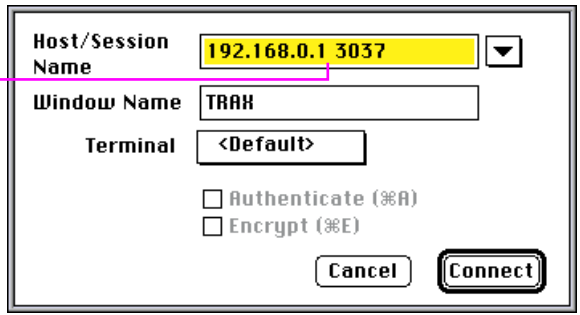
- **NOTE:** Depending on other possible hardware or software issues, you may have to configure additional settings or install more hardware/software. Please consult your computer's documentation for full details on how to configure it for TCP/IP communication.

Testing the Port from a Macintosh

Use a Telnet client application to check the TCP/IP scripting port after successfully activating it in the Scripting Options dialog box in TRAX. MacOS doesn't include any Telnet client application, but there are some free Telnet clients available, such as the NCSA Telnet application used in this example.

On the File menu in NCSA Telnet, choose "Open Connection…" and enter the IP and port number of the computer running TRAX.

Enter the TRAX server's IP number here, followed by a space and the port number 3037.



NCSA Telnet "Open Connection" dialog box.

You can run the Telnet client on the same computer as TRAX, if desired, to test basic TCP/IP communications within the computer. However, to test the actual network, you need to run Telnet on a separate computer.

Once Telnet has established communication with TRAX, you can type a command, such as "get version", into the window to see the response from TRAX.

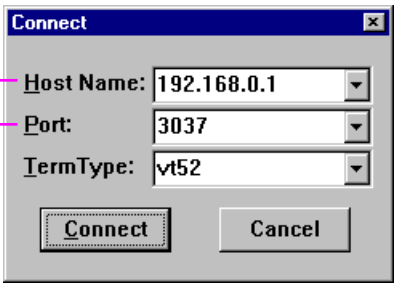
- **NOTE:** Sometimes, the first command given through Telnet may result in an error message from TRAX (eg, "!-1708"). This is caused by the behavior of the Telnet client, and can safely be ignored. Just give the command again.

Testing the Port from Windows

Use the Windows Telnet application to check the TCP/IP scripting port after successfully activating it in the Scripting Options dialog box in TRAX. Start Telnet by clicking the Start button, choose "Run…", and enter "Telnet". Once the Telnet application is started, choose "Remote System…" on the Connect menu. Enter the IP number of the computer running TRAX into the "Host Name" field. Type 3037 into the Port field, and click the Connect button.

Enter the TRAX server's IP number.

Enter port number 3037.



Windows Telnet "Connect" dialog box.

Once Telnet has established communication with TRAX, you can type a command, such as "get version", into the window to see the response from TRAX.

Line Endings

Commands sent to TRAX using the TCP/IP scripting port must be terminated by a line feed (hex 0A), carriage return (hex 0D) or null (00) character, or any combination of those. Replies sent back from TRAX are terminated by a null character.

Queries

Sending queries to TRAX using the TCP/IP scripting port works in the same way as when using the Serial scripting port. See "Sending Queries" on page 51 and "Queries and Transaction IDs" on page 51 for details.

Transaction IDs

Transaction IDs can be used to further improve the reliability of communication between TRAX and the client application. Transaction IDs work the same for the TCP/IP and Serial scripting ports. See "Queries and Transaction IDs" on page 51 for details on how to use transaction IDs.

Errors

If an error occurs while processing a script command, TRAX returns an exclamation point immediately followed by a negative error code. If the command that caused the error had a transaction ID, then that ID will appear at the head of the error message.

Examples:

count divorces
!–1723

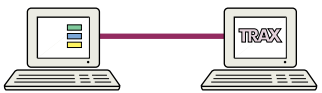
[4139]get the name of device 9581
[4139]!-1719

An error message is always returned if an error occurs, even if the command would normally execute silently. See "Scripting Errors" on page 87 for a description of the error codes.

Network Configurations

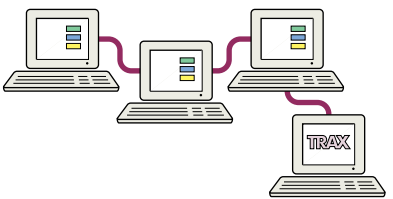
Due to the flexibility of computer networks, and the many ways networks can talk to each other using bridges and routers, there's an almost infinite number of ways TRAX can be combined with other computers. As TRAX uses TCP/IP as its scripting protocol, it is even possible to use the Internet as a way to send command to TRAX (assuming you have the required, globally unique IP number and a suitable Internet connection).

Point To Point



Using a cable from the network port of the TRAX computer to the network port of the client computer, you can set up a point-to-point connection without the need for any additional hardware. This is useful for simple set-ups and testing purposes. If using 10Base-T, you need a crossed cable to set up such a point-to-point connection without a hub.

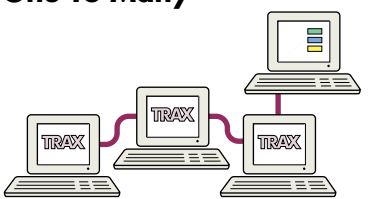
Many To One



You can have a single TRAX computer acting as a server for a number of clients. Each client that connects to TRAX will be assigned its own, individual communications path, ensuring that replies to queries from a particular client will be sent to that client only.

This can be useful, for example, in a visitor center where you may have multiple, computer-based kiosks from which you want to control the presentation environment around each kiosk.

One To Many



It is also possible to have one client talking to multiple TRAX servers. This could, for example, allow you to break up the production and control of presentation into autonomous areas (eg, lighting, sound, video, computer graphics), and then tie these together from a central point.

Scripting Language Overview

In order to write scripts for TRAX, you must first learn its language. This is a relatively simple language, expressed in plain English.

In spoken languages, there are different kinds of words, such as words for actions, things, attributes, etc. For example, the sentence "paint the car red" can be divided as follows:

- "Paint" is the command that specifies what to do.
- "Car" is the object to be manipulated.
- "Red" is an attribute of the car (more specifically, it's a possible value of its "color property").

Likewise, the scripting language in TRAX can be broken up into commands, objects and properties.

◆ **HINT:** You can find a complete list of all commands, objects and properties in TRAX' vocabulary by using the "Open Dictionary" command on the File menu of Apple's Script Editor and choosing the TRAX application.

Objects

This is a list of all the objects in TRAX.

- Application – the TRAX application itself.
- Device – a device in the Device window.
- Status – a status of a device.
- Task – a task in the Task list.
- Timeline – a timeline.
- Cue – a cue in the task list or along a timeline.

Identifying Objects

The sentence “paint the car red” would make sense only if there’s a single car in front of you. If there’s more than one, you need a way to tell which car to paint, or you may get yourself into serious trouble.

In TRAX, only the application object (ie, TRAX itself) is guaranteed to exist on its own. Thus, you don’t need to specify which application object you want to operate on if you, for example, would like to know which version of TRAX you’re talking to. In this case, simply write:

```
get the version
```

And it will return its version number.

For all other objects, there may be more than one, so you must be able to specify on which one to operate. This applies even if there happens to be only a single object (for example, a single cue on a timeline).

There are two ways by which you can identify an object; by name or by number.The name of an objects is usually specified in a dialog box associated with the object. For example, you specify the name of a device in the device’s configuration dialog box.

Assuming that there’s a tape device named “VTR”, you can obtain its type (ie, the name of its device driver, as specified using the Type pop-up menu) using this command:

```
get the type of device "VTR"
```

Note that the name must be specified within quotes.

In some cases you may not know the name of the object, or the name may be ambiguous (eg, there may be many cues along a timeline named “Play”). In this case, you may instead use the object’s number. For a tasks the number is

shown in the leftmost column in the Task window. For a cue, it is its sequential position in the timeline window when shown in list view. For devices, it is its sequential number as it appears in the text list produced by copying all devices and pasting them into a word processor.

Positive numbers specify an object as counted from the beginning of the list. Negative numbers denote objects as counted from the end of the list. Thus, to get the time position of the last cue on a timeline named "Show", you could write:

```
get the time of cue -1 of timeline "Show"
```

Often, object numbers are used as a way to access all objects in a list. This can be accomplished by writing a program loop that increases the number by one for each time around. The total number of objects can be obtained using the count command:

```
count devices
```

This returns the number of devices in the Device window. When accessing devices by their number, the number must be between 1 and the number returned by "count devices", inclusive.

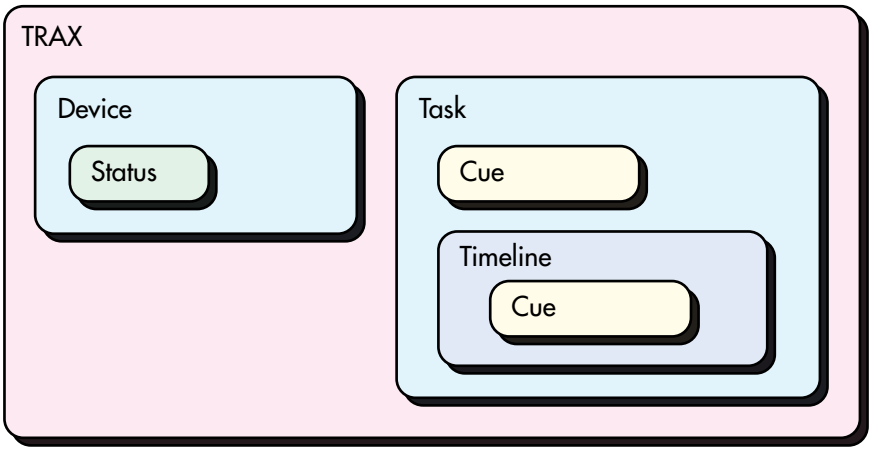
Object Hierarchy

Some objects in TRAX exists inside other objects. For example, cues sit along timelines. This is important when specifying an object. For example, the statement

```
get the time of cue 3
```

Doesn't make sense to TRAX, since you didn't specify where the cue is supposed to be found. The cue could potentially be on any timeline. This applies even if your show only has a single timeline.

Thus, in order to access an object, you must know where it is, and say so explicitly. This is similar to how files on a computer can be stored in folders or subdirectories. There may be many files called "Letter" on your hard disk, so in order to pick the right one, you must know the name of the folder(s) in which it is stored.



TRAX object containment hierarchy.

As you can see, devices and tasks live at the top level inside the application. Thus, you never need to specify a container when accessing devices or tasks. However, to access a cue, you must specify its containing task or timeline. Assuming that task two is a single-cue task (ie, a task where the cue is placed directly in the Action-column in the task list, and not inside a timeline), you can get the name of the cue with the following statement:

```
get the name of cue 1 of task 2
```

Assume that task three has a timeline with six cues, then you can obtain the name of the fifth cue along that timeline as follows:

```
get the name of cue 5 of timeline 1 of task 3
```

Note that you must specify the number of the timeline, even though there can never be more than one timeline in a task.

As you can see, objects are specified beginning with the innermost object and then using the "of" keyword followed by the specification of the enclosing object, again using its name or its number.

Shortcut for Specifying Timelines

As timelines are commonly accessed by scripts, TRAX provides a shortcut for specifying a timeline without specifying its enclosing task. This makes sense since there can only be a single timeline per task anyway. This allows you to treat timelines as if they were top level objects – just like devices and tasks. Thus, the following two commands are equivalent:

```
set the name of timeline 1 of task 3 to "Bill"

set the name of timeline 3 to "Bill"
```

When using this alternative way of specifying a timeline, the number of the timeline is actually the number of its enclosing task.

Likewise, when referring to a task by name, you use the name displayed in the task's Action field, which is actually the name of the cue or timeline that constitutes the task's action. However, as there may be many tasks with the same name (or without name, for that matter), it's often better to specify tasks by number.

Properties

Properties are attributes of object, just as "color" was an attribute of the object "car" in the example earlier in this chapter. Different kinds of objects have different sets of properties. For example, a device has a port property (ie, the value displayed in the "Port" field in the device's configuration dialog box). A cue, on the other hand, has a time property (ie, the time position of the cue).

All objects of the same class have a common set of properties. For example, all devices have a port property – cues don't. The names of all properties for each class of object is described later, and can also be seen by opening the scripting dictionary of TRAX using Apple's Script Editor.

You access a property like this:

```
get the port of device "VTR"
```

This returns the port assignment of the specified device. Note that the name of the property is a keyword (ie, it's part of the scripting language proper), and is not quoted.

Some properties are read only, meaning that they can not be changed through scripting. Such properties are indicated by "[r/o]" in the scripting dictionary of TRAX.

Property Types

Most properties are handled as text. When you set such a property, you must specify it within quotes. For example, the name of a timeline is a text string, so you must specify it like this:

```
set the name of timeline 3 to "Bill"
```

When displayed in Apple's Script Editor, such values are also shown within double quotes.

Some properties are numeric. For example, you can obtain the number of the task that currently owns a device using this command:

```
get the owner of device "VTR"
```

Finally, some properties have values specified by keywords. For example, the priority property of the task object can be set to *none*, *low*, *medium*, *high* or *override*. As these keywords are part of the scripting language, they are never quoted.

Example:

```
set the priority of task 3 to high
```

Specifying the Desired Type

When using the *get* command to obtain the current value of a property, you can specify the desired type of the reply. This is particularly useful for time or boolean properties. For example, if you ask a cue about its time position like this:

```
get the time of cue 1 of timeline "Show"
```

it returns the time as a text string, using the standard TRAX format (eg, 2:12.30). Often, however, it is more practical to get the time as a number, allowing you to directly make calculations on it. This can be accomplished by specifying "as integer" at the end of the get command:

```
get the time of cue 1 of timeline "Show" as integer
```

For time values, this will cause the time to be returned in centiseconds (hundredths of seconds). When setting time values, TRAX accepts both forms, and will automatically do the conversion as required.

Device Status Properties

In addition to its configuration properties (name, type, port...), each device also has a set of status properties. However, the set of status properties vary with the type of device, as well as with the existence of any device-specific modes. Therefore, the names of the status properties could not be included in the fixed part of the scripting language. Instead, the status properties are treated as objects enclosed within their devices.

Refer to these status objects by name. Each such status object has a single property, which is its current value.

Examples:

```
get the value of status "Slide" of device "P12"
```

```
set the value of status "Level" of device "Lamp3" to 75
```

The first example returns the current slide number of the slide projector named P12. The second example sets the brightness of Lamp3 to 75 percent.

The names of the status objects inside each type of device are listed under the heading “Status Pop-up Menu” for each device in the TRAX 3 handbook. The same name is used when referring to the status property from the an expression in the Condition field of the Task window, or from the Expression window.

- **NOTE:** It’s often easiest to pick the name from the object into an expression, and then copy it from there and paste it into the script.

Accessing Device-specific Modes

Device-specific modes, used by some kinds of devices, are referred to like this:

```
get the value of status "Mode:Audio 2:External" of device "LDP"
```

In this example, “Mode” specifies that it is a device-specific mode, “Audio 2” is the name of the mode and “External” is the state of the mode to check.

Furthermore, when referring to a state that has a numeric value, you must specify whether you want to refer to the state itself, or to the state's value, ie:

get the value of status "Mode:Rate:Frames#:State" of device "DFS"

get the value of status "Mode:Rate:Frames#:Value" of device "DFS"

The first example returns true if the "Rate" mode is in its "Frames#" state, false otherwise. The second example returns the numeric value of the "Frames#" state (assuming that the mode is in that state, otherwise an error is returned).

Object Reference

This reference section describes all the TRAX objects that can be scripted, and their enclosed objects and properties.

For each enclosed object, you can determine whether it can be accessed by number, by name or both.

For each property [r/o] specifies that it is read only (ie, it can't be changed using a "set" command). The type of the property's value (eg, string, integer, boolean, etc) is also indicated. Some properties have values specified by keywords, which are then listed, separated by slashes, instead of the type name.

- **NOTE:** The value of boolean properties are specified by the keywords *true* and *false*.

Application Object

Enclosed Objects:

device by number, by name
task by number, by name
timeline by number, by name

Properties:

name string [r/o] — The application's name
version version [r/o] — The version number of the application
frontmost boolean [r/o] — Is this the frontmost application?
run mode *stop/play* — Determines whether tasks start when conditions are met

Device Object

Enclosed Object:

status by name

Properties:

name string [r/o] — The name of the device
number integer [r/o] — The index number of the device
port string [r/o] — The address of the SMARTPAX port
type string [r/o] — Specific brand and model name of device, eg "Sony:LDP-3600D"
class string [r/o] — General class of device, eg "Audio Disc" or "Switch"
owner integer [r/o] — Number of task owning the device, or 0 if no owner

You can also use the word *devices* to refer to objects of this class, eg, together with the *count* command:

```
count the devices
```

Status Object

The status object only has a single property (see the discussion under "Device Status Properties" on page 72):

value integer, string or boolean — The current value of the status property

The value is expressed as an integer, string or boolean, as appropriate for the type of status property specified. For instance, if the status is represented by a checkbox in the device's status window, it will be returned as a boolean.

You can sometimes override the type of the status returned by augmenting the get command with the desired type. For example, if you prefer to have 0 and 1 returned instead of the *false* and *true* keywords, specify "as integer":

```
get the value of status "Hard Snap" of device "P12" as integer
```

Likewise, when setting such a property, you can provide either the *false* and *true* keyword, or the number 0 or 1. Thus, the following two commands are equivalent:

```
set the value of status "Hard Snap" of device "P12" to true
```

```
set the value of status "Hard Snap" of device "P12" to 1
```

When using the *get* command to obtain the time position of a device such as a laserdisc player, TRAX will normally return the value in the same format as specified in the device's status window. For instance, if "Normal" is specified as the time format, it will be returned in a string, like this: "3:42.12". If a frame number format is selected, the current frame number, as displayed in the device's status window, will be returned as an integer.

If you want to obtain the time position of such a device in a predictable format that's independent on the time format selected in the device's status window,

specify “as integer” at the end of the *get* command. In this case, TRAX always returns the time position as an integer, in hundredths of seconds.

Task Object

Enclosed Objects:

cue by number
timeline by number

When referring to the cue or timeline enclosed in a task, always specify by number using the number 1 (a task can only hold a single action item). Alternatively, you can refer to timelines as top level objects, as described under “Shortcut for Specifying Timelines” on page 69.

Properties:

name string [r/o] — The name of the task's action item
number integer [r/o] — The index number of the task
condition string — The task's starting condition
variant *becomes true/while true/becomes false/while false/changes* — Starting condition variant
priority *none/low/medium/high/override* — The task's priority

You can also use the word *tasks* to refer to objects of this class, eg together with the *count* command:

```
count the tasks
```

Timeline Object

Enclosed Object:

cue by number

Properties:

name string — The name of the timeline

run mode *stop/pause/play* — The run mode of the timeline

Cue Object

Properties:

name string — The name of the cue

number integer [r/o] — The index number of the cue

class string [r/o] — General class of cue, eg "Locate" or "Trigger"

assignment string — Devices assigned to the cue, as a string of names separated by semicolons

contents string — The cue's contents, as a string, as displayed in list view

time string — The cue's time position along a timeline

track integer — The cue's track number, 1…16

The *contents* and *assignment* properties are specified as strings using the same syntax as used by a timeline window in list view. Thus, the easiest way to create these properties is to enter the cue into a timeline window, copy it, paste it into the script and then extract the relevant parts.

You can use the word *cues* to refer to objects of this class, eg, together with the *count* command:

```
count the cues of timeline "Show"
```

Commands

This reference section describes all TRAX scripting commands and their parameters. Most commands do not return any data except if an error occurs, in which case they return a negative error code (see "Scripting Errors" on page 87).

The first parameter of most commands identifies the object. This parameter is called a reference, and is specified by the kind of object (ie, device, cue, timeline, etc), its name or number, and the reference to its enclosing object (eg, the name of the timeline containing a cue). See "Identifying Objects" on page 66 and "Object Hierarchy" on page 67.

Some commands take additional parameters, which are then introduced by a keyword followed by the parameter value. For example, the Open command takes two parameters. The first parameter is the name of the file to open. The second parameter – introduced by the keyword *saving* – specifies whether to save the old show before opening the new one:

```
Open file "Second Show" saving no
```

Here, the string "Second Show" is a reference to the object to be opened (ie, the name of the show file). The keyword *saving* introduces the second parameter, whose value is given by the keyword *no*. Note that keywords are never quoted – they are part of the TRAX scripting language.

In some cases, parameters are optional. These parameters are shown within square brackets in the command reference below. You don't enter these square brackets when using the command – they are only used in the reference to indicate that the parameter is optional. Either you include the parameter's introducing keyword and value, or you omit both.

Get

get reference — The object property to be returned
[***as*** type class] — The desired type for the data
Result: anything — The data from the object's property

Obtains the current value of the specified property. The type of the returned data is normally determined by the type of the property. You can use the *as* keyword to override this behavior if desired.

Examples:

get the version

Returns the version number of the TRAX application.

get the run mode of timeline "Main Show"

Returns the current run mode of the specified timeline, using the keyword *stop, pause* or *play* .

get the time of cue -1 of timeline "Main Show" as integer

Returns the time of the last cue on the specified timeline. Normally, this would be returned as a string in the standard time format (eg, 3:12.24). In this case, the optional parameter introduced by the *as* keyword overrides this, causing it to be returned as an integer, in centiseconds, which makes the value easier to use for calculations.

Set

set reference — The object to change
to anything — The new value

Changes the value of the specified property to the new value. Some properties are marked "[r/o]", and can not be changed using the *Set* command.

Normally, the *to* parameter should have the same type as the property you try to set (ie, don't use a string if the property is numeric). TRAX will attempt to convert the value to the desired type. If it fails, an error message will be returned.

Examples:

set the run mode to play

Makes TRAX switch to run mode (ie, activates the play button in the Task window).

set the name of timeline "Eric" to "Lambert"

Changes the name of the specified timeline.

set the value of status "Transport:Play" of device "VTR1" to true.

Puts VTR1 into play mode.

set the time of cue 3 of timeline "Room 5" to 450

Moves the specified cue to the specified time position on its timeline. Note that the time is given as an integer, which causes TRAX to interpret it as centi-seconds (hundredths of seconds). Alternatively, you can use a string (within quotes) to specify a time, in which case it will be interpreted as being in the most natural format for the property being set. For cues, this is always in the standard time format "HH:MM:SS.hh" (where "hh" is hundredths of seconds). For device status properties, such as the position of a laserdisc, it is the format specified on the time format pop-up menu in the device's status window.

Count

count reference — The objects to be counted (eg, devices)
Result: integer — The number of objects

Counts specified objects.

Examples:

```
count devices
```

Returns the number of devices in the Device window.

```
count tasks
```

Returns the number of tasks in the Task window.

```
count cues of timeline 1
```

Returns the number of cues on the timeline of task number 1. Note that the short form for specifying timelines is used (see "Shortcut for Specifying Timelines" on page 69).

Perform

perform type class — Cue
    with properties record — The properties of the cue

Performs the specified cue. You specify the properties of the cue (ie, its contents and device assignment) within curly brackets following the *with properties* keyword. Each property is introduced by its keyword followed by a colon. Multiple properties are separated by a comma.

Examples:

perform cue with properties {contents:"Set Scene; Mode Multiple; Rate 3.5; Level 40'91'68", assignment:"Ch1; Ch2; Ch3"}

Performs a Set/Fade cue to fade device Ch1, Ch2 and Ch3 to the specified levels at 3.5 seconds. The *contents* and *assignment* properties are specified as strings using the same syntax as used by a timeline window in list view. Thus, the easiest way to create these properties is to enter the cue into a timeline window, copy it, paste it into the script and then extract the relevant parts.

perform cue with properties {contents:"Control; Run; JumpCtrl Section 2; Timeline Rimini"}

Causes the timeline named "Rimini" to jump to the Control cue named "Section 2" and run from there.

Make

make
new type class — The class of the new object (eg, timeline or cue)
[***in*** reference] — Where to create the object (eg, in which timeline)
with properties record — The properties of the new object
Result: reference — A reference to the new object

Creates a new object of the specified type in the specified container and with specified properties. You can only create cues, timelines and tasks (tasks are created implicitly when you create a timeline or a cue without specifying the *in* parameter). The properties are specified within curly brackets following the *with properties* keyword. Each property is introduced by its keyword followed by a colon. Multiple properties are comma separated.

The command returns a reference to the created object. This is useful, for example, when creating a timeline, as you then typically want to proceed by adding cues to that newly created timeline. By using the reference returned by

the *make* command, you know that you refer to the right timeline, even if there happen to be multiple timelines with the same name.

Examples:

make new timeline with properties {name:"Bob"}

Creates an empty timeline and gives it the name "Bob". Returns a reference to the newly created timeline.

make new cue with properties {contents:"Trigger; Play", assignment:"VHS1"}

Makes a play cue in the task list (ie, a single-cue task). The cue is assigned to the device named "VHS1". Returns a reference to the newly created cue in the form of a reference:

cue 1 of task 5

make new cue with properties {contents:" Trigger; Stop", assignment: "VHS1", time: "3:10.12", track: 2} in timeline "Bob"

Makes a stop cue assigned to device "VHS1". This cue is created at the time 3:10.12, on the second track of the timeline named "Bob". It returns a reference to the new cue. Note that the time is specified as a string. Alternatively, you may specify it as an integer, in which case it will be interpreted as centiseconds.

Delete

delete reference — The object to be deleted (eg, timeline, task or cue)

Deletes the specified object. Can be used to delete cues, tasks and timelines.

Examples:

delete cue -1 of timeline "Bob"

Deletes the last cue of the timeline named “Bob”.

delete timeline "Bob"

Deletes the entire timeline named “Bob”. If multiple timelines with the same name exist, only the first one will be deleted.

- **NOTE:** When you delete a timeline or a cue in the task list, the entire task is deleted.

Open

open file filename — Name of show-file to open
[***saving*** *yes/no/ask*] — Specifies whether to save the current show

Opens a new show file. As TRAX can only have a single show file open at a time, the current show file will be closed. The optional *saving* parameter allows you to specify whether to save any changes to the current show file before opening the new one.

Examples:

open file "Show 2"

Opens the show named “Show 2” in the current TRAX folder. Uses the default value for the optional *saving* parameter, as it was not specified (the default value is *ask*).

open file "HD:TRAX:Shows:Star Fantasy:Part 1" saving yes

Opens the show named “Part 1”, which is located on the disk named “HD”, inside the specified folder hierarchy. If there are any changes to the current show, those will be saved to disk before opening the new show.

Save

save

[***in file*** filename] — The file in which to save the show

Saves the current show to disk. If used without the optional *in file* parameter, it will be saved under its current file name.

Examples:

save

Saves the show using its current filename.

save in file "HD:TRAX:Shows:Star Fantasy:Part 3"

Saves the show using the name “Part 3” onto the disk named “HD”, inside the specified folder hierarchy.

Quit

quit

[***saving*** *yes/no/ask*] — Specifies whether to save the curren show

Quits TRAX. Using the optional *saving* parameter, you can specify whether to save any changes made to the current show before quitting.

▼ **IMPORTANT:** If you use this command through the TCP/IP or Serial scripting ports, you will lose contact with TRAX, and will not be able to send any further commands.

Replies

Some commands return replies. When using the AppleScript scripting port, the reply can be picked up from *the result*.

Example:

```
get the value of the status "Slide" of device "P14"
put the result into slideNumber
```

These AppleScript lines query TRAX about the current slide number of projector "P14", and store the result in a variable named slideNumber.

When using the TCP/IP or the Serial scripting ports, replies are sent back the same way, and are preceded with an equal sign. For example, if you send the command

```
get the value of the status "Slide" of device "P14"
```

to TRAX through its Serial scripting port, it returns something like this:

```
=12
```

If the command sent to TRAX through the TCP/IP or the Serial scripting ports was tagged with a transaction ID, the same transaction ID will be inserted in front of the reply (see "Queries and Transaction IDs" on page 51):

```
[4117]get the value of the status "Slide" of device "P14"
[4117]=12
```

The *get* command allows you to override the returned type in some cases by using the *as* parameter (see "Get" on page 79).

Scripting Errors

If a command fails, an error code is returned. This applies even if the command normally doesn't return anything.

AppleScript Error Messages

When using AppleScript, this normally results in the script being aborted and an error message displayed. This message may include the error code number and/or a descriptive string, depending on the type of program used to send the script and the type of error that occurred. If you anticipate errors at some point in your script, you can handle those gracefully, without causing the script to be aborted, using a "try" statement. Please refer to your AppleScript documentation for details.

TCP/IP and Serial Scripting Port Error Messages

If an error occurs while processing a script command, TRAX returns an exclamation point immediately followed by a negative error code. If the command that caused the error had a transaction ID, then that ID will head the error message.

Examples:

count divorces
!–1723

[4139]get the name of device 9581
[4139]!-1719

Error Codes

The table below lists the error codes returned by TRAX. In AppleScript, additional errors may occur, not directly related to the internal script processing of TRAX.

Code	*Description*
–1717	Unknown Command.
–1708	Command failed (general error code when others don't apply).
–1728	Unknown object name (when accessing an object by name).
–1719	Index out of range (when accessing an object by number).
–1723	Unknown property or keyword.
–1703	Invalid type of data (eg, got a string "XYZ" when expected an integer).
–1718	Value not available in the current state of a device-specific mode.
–1713	Temporrily unable to handle scripting command (eg, while in a modal dialog box).

Scripting Examples

Following are some scripting examples that you can use as a starting point for learning how to script TRAX. The examples use AppleScript, which allows you to run them on the same computer as TRAX. Furthermore, most AppleScript editors include good editing, testing and debugging support, which may be helpful when learning the scripting language.

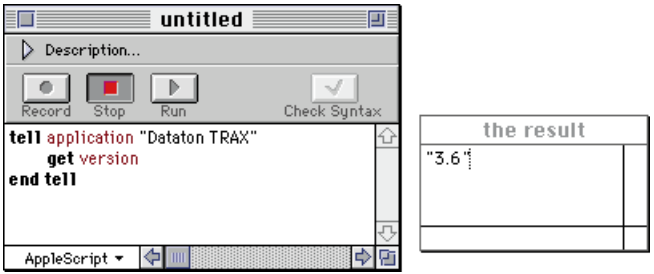
You may want to open the scripting dictionary of TRAX and keep it open as a quick reference. This is done using the "Open Dictionary" command on the File menu in most script editors.

If you later want to use your scripting commands through the TCP/IP or Serial scripting ports, remember that only those commands specifically directed to TRAX can be used. Other commands and control structures will have to be handled separately on the client platform. The section titled "Scripting TRAX from Macromedia Director" on page 103, gives some examples on how this can be done using Lingo, the internal scripting language of Director.

While AppleScript excels at editing, testing and debugging scripts, it executes the scripts very slowly. The TCP/IP or Serial scripting ports don't suffer from this limitation. Furthermore, as the client in those cases doesn't run on the same computer as TRAX, the impact on TRAX itself is negligible.

Your First Script

The script below queries TRAX about its version number.



Of the three lines that make up this script, only the second line is seen by TRAX. The first line tells AppleScript that it's supposed to talk to the application named "Dataton TRAX". You may try substituting it with "Finder" to see how the script then retrieves the Finder's version number.

▼ **IMPORTANT:** The above script assumes that the name of the currently running copy of TRAX is "Dataton TRAX". If your TRAX application has a different name, the script won't find it. The name of the currently running TRAX application can be seen on the application menu, located to the far right in the menu bar.

The last line marks the end of the "tell application…" block. This is needed since you may put several commands to TRAX inside the tell block. Again, this command is for AppleScript's internal use, and is not related to TRAX.

This script activates the output of a Switch device named "Door".

```
tell application "Dataton TRAX"
    set the value of status "Output On" of device "Door" to true
end tell
```

Note how you refer to the status of a device using *the value of status* followed by the name of the status property within quotes. See "Device Status Properties" on page 72 for more details.

To turn off the same status property, set it to *false*:

```
tell application "Dataton TRAX"
    set the value of status "Output On" of device "Door" to false
end tell
```

You can learn the name of a device's status properties by using the device's status pop-up menu, as shown under "Device Status Linkage Menus" on page 120 in the TRAX 3 handbook.

The next example sets the Level of a lamp device named Ch1 to 45 percent:

```
tell application "Dataton TRAX"
    set the value of status "Level" of device "Ch1" to 45
end tell
```

Some status properties have named states. This applies to, for example, the "Transport Mode" of a tape device. You activate the desired state by setting it to *true*:

```
tell application "Dataton TRAX"
    set the value of status "Transport:Play" of device "VHS" to true
end tell
```

When a mode has multiple, named states – such as the Transport mode of a tape device, you can not turn off the current state by setting it to false. Instead, you turn it off by activating another state of the same mode. The script below stops the tape device started in the previous example by activating the Stop state of the Transport mode instead:

```
tell application "Dataton TRAX"
    set the value of status "Transport:Stop" of device "VHS" to true
end tell
```

▼ **IMPORTANT:** It is not possible to set the status of a device currently being owned by a task in the TRAX task list. You can determine if a device is owned by getting its *owner* property. If the device is owned, you get the task number of its owner. If the device is not owned, you get the number 0. See "Device Ownership" on page 121 in the TRAX 3 handbook for more details.

Getting Device Status

In the same way as you can set the status of a device using the *set* command, you can retrieve it using the *get* command:

```
tell application "Dataton TRAX"
    get the value of status "Level" of device "Ch1"
end tell
```

This returns the current level of the lamp device named Ch1. The value is displayed in the result window of Apple's Script Editor.

If you want to use the result in your script you must put it into a variable, which can then be used in expressions and as a parameter to other commands:

```
tell application "Dataton TRAX"
    get the value of status "Level" of device "Ch1"
    put the result into lightLevel
    set the value of status "Level" of device "Ch2" to lightLevel
end tell
```

- **NOTE:** The capability of storing the result of the *get* command into a variable, and then using that variable in the *set* command is specific to AppleScript. When using the TCP/IP or Serial scripting ports, this will have to be managed separately by the client program.

If you need to know the state of a status property with named states, such as the Transport status of a tape device, you must specify the expected state explicitly:

```
tell application "Dataton TRAX"
    get the value of status "Transport:Play" of device "VHS"
end tell
```

This returns *true* if the device is currently playing, *false* if it isn't.

Performing Cues

The example below performs a Set/Fade cue to fade device Ch1, Ch2 and Ch3 to the specified levels at 3.5 seconds.

```
tell application "Dataton TRAX"
    perform cue with properties {¬
        contents:"Set Scene 3; Mode Multiple; Rate 3.5; Level 40'91'68", ¬
        assignment:"Ch1; Ch2; Ch3"¬
    }
end tell
```

The *contents* and *assignment* properties are specified as strings using the same syntax as used by a timeline window in list view. Thus, the easiest way to create these properties is to enter the cue into a timeline window, copy it, paste it into the script and then extract the relevant parts.

The next example uses a Control cue to cause the timeline named "Rip" to jump to the Control cue named "S2" and run from there.

```
tell application "Dataton TRAX"
    perform cue with properties {¬
        contents:"Control Run Section;  Run; JumpCtrl S2; Timeline Rip"¬
    }
end tell
```

▼ **IMPORTANT:** The line continuation character "¬" (option-Return), used in the above examples, is specific to AppleScript. When using the TCP/IP or Serial scripting ports, the entire *perform* command must be presented to TRAX as a single line.

Controlling Timelines

This example starts the specified timeline:

```
tell application "Dataton TRAX"
    set the run mode of timeline "Main Show" to play
end tell
```

Likewise, you can query the current run mode of a timeline like this:

```
tell application "Dataton TRAX"
    get the run mode of timeline "Main Show"
end tell
```

This returns the keyword *stop*, *pause* or *play*.

▼ **IMPORTANT:** Remember that pausing a timeline is not the same thing as stopping it. A paused timeline retains ownership of all its devices, allowing it to resume at any time. When you stop a timeline, it releases all its devices, allowing them to be used by other timelines, panels or scripts.

Controlling the Task List

Just as you can start and stop individual timelines, you can also start and stop the scanning of the task list:

```
tell application "Dataton TRAX"
    set the run mode to play
end tell
```

This causes TRAX to start scanning its task list, starting eligible tasks.

▼ **IMPORTANT:** In order to retain AppleScript control while running timelines or the task list, you must have activated AppleScript in the Scripting Options dialog box. See page 35 for details.

Creating Timelines and Cues

The script below creates a new timeline named "Bob".

```
tell application "Dataton TRAX"
    make new timeline with properties {name:"Bob"}
end tell
```

It returns a reference to the newly created timeline. This can be used if you later want to add cues to that timeline. This ensures that the cues are added to the newly created timeline, event if there are multiple timelines named "Bob" (which is possible, although not advisable).

The next example uses the reference returned to add two cues to the new timeline:

```
tell application "Dataton TRAX"
    make new timeline with properties {name:"Bob"}
    put the result into timeLineRef
    make new cue in timeLineRef with properties {¬
        contents: "Note 1; Text First note",¬
        time: 0 ¬
    }
    make new cue in timeLineRef with properties {¬
        contents: "Note 2; Text Second note",¬
        time: 200 ¬
    }
end tell
```

The first cue is put at time 0:00.00, and the second at time 0:02.00 As the time properties are specified as numbers, TRAX interprets them as centiseconds. If you want to specify them using the HH:MM:SS.hh format, you must quote the time, since AppleScript doesn't know how to deal with time values in this format.

The cues are put on the currently selected track (ie, the first track, since the time-line was just created). You can specify the track and other properties of the cues as well, if desired. See "Cue Object" on page 77 for a description of the cue properties.

Opening and Saving Shows

The script below tells TRAX to open the show file named "Show 2", which is assumed to be located in the current folder.

```
tell application "Dataton TRAX"
    open file "Show 2"
end tell
```

As the "current folder" concept is somewhat vague, you may prefer to specify the location of the file explicitly:

```
tell application "Dataton TRAX"
    open file "HD:TRAX:Shows:Star Fantasy:Show 2" saving yes
end tell
```

This specifies the full path to the file, beginning with the name of the disk (HD), and ending with the name of the file. You may have to adapt the script according to the structure and name of your own hard disk.

- **NOTE:** The second script above specifies that TRAX should save any changes to the current show before opening the new one. If you don't specify this by appending *saving yes* or *saving no* to the command, TRAX may put up a dialog box asking you what to do, which may not be desirable while executing a script.

You can explicitly tell TRAX to save the current show into a file of your choice using this script:

```
tell application "Dataton TRAX"
```

```
    save in file "Moxie"
end tell
```

This saves the show into the current folder. You have to enter the full path name, beginning with the name of the disk, if you want to specify explicitly where to store the show.

If you just want to save the show into its current file (assuming that it has been opened or saved previously), you use the *save* command without the *in file* parameter. Note, however, that this will cause TRAX to display a dialog box if the show has never been saved.

Scripting with a Word Processor

One of the chief advantages of AppleScript is that it allows you to script two or more applications at the same time. This can be used, for example, to copy data to or from TRAX and other applications.

This first example shows how the contents of all Note cues on the first timeline can be extracted as text paragraphs using Apple's Scriptable Text Editor (a simple word processor, included with Apple's scripting software).

```
set AppleScript's text item delimiters to ";" -- To extract cue content items
tell application "Dataton TRAX"
    count cues of timeline 1
    copy the result to numCues
    repeat with cueNum from 1 to numCues
        get the contents of cue cueNum of timeline 1
        copy the result to cueContents
        if the first word of cueContents is "Note" then
            copy text item 2 of cueContents to cueContents
            tell application "Scriptable Text Editor"
                make new text with data cueContents & return
                delete characters 1 through 6 of the result -- " Text "
```

```
            end tell
        end if
    end repeat
end tell
```

This script contains two tell statements, one that targets “Dataton TRAX” to extract the cues, and another one that targets "Scriptable Text Editor", to add the text. A repeat loop causes the script to scan through all the cues along the timeline. Only the contents of the cues that begin with the word “Note” are kept. The desired information is then added to the text using the *make* command of the scriptable text editor.

The next example performs the opposite function, copying text paragraphs from the scriptable text editor into Note cues in TRAX. This script is designed to be easy to modify by storing the names of things into clearly documented variables up front, making it easier to re-use the script.

In order to use this script, you must have everything set up as specified by the variables at the top; ie, the name of the timeline to which the Note cues are to be added must be “Timeline 1”, and there must be a Text device named “Notes” in the Device window. You may of course modify those variables to suit your requirements.

```
-- Name of the application to get the text from
copy "Scriptable Text Editor" to scriptableTextEditor

-- Name of the timeline to receive the Note cues
copy "Timeline 1" to timelineName

-- Name of device to attach the Note cues to
copy "Notes" to noteDeviceName
```

```
-- Distance between note cues, in hundredths of seconds
copy 200 to timeDistance

tell application scriptableTextEditor
    count paragraphs of window 1
    copy the result to paraCnt
    copy 0 to cueTime
    repeat with paraNum from 1 to paraCnt
        copy paragraph paraNum of window 1 to noteText
        tell application "Dataton TRAX"
            make new cue in timeline timelineName with properties {¬
                time:cueTime, ¬
                assignment:noteDeviceName, ¬
                contents:"Note " & paraNum & "; Text " & noteText¬
            }
            copy cueTime + timeDistance to cueTime
        end tell
    end repeat
end tell
```

- **NOTE:** The maximum amount of text that can be stored inside a Note cue is 255 characters. Paragraphs longer than that will be truncated.

Notice how the *&* operator is used in assembling the contents of the cue. This is done by concatenating "Note ", the paragraph number, "; Text " and the contents of the paragraph. AppleScript assembles all this, and passes it along to TRAX. Assuming that the variable paraNum contains 5 and the variable noteText contains the text "The presenter enters the room stage left", this will result in the following text being passed to TRAX as the contents property for the new cue:

"Note 5; Text The presenter enters the room stage left"

When viewing the contents of the Note cue with the timeline window in TRAX set to list view, this is exactly the format in which it is displayed.

▼ **IMPORTANT:** When assembling a string to be used as a parameter to TRAX, you must make sure that the resulting string conforms to what TRAX expects. If not, TRAX will return an error message.

Scripting with a Database

The script below extracts data from a FileMaker Pro database, used to keep track of songs on a set of CDs. There's one record per CD in the database, each containing the following fields:

Title: The title of the song.
Disc: The number of the disc.
Track: The number of the track.
Length: The duration of the song, in the format MM:SS.

The purpose of the script is to build a timeline that will play the songs in the order they appear in the database. For each song, two cues are added; a Locate cue, containing the name and number of the song, and a Play cue to start playing it. The cues are spaced along the timeline according to the duration of each song. The Locate and Play cues are both assigned to a device named "CD", which is assumed to exist in the Device window in TRAX.

```
tell application "Dataton TRAX"
    make new timeline with properties {name:"PlayList"}
    copy the result to playList -- A ref to the new timeline
end tell

tell application "FileMaker Pro"
    set AppleScript's text item delimiters to ":" -- To split minutes:seconds
    count every record
    copy the result to numRecs
```

```
    copy 0 to currTime -- in centiSeconds
    repeat with recordNumber from 1 to numRecs
        go to record recordNumber -- To see the bar move
        copy cells of record recordNumber to recordData
        copy item 1 of recordData to title
        copy item 2 of recordData to disc
        copy item 3 of recordData to songNumber
        copy item 4 of recordData to songLength
        copy text item 1 of songLength as integer to mins
        copy text item 2 of songLength as integer to secs
        copy (mins * 60 + secs) * 100 to songDuration -- in centiSeconds
        tell application "Dataton TRAX"
            make cue in playList with properties { ¬
                contents:"Locate; Number " & songNumber,  ¬
                time:currTime,  ¬
                assignment:"CD" ¬
            }
            copy currTime + 200 to currTime -- Time needed to locate
            make cue in playList with properties { ¬
                contents:"Trigger Play;  Play",  ¬
                time:currTime,  ¬
                assignment:"CD" ¬
            }
            copy currTime + songDuration to currTime
        end tell
    end repeat
end tell
```

- **NOTE:** The disc number is not taken into account by the above script. This could, for example, be handled by a separate CD-changer device, controlled by a Serial driver. Adding those cues is left as an exercise for the reader.

Scripting TRAX from Macromedia Director

As an example of how the scripting capabilities of TRAX can be used from other applications, TRAX comes with a plug-in for Macromedia Director. This allows TRAX to be scripted from Lingo, the scripting language of Director.

This scripting interface uses the TCP/IP scripting port of TRAX, allowing it to be used from Director running on either MacOS or Windows. The computer running Director and the computer running TRAX must both be connected to the same network, configured to use the TCP/IP protocol.

- **NOTE:** It is not possible to use this solution to run TRAX and Director on the same computer.

The following instructions and examples assume that you have a basic understanding of Director and Lingo.

Installing the TRAX Xtra

In addition to configuring the network hardware and software, as described under "Configuring Open Transport" on page 54 and "Configuring a Windows 95 Client" on page 58, you must also install the "TRAX Scripting" plug-in on the computer running Director.

Macintosh. Copy the file "TRAX Scripting Xtra.Mac" to the Xtras folder, located in the same folder as the Director application. Restart Director.

Windows 95/NT. Copy the file "TRAX Scripting Xtra.x32" to the Xtras folder, located in the same folder as the Director.exe application. Restart Director.

- **HINT:** If the file is on your Macintosh, you may be able to use the network to copy it to your PC. If not, first copy it to a PC formatted diskette, and then transfer it to the PC. The control panel "PC Exchange" is required in order to use PC-formatted diskettes on a Macintosh.

Opening and Closing the Connection

You typically open the connection to TRAX in the startMovie handler. This handler must be stored in a script cast member of type "Movie".

```
on startMovie
    global t1
    if objectP(t1) then exit -- In case stopMovie was never called
    set t1 = new(xtra "TRAX")
    if not objectP(t1) then
        alert "Can't find the TRAX Xtra"
        exit
    end if
    ConnectTRAX t1, "192.168.0.1" --Enter TRAX' IP number here
    if the result < 0 then alert "Can't find the TRAX server"
end
```

▼ **IMPORTANT:** If your movie already has a "on startMovie" handler, append the contents of the handler shown above to the end of the already existing handler. Don't add it as a new handler.

The handler shown above is executed automatically whenever the movie is started (ie, when you click the play button in Director's Control Panel window). It creates a global object named t1, which will be used as a reference to the TRAX server. It attaches t1 to the TRAX extra. Finally, it attempts to connect to the TRAX server using the IP number of the computer running TRAX as a parameter, within quotes. The remaining lines in the handler deals with various kinds of errors that may occur.

A stopMovie handler is used to disconnect from the TRAX server when the movie is stopped. This handler should be placed in the same script cast member as the startMovie handler (see next page).

```
on stopMovie
    DisconnectTRAX t1
    set t1 = 0 -- So we know we disconnected properly
end
```

◆ **HINT:** If desired, you can open connections to multiple TRAX servers, each running on its own computer. Simply duplicate the contents of the startMovie and stopMovie handlers shown above for each server, and choose another global variable name than t1 (eg, t2, t3, t4...).

Sending Commands

You send commands to TRAX using TellTRAX. This can, for example, be put inside a mouseUp handler of a button.

```
on mouseUp
    TellTRAX t1, "set the value of status ´Level´ of device´Ch1´ to 65"
end
```

The script command to TRAX is sent as a string, within double quotes, as the second parameter to the TellTRAX command. The first parameter is the global variable used to refer to the TRAX server, which was assigned in the startMovie script.

▼ **IMPORTANT:** You can't use double-quotes to delimit strings inside the command. See "Resolving Quoting Conflicts" on page 106 for more details.

The TellTRAX command doesn't check for any error from TRAX. If you need to know if the command succeeded, use QueryTRAX to send the command instead, which will wait for the response from TRAX and return any error codes back to you as a negative number in *the result*.

Resolving Quoting Conflicts

Many scripting commands to TRAX use quoted strings. Normally, you quote strings using straight double-quotes. However, as Director also uses straight double-quotes to delimit the entire script string, those can't be used to delimit strings inside the script string. Instead, use a grave accent to delimit strings within the command, as shown around the object names "Level" and "Ch1" in the example under "Sending Commands" on page 105.

Sending Queries

Use QueryTRAX to get data from TRAX. QueryTRAX can also be used to send commands to TRAX if you want to make sure that no error occurred while executing the command. The handler below obtains the current level of the device named "Ch1", and displays it in Director's Message window.

```
on mouseUp
    QueryTRAX t1, "get the value of status ´Level´ of device ´Ch1´"
    if stringP(the result) then -- Got the requested value
        put the result -- Display the value in the Message window
    else
        alert "TRAX Error " & the result
    end if
end
```

QueryTRAX waits for the reply to come back from TRAX. When it returns, *the result* contains either the data received back from TRAX (as a string), or a negative error code (as a number). If the command executes successfully, but without returning any data, *the result* will contain an empty string. Thus, you can use the built-in Lingo function stringP() to check for errors. If stringP(the result) returns false, that means that an error occurred and *the result* contains the negative error code (see "Scripting Errors" on page 87).

Optimizing Performance Using Ask and Retrieve

Sometimes, the delay caused by waiting for the reply to QueryTRAX to arrive may be unacceptable. In this case, you can use AskTRAX and RetrieveTRAX instead. You use AskTRAX in the same way as you use TellTRAX – ie, it will send the query to TRAX, but it will not wait for any reply to come back. Instead, you use RetrieveTRAX to pick up the reply at a later time. In the meantime, Director can go about its business uninterrupted.

Furthermore, as AskTRAX returns immediately, without waiting for the reply to come back from TRAX, it allows you to fire off several queries in rapid succession, rather than one at a time. This can significantly speed up the process of getting several pieces of information from TRAX.

In order to help you match the right reply to the right question, AskTRAX returns a transaction ID number in *the result*. This is a positive integer greater than zero. You must supply this transaction ID number to RetrieveTRAX in order to retrieve the reply to the question sent by the corresponding AskTRAX.

RetrieveTRAX will return the result of the query, as a string, or an empty string if the query didn't return any data (for example, if you used AskTRAX to send a set command, which doesn't return anything except if an error occurs). If the result hasn't arrived yet, RetrieveTRAX returns the number 0. Thus, you can use the built-in Lingo function stringP() to check for any data. If stringP(the result) returns false, you can look at the returned number to see if it is 0 (meaning that the reply hasn't arrived yet), or a negative error code.

Example:

```
on idle
    global t1, transID -- transID keeps track of question in progress
    if transID = 0 then -- No question in progress
        AskTRAX t1, "get the value of status ´Level´ of device ´Lights´"
        if the result < 0 then
            alert "AskTRAX error " & the result
        else
            put the result into transID -- Needed to get reply later
        end if
    else
        RetrieveTRAX t1, transID
        if stringP(the result) then -- Got the reply
            put the result into field "Display"
            put 0 into transID
        else-- No reply yet - see if I got a negative error code
            if the result < 0 then --Was an error
                alert "RetrieveTRAX error " & the result
                put 0 into transID
            else
                -- Must continue waiting for the answer
            end if
        end if
    end if
end idle
```

Put this handler in some suitable place, such as in a movie script. If your movie already has an idle handler, insert the body of the code above into that handler rather than adding it as a new idle handler. You also need to add the transID global variable to your startMovie handler and set it to 0 there.

Director Xtra Error Codes

The following error codes can be returned by the Director Xtra. These are in addition to the scripting error codes listed under “Scripting Errors” on page 87.

Code	*Description*
-1	Insufficient OS networking support (ConnectTRAX)
-2, -3, -4	Failed establishing network connection (ConnectTRAX)
-5	Can’t find TRAX server on the network (ConnectTRAX)
-6	Communication error (TellTRAX, QueryTRAX, RetrieveTRAX)
-7	Too many transactions in progress (AskTRAX)
-8	No such transaction in progress (RetrieveTRAX)
-9	Bad data in reply (QueryTRAX, RetrieveTRAX)
-10	Insufficient memory
-11	Bad parameter (eg, invalid IP number)

Scripting TRAX from Microsoft Windows

In addition to the Xtra plug-in for scripting TRAX from Macromedia products, TRAX also comes with an ActiveX plug-in, called TRAXSCRIPT. This allows you to script TRAX from many Windows applications via a TCP/IP compatible network. Examples of ActiveX-compatible applications include Microsoft Visual Basic, PowerPoint and Excel, as well as Asymetrix Toolbook and Borland C++ Builder.

Most of the Microsoft products use Visual Basic as their internal scripting language. The following examples demonstrate how to script TRAX from PowerPoint. Excel and Visual Basic are very similar. If you're using another application or programming environment, please refer to its documentation for details on how to interact with ActiveX software components.

Installing TRAXSCRIPT

The TRAXSCRIPT ActiveX plug-in comes with TRAX. Locate the file named TRAXSCRIPT.DLL, copy it onto a PC formatted diskette, and move it to your PC.

PC Exchange

- **NOTE:** In order to copy the file onto a PC formatted diskette, you must have the "PC Exchange" control panel installed on your MacOS computer.

Put the TRAXSCRIPT.DLL file in the System (Windows 95) or System32 (Windows NT) folder of your PC, or any other folder of your choice.

Registering TRAXSCRIPT

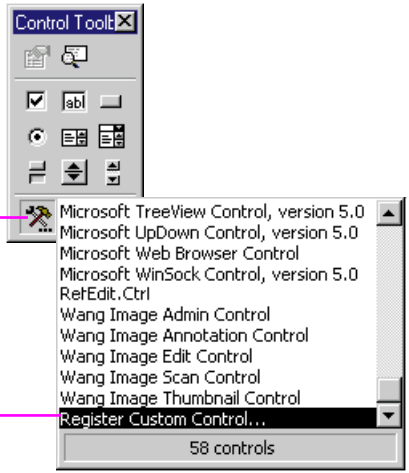
Before you can use TRAXSCRIPT, you must register it so Windows knows where to find it. You only have to do this once for any given computer. This can be done from within PowerPoint, as well as most other ActiveX-aware applications.

Start PowerPoint and create a new, blank presentation. Make sure that the Control Toolbox window is visible. If not, choose "Control Toolbox" on the View:Toolbars menu.

Click the "More Controls" button at the bottom of the Control Toolbox window. Scroll down to the bottom of the list, and choose "Register Custom Control…".

Click the "More Controls" button.

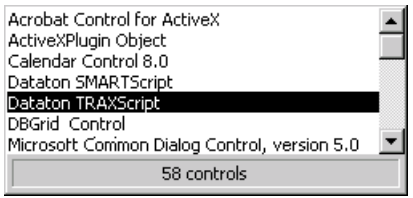
Choose "Register Custom Control…".



Locate and select the newly installed TRAXSCRIPT.DLL file. Once TRAXSCRIPT has been registered in this way, it becomes available to all ActiveX-aware applications on your computer. If you move your TRAXSCRIPT-based application to another computer, you must repeat this registration procedure before you can use your application on that computer.

- **HINT:** If you build your application using Visual Basic, it includes an "Application Setup Wizard", which allows you to create an installer that automates the registration procedure. Other development environments may have similar capabilities.

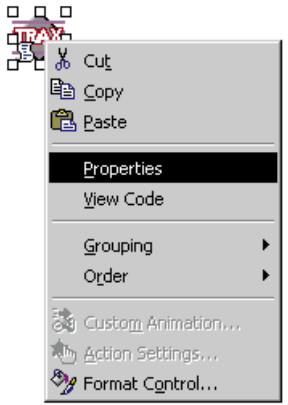
Drawing the TRAXSCRIPT icon



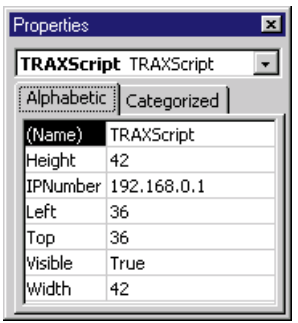
After registering TRAXSCRIPT on your computer, you must click the "More Controls" button in the Control Toolbox again and select "Dataton TRAX-SCRIPT" from the list of available controls. This displays a crosshair cursor, which you use to draw the TRAXSCRIPT icon onto the PowerPoint slide.

The TRAXSCRIPT icon is only shown in design mode. It will not be shown when you run your PowerPoint presentation.

Setting the Properties of TRAXSCRIPT



Click the TRAXSCRIPT icon using the right mouse button and choose Properties from the menu, as shown to the left. In the Properties window that appears, set the Name and IPNumber properties as shown below.



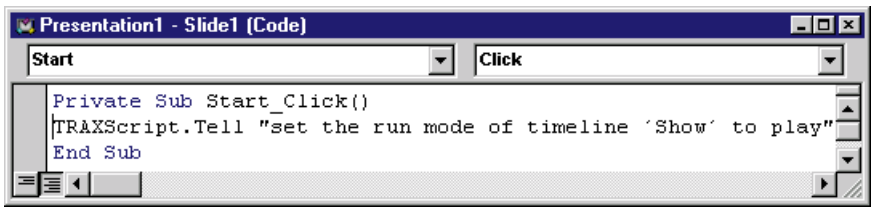
- **NOTE:** As an alternative to specifying the IP Number using the IPNumber property, you may use the Connect command. If you choose the Connect command, then leave IP Number property empty. You must then also use the Disconnect command when closing the connection. The IPNumber property is the recommended method for establishing the connection.

The IPNumber property is the IP Number of your TRAX computer (see "TCP/IP Network Port Scripting" on page 54). TRAX must be on the network, and must have "Network (TCP/IP)" activated in its Scripting Options dialog box (see page 35). If TRAX can't be found, TRAXSCRIPT will display an error message.

Adding a Button

To try out TRAXSCRIPT, draw a Command Button next to the TRAXSCRIPT icon. This is done by clicking the button icon in the Controls Toolbox and dragging diagonally on the slide.

Set the Name and Caption properties of the button to Start. Double-click the button to open its code window. Enter the TRAXSCRIPT statement into the Start_Click handler, as shown below. You refer to TRAXSCRIPT using the name you gave to the TRAXSCRIPT icon (see the Properties window on the previous page).



Scripting dictionary in TRAX, as seen from within Apple's Script Editor.

After typing "TRAXScript" (or whatever name you chose), type a period. This displays a drop-down list of the available TRAXSCRIPT commands.

TRAXSCRIPT Commands

The commands in the ActiveX implementation of TRAXSCRIPT are identical to those in the Director Xtra implementation (see page 105 through page 107). Although you can use the Connect and Disconnect commands (as described on page 104), you may prefer the simpler method of specifying the IPNumber property. Use either Connect/Disconnect or the IPNumber property method – but never both at the same time.

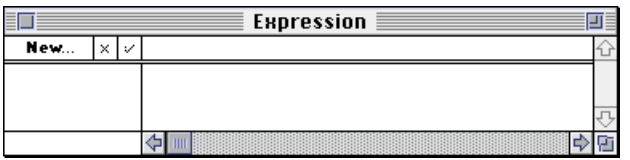
TRAXSCRIPT Errors

The ActiveX implementation of TRAXSCRIPT returns the same error codes as the Xtra implementation (see "Director Xtra Error Codes" on page 109).

6 NAMED EXPRESSIONS

The Expression window allows you to create named expressions.

Open the Expression window by choosing "Expression" on the Window menu.



These expressions are similar to those you can enter in the Condition column of the Task window. They combine numeric constants, device status properties and symbols into mathematical formulas. However, rather than directly using the result of such a formula, as the Task window does, the Expression window simply associates it with a name of your choice. You can then use that name to represent the value of its expression in other places:

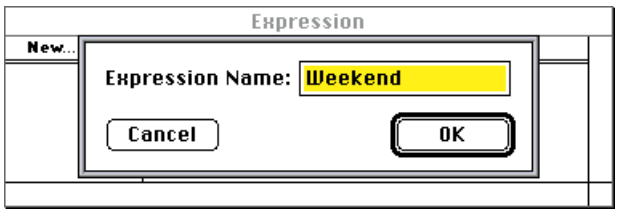
- As part of other expressions in the Expression window.
- In the Condition column of the Task window, either on its own or as part of more complex expressions.
- In the value field of some cues, instead of a numeric constant.

You can think of a named expression as a kind of "variable" because it can be derived from a real or virtual device, which can be set by the user or by cues.

Creating a Named Expression

Follow these steps to create a named expression:

- Choose "Expression" on the Window menu to open the Expression window.
- Click "New…" to create a new expression (Command-N).
- Give it a name and click OK.
- Enter device status references, symbols and numbers into the expression.



Creating a new expression.

You edit expressions in the same way as you edit the contents of the Condition column in the Task window. Device status properties are included by clicking the device in the Device window while editing an expression. This displays a pop-up menu from which you choose the status property for insertion into the expression. See "Linking Conditions to Device Status" and "Editing Conditions" on page 54 in the TRAX 3 handbook for more details.

You can re-arrange the expressions in the Expression window by cutting and pasting them. Select multiple expressions by Shift-clicking or Command-clicking their names. To paste expressions at a specific point in the list, first select the expression currently at that point, then choose Paste. To paste expressions at the end of the list, make sure that no expressions are selected before choosing Paste.

Using Expressions

When you create an expression it is added to the Expression menu. The Expression menu can be sorted alphabetically, if desired, or left in the same order as in the Expression window. You can use the Expression menu to enter the name of an expression into other expressions and into some cues.

Sub-expressions

You can use the result of a named expression in other expressions by inserting the name of the expression. This makes your program more modular, since you then avoid repeating the same expression. See "Common Subexpressions" on page 120 for an example of how this can be used.

Using Expressions in Cues

One of the most powerful uses of named expression is inside cues. You can use the name of an expression instead of a constant value in the following places:

- In the "By Number" or "By Time" field of the Locate cue.
- In the "Scale Factor" field of the Set/Fade cue (see page 30).
- In the "Device Specific Mode" value field of the Trigger cue.

When you enter any of these fields, the Expression menu becomes accessible.

When executing the cue, it will use the value of the named expression at that moment. By basing the named expression on a device status property you could, for example, make the same cue access different songs on a CD.

◆ **CAUTION:** If you use cues with named expressions on a timeline, the resulting status of controlled devices when jumping backwards on the timeline may vary from time to time, depending on the current values of those expressions. This may result in seemingly unpredictable behavior of the entire timeline.

Examples

Following are some examples of how you can use named expressions. The key to understanding the power of named expressions is to understand how you can use device status properties in expressions. Often, the value you want to use in a task condition or a cue is already available inside some device. Sometimes, you may need to adapt the value, for example by adding an offset or changing its scale by multiplying it with a constant.

If the value isn't provided by some real device, you can add a virtual device to keep track of the value. Sometimes you may need several virtual devices to create the desired result (such as in the "Numeric Keypad" example on page 122). Depending on the type of value you need, you can choose any of the following kinds of virtual devices:

- Switch, for on/off (boolean) values.
- Still Store, for integers (0 through 64000).
- Level, for percentages and fractionals (0 through 100, with two decimals).
- Time, for time values (0:00.00 through 23:59:59.99).

◆ **HINT:** It is often better to use a single Still Store instead of several Switch devices, particularly when the functions are mutually exclusive. For example, assume you want to keep track of the user's choice from five buttons on a Panel. It would be better to use a Still Store with the number 1 through 5 to represent the user's choice rather than five Switch devices.

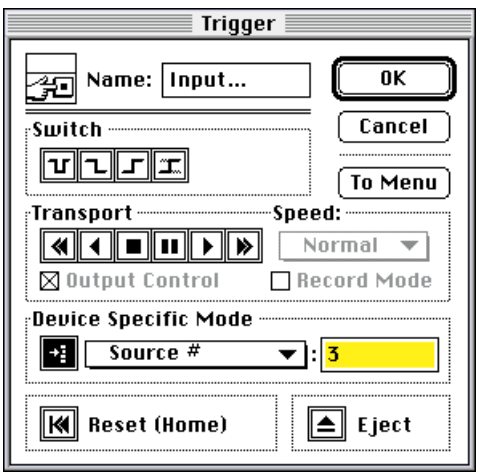
Named Constants

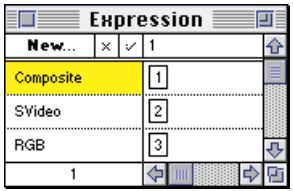
Slot	*Input Module*
1	Composite
2	S-Video
3	RGB

The simplest thing you can do with a named expression is to associate a name with a number. Although this doesn't add anything to the functionality of your program, it can make it easier to read and maintain.

Assume, for example, that you're using a video projector with a number of input slots. These slots can take different kinds of signals depending on the modules that are installed. In a particular installation, you have the projector equipped as shown to the left.

To switch among these inputs, you use a Trigger cue, where you enter the slot number into the value field of the "Source" mode (which is a device-specific mode of a projector, like the Barco Data 8100).

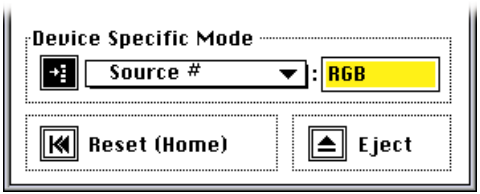




Examples of named constants.

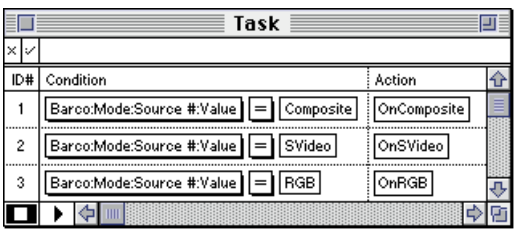
While the above described method works, it's not particularly informative. Furthermore, if you, for some reason, want to swap the input modules around in the projector, you need to track down and change those numbers.

A better way would be to associate names with those numbers, using named expressions, as shown to the left. Now, you can instead use a meaningful name in the value field of the trigger cue:



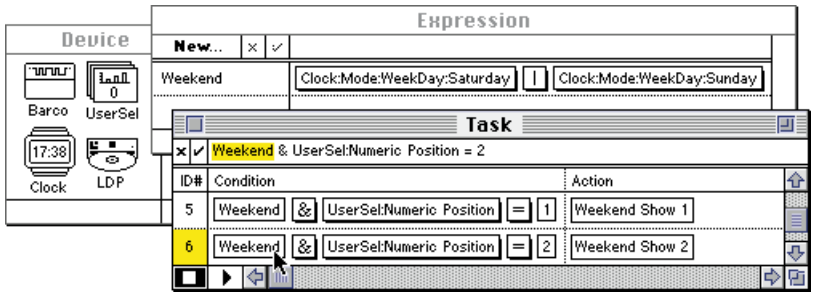
Not only does this make the cue more readable, but it also means that you only have to change the slot number referring to the RGB module at one place in your program.

If you use the current input selection of the video projector in the starting condition of a Task, you can use those names here as well, making those starting conditions easier to read and maintain maintain.



Common Subexpressions

Assume that you want different tasks in the Task list to be performed on weekdays and weekends. You can then define a named expression called "Weekend", which you then use as part of the those starting conditions in the Task list. This makes them more readable and easier to maintain in case you, for some reason, would like to alter the weekend definition, for example to include national holidays.



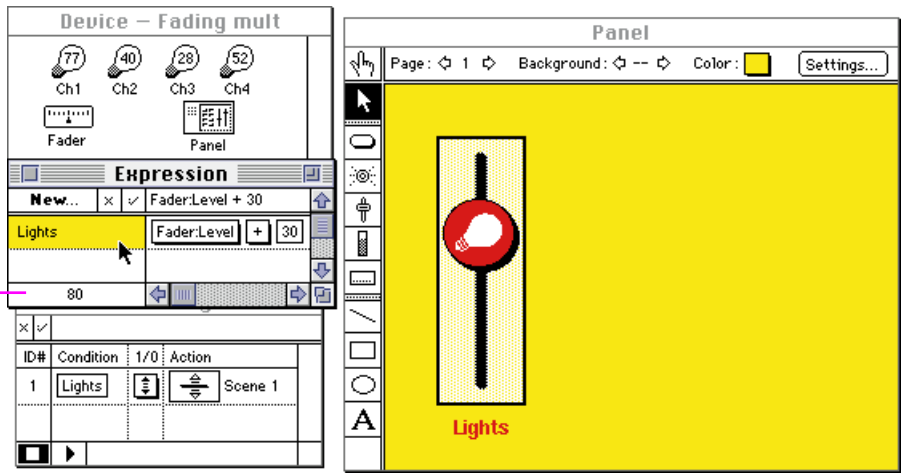
Fading Multiple Devices

By using a named expression in a cue, you can make the behavior of the cue more dynamic. For example, assume you have a cue in the Task window that sets the levels for a number of lighting channels in a ballroom. This preset, called "Evening" is triggered by a button on a panel. However, you may find this arrangement to be too static, since it will recall the same preset every time the button is pressed.

You can easily turn this into a more dynamic behavior by using a slider instead of a button on the panel. This slider, when fully on, could give the same preset as before. However, by moving the slider, you can now gradually reduce the overall lighting in the room, while maintaining the same overall look.

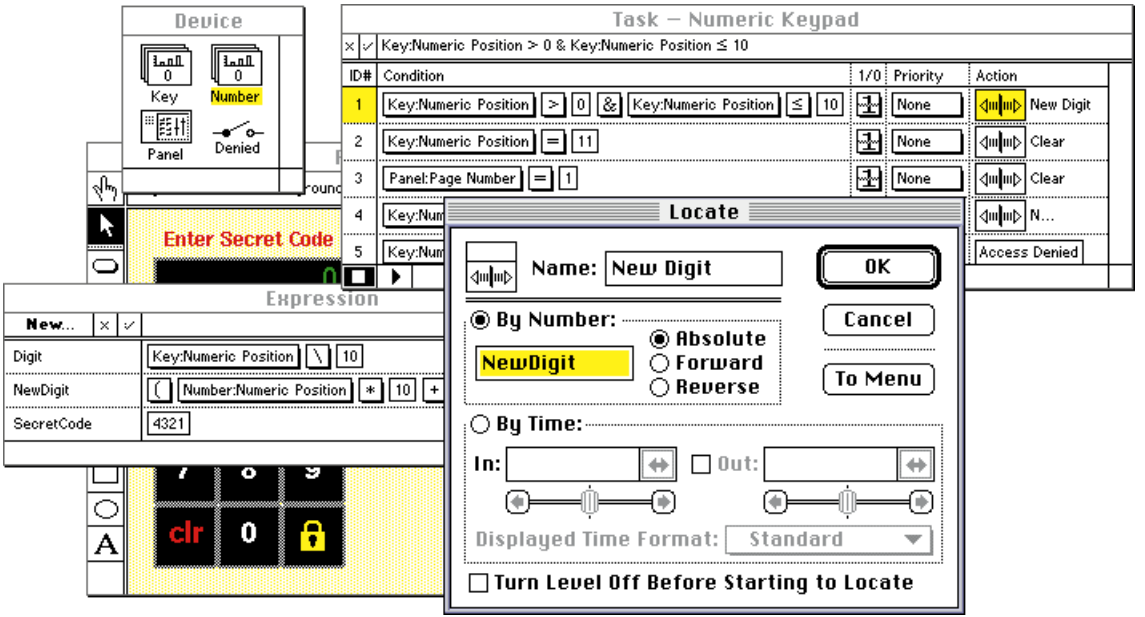
To accomplish this, add a virtual Level device to the Device window. Attach the slider on the panel to this device. Use the value of the virtual device in an expression named "Lights". You may want to add an offset here to make sure that you can't turn the lights off completely by mistake. Use the "Lights" expression in the Scale Factor field of the Set/Fade cue. Finally, change the starting condition of the task containing the Set/Fade cue so it will be triggered whenever the value of the "Lights" expression changes.

The value of the selected expression is displayed here.



Creating a Numeric Keypad

A numeric keypad is often useful to enter, for example, a password or other numeric data. This can be combined with a display item on a panel to provide feedback.



You can use two virtual Still Store devices and some clever expressions to implement the numeric keypad. Each of the keys on the keypad is tied to the device named “Key”. The keys all set the value of “Key” to something greater than zero. This triggers the task named “New Digit”, which uses the expression named “NewDigit” to append the digit to the value in the “Number” device.

The “Digit” expression is somewhat interesting:

```
Key:Numeric Position \ 10
```

It uses the “Remainder of Divided by” symbol to get the numeric value from “Key”. This is necessary because “Key” is set to 10 to represent the value 0, since the value 0 indicates that no key is pressed.

The result of the expression named “Digit” is then used in the “NewDigit” expression to shift in the new digit. This is done by multiplying the old value in “Number” by 10 to shift it one position to the left, and then adding the result of Digit. Finally, the “Remainder of Divided by” symbol (backslash) is used again to clip the result to four digits:

```
( Number:Numeric Position * 10 + Digit ) \ 10000
```

The result of the “NewDigit” expression is used in a Locate cue to update the “Number” device (see picture on page 122).

- **HINT:** The complete programming of this numeric keypad is included on the TRAX diskette in the sample show named “Numeric Keypad”.

The same keypad can be used over and over again for different purposes. You can put its items on a background page, which is then used from all pages that need a numeric keypad. Include the page number of the panel as part of the starting conditions in the task list for the different actions. For example, you could use the same keypad to access an image in a random-access slide projector or a chapter number on a laserdisc.

7 SMARTPAX QC

The latest addition to the Dataton range of control units, SMARTPAX QC (product number 3341) acts as an interpreter between the standard Dataton language spoken by TRAX and the individual languages spoken by each brand and model of device. The device driver software loaded into SMARTPAX QC handles the syntax of each device's language. The smartlink cables, connecting SMARTPAX QC to the devices, handle the physical interface, ensuring that the proper kind of signal reaches the device.

SMARTPAX QC vs. SMARTPAX

SMARTPAX QC is fully upward compatible with its predecessor SMARTPAX. It can use all the same device drivers and smartlink cables.

These are the major differences:

- The display and three buttons on the front panel have been replaced by a control unit ID selector. This streamlines system configuration by allowing TRAX to assign addresses automatically.
- A number of LED indicators on the front panel aid in system set-up and troubleshooting.
- The 24 V AC connector on the rear panel has been replaced by a 12 V DC connector, which is used together with 12V DC ADAPTOR (product number 3334). This provides a higher capacity for powering external devices such as TOUCHLINK.

Front Panel

IN Connector and Indicator

The IN connector links SMARTPAX QC to TRAX or the previous unit in a chain of multiple control units. Use Dataton SYSTEM CABLE to link units together. This cable is available in 0.4, 1, 2 and 5 meter standard lengths. A system cable kit is available for making custom length cables.

When connected to TRAX, the cable between SMARTPAX QC and TRAX may be extended up to 100 meters. When connecting the DATA IN port to a TRANSPAX or AIRLINK RECEIVER, the maximum cable length is 25 meters. This limitation is due to the fact that SMARTPAX QC, in these cases, will supply power to the connected device through the cable.

The IN indicator shows when data is being received by SMARTPAX QC through the IN connector. This is very useful when installing a system as it allows you to follow the signal from one unit to the next, making sure that all units receive data.

OUT Connector and Indicator

The OUT connector allows you to daisy-chain multiple control units. You can connect units in any order. The maximum cable length between units is 100 meters (see above under “IN Connector” for more details).

The data sent to the next unit through the OUT connector is a high-level, digital signal. The signal is optically isolated and re-shaped in each unit to ensure maximum reliability even in very large systems.

The OUT indicator lights up when data is transmitted back to TRAX. This data comes from a device connected to the SMARTPAX QC or from other control units further down the line.

Power can be supplied to the SMARTPAX QC through the OUT connector. This is useful when installing multiple SMARTPAX QC units in a rack as it allows those units to be powered together from one power source (see illustration on page 131).

TAPE Connector and Indicator

The TAPE input accepts a standard Dataton SYNCODE control track, as recorded using a TRANSPAX+ or TIMECODE SMARTLINK (page 132). This allows you to run a presentation from a control track on an audio tape player rather than from a computer.

SMARTPAX QC decodes this tape signal and passes it on to any following units through the OUT connector as a high level, digital signal. As the signal that comes from the tape player is a relatively weak, line-level signal, the cable between the tape player and the first unit in the chain should be kept as short as possible, preferably no longer than two meters.

For final playback without a computer connected, the control signal from the tape player must be connected to the TAPE input of the first unit in the chain – the control signal will not travel backwards through the SYSTEM CABLE. You should never use both the IN and TAPE connectors of a SMARTPAX QC at the same time – always disconnect the computer while playing a control signal through the TAPE input.

You can monitor the quality of the received control signal by watching the TAPE indicator of the unit to which the tape signal is connected. The indicator should remain on all the time while playing the tape. Intermittent flickering indi-

cates a poor quality or weak signal. Try adjusting the playback level to correct the problem. If the tape is old, you may have to replace it with a new tape.

These are some specifications of the TAPE input which might be helpful when troubleshooting an installation:

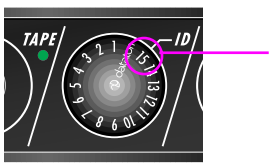
Input sensitivity: 150 mV peak relative ground.

Input impedance: 10 kOhm.

Tape speed tolerance: ±20%.

- **NOTE:** Due to the sparse modulation scheme used by the control signal, it appears as very weak on most VU meters.

Unit ID Selector



Selected unit ID.

The ID selector on the front is used together with the "Choose Port" dialog box in TRAX when setting up a system (see page 15). The ID selector is lit while set to the position specified by the most recently downloaded configuration.

- **NOTE:** The Dataton setting of the ID selector can not be used for normal operation. This setting is reserved for factory configuration and testing purposes. If set to this position, the ID selector will flash rapidly the next time the unit is turned on.

Power Indicators

Three LED indicators show how the SMARTPAX QC is powered:

POWER. Indicates that the SMARTPAX QC has power.

LOCAL. Powered locally through the 12V DC connector on the back.

REMOTE. Powered from the next SMARTPAX QC unit through the OUT connector. Use this only for powering units within close proximity of each other.

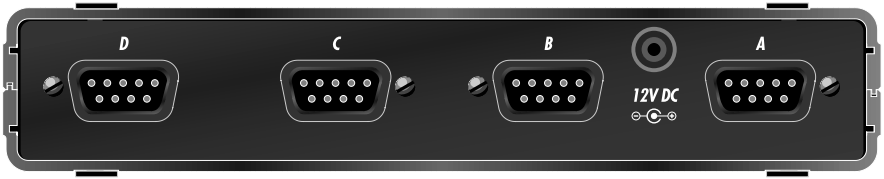
FAILURE Indicator

The FAILURE indicator will be turned on if there's not enough power available. This may happen if you attempt to power too many units from the same 12V DC ADAPTOR (see "12V DC Power Supply" on page 130).

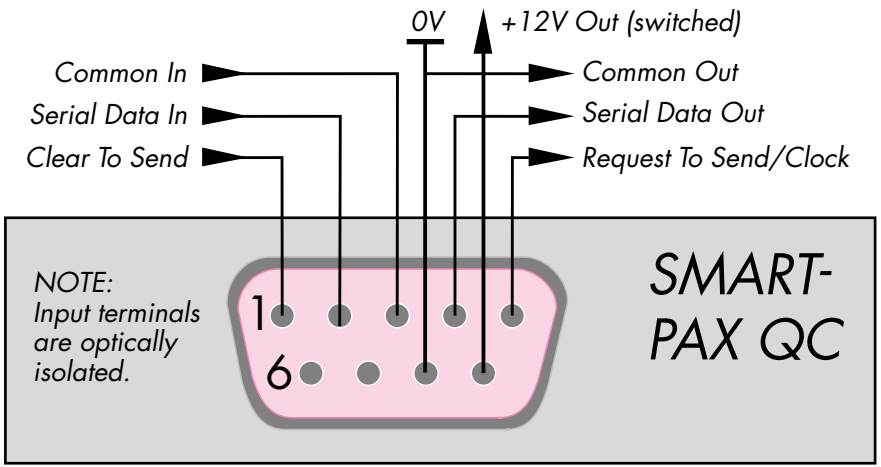
If a serious software error occurs, the FAILURE indicator will flash rapidly. This may happen if the data signal is interrupted, or the power is switched off, while downloading device drivers. Should this happen, proceed as follows to restore SMARTPAX QC to normal operation:

- Connect the SMARTPAX QC directly to the computer, with no other units connected to the bus. Open the Device Support window in TRAX, and select "Device Drivers, Manual Mode" on the pop-up menu. Make sure that at least one driver is checked in the list, and click the Download button. Wait for the download to complete.
- Return the SMARTPAX QC to its proper place in the system. Make sure that the ID selector is set according to its TRAX configuration. Select "Download Everything" on the pop-up menu in the Device Support window, and click the Download button. Once the download is complete, the ID selector will light up to indicate proper configuration.

Rear Panel



The four ports on the back of the unit connect to the devices being controlled, using the appropriate smartlink cable. Each of the four ports is designated by a letter (A through D) corresponding to the Port pop-up menu in the "Choose Port" dialog box in TRAX (see page 15).



Wiring of one SMARTPAX QC port.

- **NOTE:** Power can not be supplied to SMARTPAX QC through its port connectors.

12V DC Power Supply

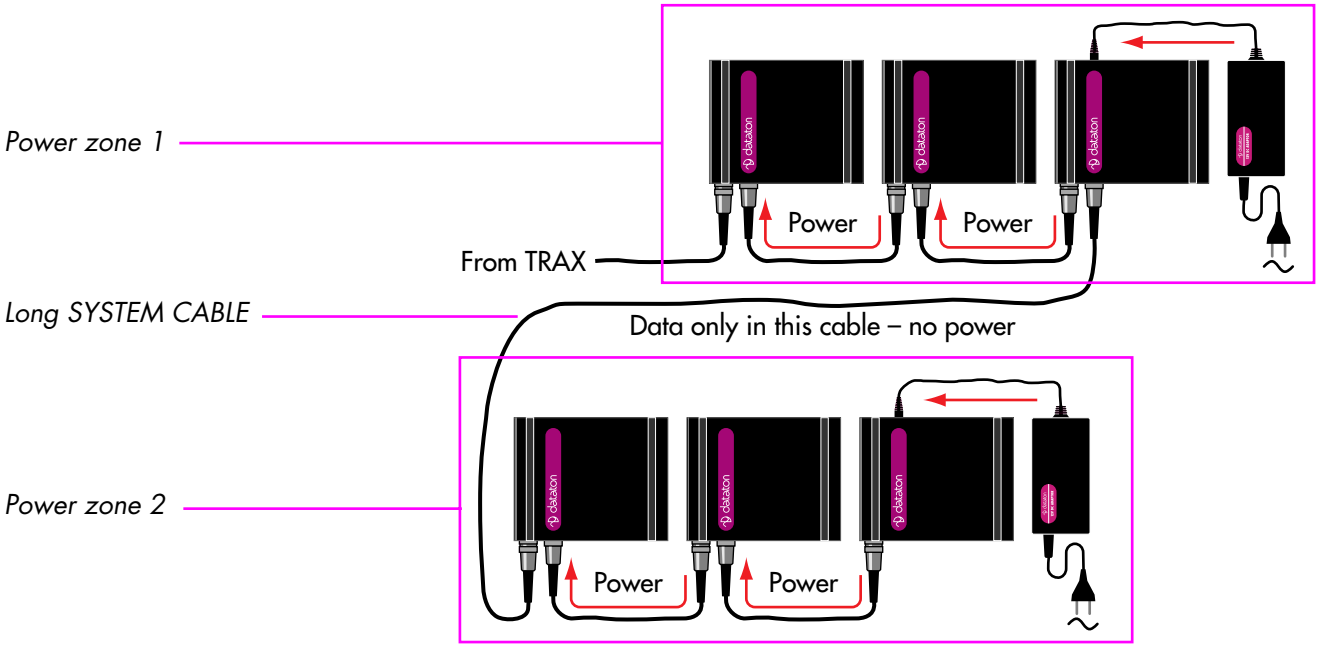
12V DC ADAPTOR power supply (product number 3334).

Connect a 12V DC ADAPTOR (product number 3334) to the 12V DC connector on the back of SMARTPAX QC.

When installing a group of SMARTPAX QC units within close proximity, such as in a rack, the last unit can supply power to other units in the same local group through the SYSTEM CABLE. Such a local group is called a power zone.

Power travels backwards through the SYSTEM CABLE, from the IN connector of one SMARTPAX QC to the OUT connector of the previous. Thus, you must connect the 12V DC ADAPTOR to the last SMARTPAX QC in each power zone, as shown in the illustration on page 131.

When powering multiple SMARTPAX QC units in this way, be careful not to overload the power supply. The FAILURE indicator on the front panel of SMARTPAX QC will turn on if the power supply is insufficient. Please refer to the manual for the power supply for more details.



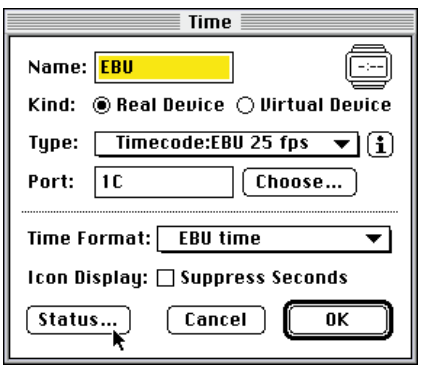
▼ **CAUTION:** Do not attempt to power SMARTPAX QC units that are far apart through the SYSTEM CABLE. Doing so may result in unreliable operation or damage to connected equipment.

The example above shows two groups of three SMARTPAX QC units each. A longer SYSTEM CABLE links the two groups together. Power is supplied within each group from its 12V DC ADAPTOR. A SMARTPAX QC unit that is powered through its 12V DC connector will not draw power from the next SMARTPAX QC unit. Furthermore, this breaks up the power connection between the units, thereby avoiding ground loops.

8 TIMECODE SMARTLINK

TIMECODE SMARTLINK provides capabilities to generate and read longitudinal SMPTE and EBU timecode, as well as to record a control signal (cue track) for later playback without the computer.

Reading Timecode



TIMECODE SMARTLINK can read standard SMPTE and EBU LTC timecode signals in order to synchronize timelines in TRAX.

- Connect TIMECODE SMARTLINK to SMARTPAX or SMARTPAX QC.
- Connect the white RCA phono connector to the timecode source.
- Add a Time device to the Device window in TRAX.
- Double-click the time device and choose the appropriate timecode format on the Type pop-up menu.
- Specify the port to which the TIMECODE SMARTLINK is connected.
- Click the Status button to open the status window.
- Open the Device Support window and click Download to configure the SMARTPAX.
- Start the timecode source and read the timecode in the status window.

Synchronizing to Timecode

Use a Control cue to synchronize a timeline to the timecode source, as described on page 223 in the TRAX 3 handbook.

When using TRAX in its "Backward Compatible" mode, you may not use more than one TIMECODE SMARTLINK for reading timecode. When TRAX is set to its "Interactive" mode, you can have any number of timecode sources active at the same time, if desired. See "System Modes" on page 16 in the TRAX 3 handbook for further details.

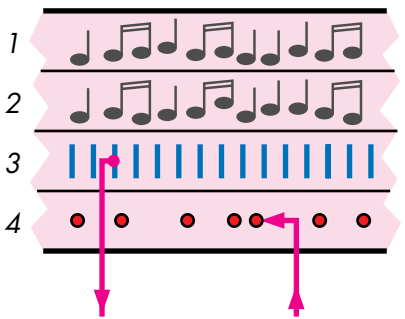
Control Signal Recording

The show can be recorded onto a separate control signal track on tape. This allows you to run the show without using the computer, simply by feeding the control signal into the first unit in the control unit chain. This can be any kind of Dataton control unit equipped with a TAPE or PLAY input on the front panel.

▼ **IMPORTANT:** To record a control signal, TRAX must be in its backward compatible mode. See "System Modes" on page 16 of the TRAX 3 handbook for more details.

Follow these steps to record a control signal using the TIMECODE SMARTLINK:

- Connect the red RCA phono connector to LINE IN of the track to be used for the control signal.
- Set that track in Record mode, and set the recording level as you would for recording a line level signal.
- Make sure that the white RCA phono connector is still connected to LINE OUT of the timecode track for synchronization purposes.
- Start the tape in record mode.
- Run the timeline as usual – the control signal is recorded as you go.



Timecode being read back.

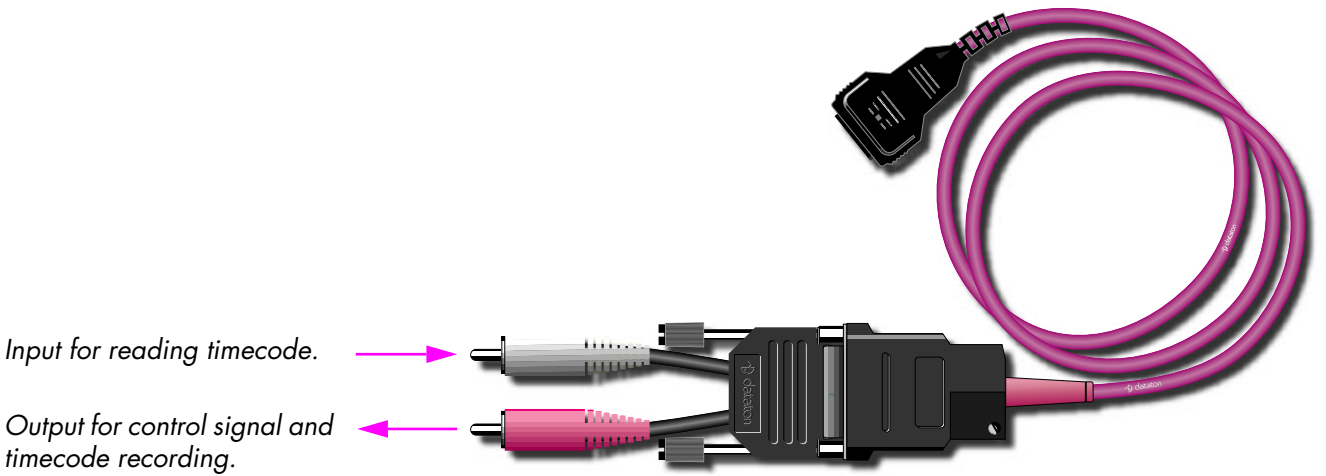
Control signal being recorded.

Refer to “Recording a Control Track” on page 225 of the TRAX 3 handbook for more details on recording the control signal.

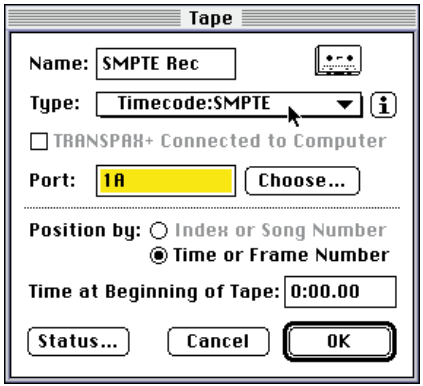
- **NOTE:** The level indicated by the VU meter may fluctuate while recording the control signal. This is due to the sparse modulation scheme used by the control signal. Do not compensate for it by increasing the recording level.

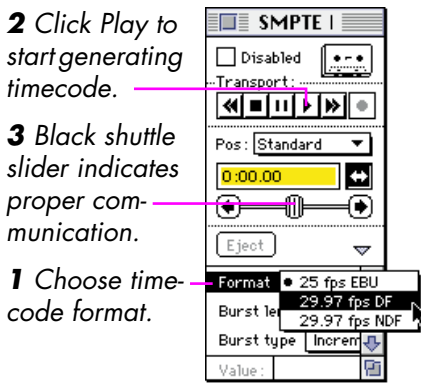
Control Signal Playback

The TIMECODE SMARTLINK is not required to play back the control signal. Simply connect the LINE OUT of the control track to the TAPE or PLAY input on the front of the first control unit in the chain, quit TRAX, and start the tape.



Generating Timecode





As the TIMECODE SMARTLINK reads standard SMPTE and EBU LTC timecode, you can use virtually any timecode generator to record the timecode. If you don't have a separate timecode generator, you can use TIMECODE SMARTLINK to record the timecode track. Follow these steps:

- Create a new show in TRAX.
- Add a single Tape device and open its configuration dialog box.
- Configure it as shown to the left. Set the Port field to 10 if using SMARTPAX or 1A if using SMARTPAX QC.
- Connect TIMECODE SMARTLINK to the leftmost port on a SMARTPAX.
- Connect the red RCA phono connector to LINE IN of the timecode track.
- Connect the SMARTPAX directly to the computer.
- Set the leftmost port to address 10 and the others to 00 (SMARTPAX), or set the ID selector at 1 (SMARTPAX QC).
- Open the Device Support window and click Download.
- Open the status window for the single Tape device, choose the desired timecode format using the Format pop-up menu.
- Click the Play button in the status window, and start the tape in record mode. The timecode is now being generated. Adjust the recording level on the recorder, if required.

Technical Specifications

Here are some specifications of the TIMECODE SMARTLINK which might be helpful when troubleshooting an installation.

Output level: 1V peak (line level).

Nominal input level: 1V peak (line level).

Input sensitivity: 100 mV peak.

Input impedance: 22 kOhm.

Tape speed tolerance: ±20%.

Timecode formats: EBU 25 fps, SMPTE 29.97 fps drop-frame, SMPTE 29.97 fps non-drop frame, SMPTE 30 fps (30 fps playback only).

Control track format: Dataton SYNCODE, compatible with all Dataton control units equipped with a TAPE or PLAY input.

Power supply: 12V DC, supplied by SMARTPAX.

Index

A

AAUI 57
ActiveX 110
address, specifying manually 16
AppleScript 35, 37
    activating 39
    dictionary 40
    editing 40
    specific capabilities of 44
    using via network 43
as, scripting keyword 71
AskTRAX, Lingo command 107
AUI 57

B

background picture, in Device window 13
BNC 57
built-in devices 5
button 26
button item, news 26

C

CD-ROM, audio playback 18
Changes, condition variant 8
client/server 34
coaxial network cable 57
condition variant 8
ConnectTRAX, Lingo command 104
control signal 133
cue
    expression in 116, 119
    locate 29
    set/fade 30
    trigger 32
    using expression in 26, 27, 29
cue track 133

D

database scripting 101
device
    configuration dialog box 14, 15
    standby 17
    status properties 72
device-specific modes 72
dictionary 40
Director (*see also* Lingo) 103
DisconnectTRAX, Lingo command 104
display item, news 28
dithering, of pictures 28

E
EBU timecode 132, 135
error, in script 53
Excel 110
expression 6, 114
    creating 115
    examples of 117
    in cues 116, 119
    in task condition 119
    using 116
    window 114
expression menu 10

F
fading multiple devices 120

H
hard disk
    audio playback 21
    upgrading the driver on 19

I
IP number 55, 59, 61
IPNumber property 112

L
line endings, in scripts 50
Lingo 103
    AskTRAX 107
    ConnectTRAX 104
    DisconnectTRAX 104
    error codes 109
    RetrieveTRAX 107
    TellTRAX 105

M
MacOS 8 12
Macromedia Director (*see also* Lingo) 103
monitor
    as panel device 23
    size of 23
    using as panel 25
    using multiple 25

N
named expressions 6, 114
network
    configuration 64
    point to point 64
    wiring 56
none priority 7
numeric keypad 122

O
objects
for scripting 65
hierarchy of 67
identifying 66
open transport, configuring 54
override priority 7

P
password protection 24
picture, color fidelity 28
pictures, on monitor panel 25
port assignment 14
PowerPC 12
PowerPoint 110
priority
none 7
override 7
properties, of scriptable objects 70

R
replies to script commands 86
RetrieveTRAX, Lingo command 107
RJ-45 57

S
scale factor
in set/fade cue 31
using with named expression 32
scripting 5, 33
AppleScript 37
application object 74
as keyword 71
controlling devices 91
count command 81
cue object 77
cues 96
database 101
device object 74
device status properties 72
device-specific modes 72
dictionary 40
error codes 88, 109
errors 53, 63, 87
get command 79, 93
getting device status 93
identifying objects 66
language overview 65
line endings 50, 62
Macromedia Director 103
make command 82, 96
network wiring 56
object hierarchy 67
objects 65
open command 84, 97
options dialog box 35
other applications 46
perform command 81, 94

scripting…
performing cues 94
properties 70
queries 51, 63, 86, 106
quit command 85
replies 86
save command 85, 97
serial port 36, 47
set command 79, 91
specifying timelines 69
task list, controlling 95
task object 76
TCP/IP 36, 54
testing and debugging 42
timeline object 77, 95
timelines 95, 96
transaction ID 51, 63
type of result 71
Windows 95 58
Windows NT 60
word processor 98
serial port
activating for scripting 48
data rate 48
handshaking 50
wiring 49
serial port scripting 36, 47
SMARTPAX QC 5, 14, 124
failure indicator 128
front panel 125
IN connector 125
OUT connector 125
port 129
power indicators 127
power supply 129
rear panel 129
TAPE connector 126
unit ID selector 127
vs. SMARTPAX 124
wiring 129
SMPTE timecode 132, 135
subnet mask 55, 59
system requirements 12, 19, 38

T
tasks 7
TCP/IP 36, 54, 58
TellTRAX, Lingo command 105
Telnet 61
timecode
    generating 135
    reading 132
timeline 8
Toolbook 110
touch overlay 24
transaction ID 51
TRAXSCRIPT 110
    installing 110
    properties 112
    registering 110
TRAXSCRIPT.DLL file 110

V
video, displaying on monitor 27
Visual Basic 110
volume control
    hard disk audio 22
    of CD-ROM 19

W
Windows 95 58
Windows NT 60

X
Xtra, for Macromedia Director 103