ToolServer Reference

030-3037-A
Developer Technical Publications

Contents

Figures, Tables, and Listings

How to Use This Manual

This manual explains how you install and launch ToolServer and how you use it from different environments to run Macintosh Programmer's Workshop (MPW) tools and to execute MPW scripts.

ToolServer is a complete tool and script execution environment extracted from the MPW Shell that you can use to execute time-consuming, noninteractive tools or scripts in the background or on a remote machine. Thus ToolServer can run most of the tools and scripts in the MPW tool suite and most tools and scripts which are written for MPW. The major exceptions are editor and Projector commands.

ToolServer is primarily designed to be run using Apple events. As such, it can be run from MPW 3.3 or later using the built-in MPW `RShell` command and from third-party applications that include support for ToolServer. You can also use ToolServer as a foreground application; you specify the scripts to be executed and obtain the status of executing scripts by choosing the appropriate File menu items.

This manual is divided into four chapters:

- Chapter 1, "Making ToolServer Work for You," describes the ways in which you can use ToolServer and discusses memory requirements, performance, and compatibility issues.
- Chapter 2, "Using ToolServer," explains how you install, launch, and quit ToolServer. It also explains how to use ToolServer as a foreground application, how scripts execute under ToolServer, and ToolServer's use of startup, input, output, and diagnostic files.
- Chapter 3, "The RShell Command," explains the syntax of the `RShell` command and how you use it from the MPW Shell (version 3.3 or later) to have ToolServer execute scripts in the background or on a remote machine.
- Chapter 4, "Using Apple Events to Communicate With ToolServer" describes the Apple events supported by ToolServer. By using these events and the routines of the Apple Event Manager, an application can communicate with ToolServer and use it as an execution environment that extends the application's functionality.

Note

Because of ToolServer's dependence on Apple events, ToolServer runs only on System 7 or later. ◆

Related Documentation

For detailed information on the syntax and use of MPW commands, tools, and scripts, you need to consult the *MPW Command Reference*. For information on writing scripts, you need to consult Volume 1 of *MPW: Macintosh Programmer's Worskhop Development Environment* or any third-party book describing the MPW Shell.

Syntax Conventions

The following syntax conventions are used to describe MPW and ToolServer commands in this manual.

`literal`	Courier text indicates a word that must appear exactly as shown. Special symbols (∂, •, §, &, and so on) must also be entered exactly as shown.
italics	Italics indicate a parameter that you must replace with anything that matches the parameter's definition.
[ ]	Brackets indicate that the enclosed item is optional.
...	Ellipses (...) indicate that the preceding item can be repeated one or more times.
|	A vertical bar (|) indicates an either/or choice.

Terminology

Although ToolServer is independent of MPW, it does share with MPW a common set of commands, scripts, and tools. This manual refers to the commands, scripts, and tools that ToolServer can execute as MPW commands, MPW scripts, and MPW tools because most readers are familiar with these from working with MPW and because the behavior of these commands, tools, and scripts is basically the same in both environments. (Any differences in behavior are documented in this manual.) However, this does not imply that ToolServer's ability to run these commands, scripts, and tools depends upon MPW running concurrently with ToolServer.

The term **client application** refers to an application that sends an Apple event to request a service. The term **server application** refers to the application that performs that service. For example, when you use the `RShell` command from the MPW Shell to have ToolServer execute a script, the MPW Shell is the client application and ToolServer is the server application.

C H A P T E R 1

Making ToolServer Work for You

This chapter describes the circumstances under which you might use ToolServer, explains ToolServer's use of memory and overall effect on performance, and reviews compatibility requirements.

Why Use ToolServer?

How you use ToolServer depends on your needs. The possibilities include the following:

- You can run ToolServer as a foreground application that you launch from the Finder. This allows you to execute scripts in a smaller partition than that required by the MPW Shell.
- You can run ToolServer in the background while you continue to work with MPW in the foreground.
- You can run ToolServer on a remote machine and use Apple events to communicate with ToolServer and to specify which scripts are to be executed. Using this process, you can off-load time consuming build operations to another machine.

  Version 3.3 or later of the MPW Shell includes an `RShell` command that uses Apple events to communicate with ToolServer. Any other application or development environment can also use Apple events to communicate with ToolServer.
- You can use ToolServer to extend the functionality of an application or development environment.

  The application or development environment that needs to make use of an MPW tool or script uses Apple events to have ToolServer execute that tool or script in the background or on a remote machine. The existence of ToolServer would be invisible to the user who would specify his needs by means of a menu command or dialog box.

Memory Requirements, Performance, and Compatibility

The following sections describe ToolServer's memory requirements, how the execution of scripts in the background or on a remote machine affects performance, and ToolServer's dependence on system software, MPW, and other client applications.

ToolServer Partition Size

The size of ToolServer's partition depends on the types of tools run. In general, the partition required by ToolServer should be 500K smaller than that required by the MPW Shell. The recommended memory requirement for ToolServer is 1536K. The minimum requirement is 512K.

If you run all your complex operations with ToolServer and use MPW primarily for editing, you can launch MPW in a smaller partition.

In the worst case, the combined memory requirements of running ToolServer in the background and MPW in the foreground might be up to twice that of the MPW Shell.

Performance

A script run in the background should not require significantly more central processing unit (CPU) time than a script run in the foreground. In fact, scripts run by ToolServer might execute faster than they would under MPW, because much of the user interface overhead has been removed from ToolServer.

Elapsed time varies depending on the amount of foreground activity and the cost of context switches. For example, if you run CPU-intensive operations from both the ToolServer and another application on the same CPU, the elapsed time could increase by a factor of 2 or more.

If you run ToolServer on a remote machine, performance is affected by network overhead—that is, whether the files required by ToolServer are available locally or are read remotely.

Compatibility

Because of ToolServer's dependence on Apple events, ToolServer runs only on System 7 or later. When ToolServer is run under A/UX, it cannot be accessed remotely, that is, by the `RShell` command.

MPW version 3.3 or later is required to control ToolServer using Apple events. ToolServer supports all the routines and interfaces available to tools under the MPW Shell, including `facess` and `getenv`. Tools should run without change in either the MPW or ToolServer environment. ToolServer and the MPW Shell do differ in the way they handle input, output, and standard error. These differences are described in the section "Input, Output, and Diagnostic Files" in Chapter 2.

Any client application that knows how to send the Apple events supported by ToolServer and how to handle ToolServer's replies to these events can communicate with ToolServer in the background or on a remote machine. Version 3.3 of the MPW Shell is such a client; you can use the `RShell` command to specify scripts to be run in the background or on a remote machine.

ToolServer has the signature `'MPSX'`. Files produced by ToolServer are of type `'TEXT'` and creator `'MPS '` and can be viewed or edited with the MPW editor or other editors that accept text files. ToolServer also supports the `RShell` command. This means that commands can be sent from ToolServer to other ToolServers (possibly on remote machines), to the foreground MPW Shell (version 3.3 or later), or to itself.

Note
ToolServer takes its input from files. A window represents a file that may contain unsaved editing changes. In order to make ToolServer aware of such changes, you must save them to a file. ◆

C H A P T E R 2

Using ToolServer

This chapter explains how you install, launch, and quit ToolServer. It also explains script execution under ToolServer and ToolServer's use of startup, input, output, and diagnostic files.

Installing ToolServer

If you have purchased or plan to use ToolServer with the MPW Shell, please follow the instructions in the file `READMEFIRST` to install ToolServer. If you have bought or plan to use ToolServer with a third-party application or development environment, please read the documentation provided with that product for installation instructions.

If you plan to use ToolServer to run scripts on a remote machine, you must install and launch ToolServer on that machine. This cannot be done remotely. You must also enable communication between the local and the remote machine. For additional information see the section "Communicating With ToolServer on a Remote Machine" in Chapter 3.

Launching ToolServer

You can launch ToolServer from the Finder or from the MPW Shell. The next two sections describe each of these procedures and also explain the organization and use of ToolServer's startup scripts.

Note
If you have purchased or plan to use ToolServer with a third-party application or development environment, please consult the documentation provided with your product for instructions on how to launch ToolServer. ◆

Launching ToolServer From the Finder

To launch ToolServer from the Finder, double-click the ToolServer application.

You can also launch ToolServer by double-clicking on the ToolServer script to be run. To identify a script as a ToolServer document, use the `SetFile` command to change the creator of the script to `'MPSX'`, the signature of the ToolServer.

Launching ToolServer From the MPW Shell

The syntax of the command used to launch ToolServer from the MPW Shell is:

[*pathname*]`ToolServer [` *script*`...]`

After executing the startup script or scripts, ToolServer executes each *script* specified on the ToolServer command line.

If ToolServer resides in the current directory, or if you include the directory containing ToolServer in the `{Commands}` variable, you can omit the *pathname* specification.

After ToolServer has finished executing the specified scripts, it remains active until you quit the application. If you want to execute additional scripts, you can bring ToolServer to the foreground and specify these by choosing the Execute Script item from the File menu.

You can launch ToolServer, execute scripts, and quit Toolserver by using the `RShell` command. For additional information see Chapter 3, "The RShell Command."

Startup Scripts

Once launched, ToolServer first executes the `StartupTS` script. The `StartupTS` script then executes the `UserStartupTS` script and then any script whose name begins with `UserStartupTS•`.

The `StartupTS` script defines variables used by ToolServer and user scripts. For additional information about the variables defined in the `StartupTS` script, see Appendix B, "ToolServer Files and Variables." You can write additional scripts and have these executed as part of the launch process by naming these scripts `UsersStartupTS•`*name*.

If the execution of the `StartupTS` script results in text written to standard output or standard error, ToolServer creates the files `StartupTS.out` and `StartupTS.err` files in the same directory as the user's `StartupTS` script and writes output and diagnostic information to these files.

Moving ToolServer or Its Startup Scripts Out of the ToolServer Folder

If you are using ToolServer with the MPW Shell, it is assumed that ToolServer is in the ToolServer folder and that its startup and quit scripts are either in the same folder as ToolServer or in the Preferences folder. It is recommended that you do not move ToolServer or its startup and quit scripts out of the ToolServer folder. If quick access to ToolServer is a problem, you can create an alias for it in the desired location.

If you must place the `UserStartupTS` and `QuitTS` scripts in a folder different from the ToolServer or the Preferences folder, you will need to change the value of the `{PrefsFolder}` variable in the `StartupTS` script. For example, if you place ToolServer's `UserStartupTS` script in a folder called `MyStartups` on the boot directory, you can include the following command in the `StartupTS` script to enable ToolServer to find its user startup scripts.

```
Set PrefsFolder {Boot}MyStartups:
```

The `StartupTS` script also defines the `{MPW}` variable to be the same as the directory containing ToolServer. If you move ToolServer to another directory, you need to set the `{MPW}` variable to the location of the ToolServer folder. For example, if you move

ToolServer to the desktop and leave the ToolServer or MPW folder on the boot volume, you would use the following command to redefine ToolServer's directory:

```
Set MPW {Boot}MPW:
```

Combining MPW and ToolServer Startup Scripts

If you find it awkward to maintain separate startup scripts for MPW and ToolServer, you might want to isolate common aspects of the startup process into one script, and to change MPW's and ToolServer's startup process to call this common script. The best way to do this would be to write a `UserStartupTS•` script which would call the scripts common to ToolServer and MPW; a corresponding MPW script, or an alias to the ToolServer script would also be needed. In the same way, you could also write a ToolServer startup script that would run any MPW startup script. You can include some conditional startup code in this combined file to examine the ToolServer variable `{BackgroundShell}` and to determine which application is running: 1 is ToolServer, 0 is MPW or any other client application.

Listing 2-1 shows a sample ToolServer startup script. This script first executes all MPW startup scripts in the ToolServer folder and then executes a script called CommonStartup, if this script exists in the ToolServer folder. The script also demonstrates the use of the `{BackgroundShell}` ToolServer variable: if the script is run from MPW, it executes all ToolServer startup scripts (except itself) instead of running all MPW startup scripts.

Listing 2-1 Sample startup script

```
Set myName "{command}"
if {BackgroundShell}
      set scriptHead "UserStartup•"
else
      set scriptHead "UserStartupTS•"
end

set oldExit "{exit}"
set exit 0
for file in "{MPW}{scriptHead}"≈
      if "{file}" !~ /≈{myName}/    # Prevent recursive execution of this
         execute "{file}"           # script
      end
end
`exists "{MPW}"CommonStartup`       # Execute CommonStartup if it exists

set exit "{oldExit}"
unset myName scriptHead file oldExit# Remove local variable definitions
```

Quitting ToolServer

Once ToolServer is launched, it executes scripts or waits for additional input until one of the following happens:

- You bring ToolServer to the foreground and choose Quit from the File menu.
- You run a script containing the built-in command `Quit.`
- The ToolServer receives a `Quit` event. For additional information see Chapter 4, “Using Apple Events to Communicate With ToolServer.”
- You specify the `-q` (quit) option to the `RShell` command. For additional information see Chapter 3, “The RShell Command.”

ToolServer executes the `QuitTS` script when asked to quit. ToolServer will also execute all `QuitTS•≈` scripts that exist in ToolServer’s directory (or any other directory if the `QuitTS` script is modified).

Note
If you are using ToolServer with the MPW Shell, it is assumed that the `QuitTS` script is either in the same directory as ToolServer or in the Preferences folder. If you move it to a folder different from these two, you must change the value of the `{PrefsFolder}` variable in the `StartupTS` script to the new location of the `QuitTS` script. ◆

Using ToolServer in the Foreground

It is a matter of convenience whether you use ToolServer in the foreground or in the background. Using MPW’s `RShell` command, described in Chapter 3, or the Apple events, described in Chapter 4, you can obtain status information about an executing script, have ToolServer execute additional scripts, or quit ToolServer—all without having to bring ToolServer to the foreground.

Whether you launch ToolServer from the MPW Shell or from the Finder, you can bring ToolServer to the foreground and use the menu items shown in Table 2-1 to execute scripts and monitor status. ToolServer supports the Apple, File, and Edit menus.

Table 2-1 ToolServer menu selections

Menu	Item	Effect
File	Open Status	Opens a status window that displays the name of the currently executing command or tool. When ToolServer is idle, the name "ToolServer" is displayed in the status window.
	Execute Script... (Command-E)	Displays a standard file dialog box from which you select a script to execute.
	Close Status	Closes the status window.
	Quit	Terminates ToolServer.
Apple	About ToolServer...	Displays version and copyright information.
Edit	(all)	Become active when you open a desk accessory or dialog box.

Script Execution

Most traditional Macintosh applications provide a document and one or more ways of opening that document. A ToolServer document is a script, a file of type `'TEXT'` that contains one or more MPW command lines. Unlike the MPW Shell, which creates a window in which to display and edit that script, when ToolServer opens a script, it simply executes the specified script.

When using ToolServer as a foreground application or with MPW, you can tell ToolServer to execute one or more documents in the following ways:

- Double-click one or more ToolServer documents in the Finder. (A ToolServer document is a file of type `'TEXT'` and creator `'MPSX'`.)
- Drop the document icon or icons onto the ToolServer icon in the Finder.
- Specify the document name as a parameter when launching ToolServer from the MPW Shell.
- Choose Execute Script from ToolServer's File menu.

Whichever of these methods you use to open a document, ToolServer sets its working directory to be the directory containing the specified file. The script contained in the file is then run as if it were executed by the MPW Shell.

Note
Third-party application or development environments may provide additional ways of executing scripts. Please consult the documentation provided for additional information. ◆

Aborting Script Execution

You can abort a script running under ToolServer by bringing ToolServer to the foreground and entering Command-. (Command-period) from the keyboard. This command aborts the currently executing script. If scripts are queued waiting to run, ToolServer executes the next script in the queue.

You can also abort scripts by using the `RShell` command, described in Chapter 3 or the `'abrt'` Apple event described in Chapter 4.

Limitations on Scripts That Run Under ToolServer

Scripts can invoke any MPW tool and can contain the MPW built-in commands listed in Table 2-2. Scripts can also include variables, command aliases, substitution characters, quotes, wildcard operators, and redirection commands. Projector and editor commands are not supported; attempts to execute such commands cause ToolServer to set the `Status` variable to 1. ToolServer sets the variable `BackgroundShell` to 1 so that scripts can determine if they are running under the ToolServer (1) or under the MPW Shell (0).

Table 2-2 Valid MPW built-in commands for scripts

Alert	Directory	Exit	Newer	Shift
Alias	Duplicate	Export	NewFolder	ShutDown
Beep	Echo	Files	Parameters	Unalias
Begin...End	Eject	Flush	Quit	Unexport
Break	Else	For...	Quote	Unmount
Catenate	Equal	Help	Rename	Unset
Confirm	Erase	If...	Request	Version
Continue	Evaluate	Loop...End	RShell*	Volumes
Date	Execute	Move	Set	Which
Delete	Exist	Mount	SetFile	

* The `RShell` command is part of ToolServer's command set, but is only included in version 3.3 or later of the MPW Shell.

It is also a good idea to make your scripts as generic as possible so that they can run in different contexts without requiring changes. In general, observe the following points in writing scripts:

- Do not use command aliases in a script unless they are also defined in that script.
- A script should not depend on variables being defined globally except for the pre-defined MPW variables.
- Place source files in their own folder; don't put them in the MPW folder.
- Don't depend on volumes being available. A script might need to include code that determines which context it is running in and mounts the required volumes if it is not running in the same location where the files are placed.
- If you plan to run a script on a remote machine, avoid commands that put up dialogs. If the command puts up dialogs, and there are options available for disabling these dialogs, use these options.
- Make sure that your script handles error returns; the script can ignore an error, but it should not abort as a result of an error, except by design.

Input, Output, and Diagnostic Files

ToolServer and the MPW Shell differ in the way they handle input, output, and standard error. This section describes these differences.

The MPW Shell provides tools and scripts with three preopened files: standard input, standard output, and standard error. The MPW command language allows you to override these defaults to redirect input/output (I/O). MPW also provides a set of pseudodevices; these include `Dev:Console`, `Dev:Null`, `Dev:StdIn`, `Dev:StdOut`, and `Dev:StdErr`.

When you run a tool or script in MPW without specifying I/O redirection, standard input, standard output, and standard error are initially directed to `Dev:Console`. The MPW Shell installs a device handler for `Dev:Console` that captures reads and writes, and directs them to the frontmost window. This is generally the MPW Shell's Worksheet window. ToolServer does not have a worksheet or any window, and thus cannot provide a console device in the traditional way.

If you run a tool or script under ToolServer that does not redirect output to a file, output is sent to either of two files (pseudodevices) introduced by ToolServer to handle this situation: `Dev:Output` and `Dev:Error`. Standard output is sent to `Dev:Output`, and standard error is sent to `Dev:Error`. What ToolServer does next depends on the method used to execute scripts:

- If the script is executed by using one of the methods described in “Script Execution” on page 2-6, ToolServer creates files to hold output and error text. The name and location of these files depend on the value of the variables `BackgroundOut` and `BackgroundErr`. These variables are not set when ToolServer starts; you can set them in the `UserStartup•` script.

  ToolServer evaluates the `BackgroundOut` and `BackgroundErr` variables and creates the necessary files when the tool or script first writes to `Dev:Output` or `Dev:Error`. Table 2-3 shows the effect of various settings for these variables on the files created by ToolServer.

  If `BackgroundOut` and `BackgroundErr` are not set, the output files are created in the directory containing the script and are the name of the script with `.out` and `.err` appended respectively. For example, if the script `Test` in the folder `HD:TestScripts:` is executed, the output and error files would be `Test.out` and `Test.err` in the folder `HD:TestScripts:`.

  Files produced by ToolServer are of type `'TEXT'` and creator `'MPS '`. You can use the MPW editor or any text editor to view and edit these files.

- If the script is executed by using Apple events, ToolServer sends the output back to the client application via an Apple event. Whether the output is buffered or sent immediately depends on the Apple event used to execute the script. For additional information see the section “I/O With Apple Events” in Chapter 4. Please note that the `RShell` command is an instance of executing a script by using the script (`'scpt'`) Apple event.

▲ **WARNING**
ToolServer does not support reading from standard input unless it is redirected from a file or a pipe. Attempts by tools and scripts to read from `Dev:Console`, `Dev:StdIn`, `Dev:Output`, or `Dev:Error` will return an End of File (EOF) error. ▲

Table 2-3 Files defined by `BackgroundOut` and `BackgroundErr`

Variable	Setting	Effect
`BackgroundOut`	none	ToolServer creates a file named *script*`.out`, where *script* is the name of the input script. The file is created in the directory containing the executing script.
	filename	ToolServer creates a file named *filename.*
	`DEV:NULL`	ToolServer does not create an output file.
`BackgroundErr`	none	ToolServer creates a file named *script*.`err`, where *script* is the name of the input script. The file is created in the directory containing the executing script.
	filename	ToolServer creates a file named *filename.*
	`DEV:NULL`	ToolServer does not create an error file.

If ToolServer cannot open the specified output files because they are in use (locked or already open), it opens a new file and generates a new, unique name by appending an ordinal number (0,1,2,...) to the specified output filename.

If ToolServer is unable to write to standard error, it will write errors to the log file `ToolServer.Log`. This log includes messages about errors that occur during the processing of Apple events or the creation of the output or error files.

The log file is overwritten each time you launch ToolServer. If you want to save a copy of the file, you can rename it in your `QuitTS` script.

▲ **WARNING**
Tools and scripts can use the I/O redirection facilities of the MPW Shell language and I/O libraries to read and write files. There is one important difference between running scripts under the MPW Shell and running them under ToolServer: The MPW Shell allows tools and scripts to read the contents of files that have been changed by the editor but have not yet been saved to disk. ToolServer does not intercept file system calls and does not allow tools to read from MPW windows. You must save any changed files before running a tool or script that accesses these files. ▲

Using Alerts to Monitor Script Execution

If you use ToolServer to execute a script locally, you can use the MPW `Alert` command to send messages about script execution.

As the sample script in Listing 2-2 shows, you can include the MPW Alert command (which uses the Notification Manager) to display a message. Otherwise, ToolServer runs unobtrusively in the background, directing diagnostic information to standard error. If an error occurs while attempting to open the standard error file or during initialization, ToolServer writes error information to the file `ToolServer.log`.

Listing 2-2 shows a sample script used to do a build. The build commands output by the Make tool are written to the file `domake.cmd` and executed. Note that the sample script includes some error handling.

- If any of the build commands fail to execute, the script displays an alert box specifying that the make failed and gives the name of the file containing diagnostic information.
- If the build succeeds, standard output and standard error are written to the file `domake.results`, and the script displays an alert box specifying the name of the file containing this information.

Listing 2-2 Sample script

```
Directory 'HD:MPW:Demo:'
Begin
               echo "Beginning Build"
               date
               make -f Demo.make > domake.cmd
               set done 1
               set echo 1
               domake.cmd || begin; alert 'Make failed!!!'; unset done;end;
               set echo 0
               date
End ∑ 'HD:MPW:Demo:domake.results'
               If {done}
                  Alert "Build is complete.∂nSee HD:MPW:Demo:domake.results"
   Else
                  Alert "Build had errors.∂nSee HD:MPW:Demo:domake.results"
   End
   Quit
```

Note
If you are using version 3.3 or later of MPW, you can use the `TraceFailures` variable to obtain more detailed information about the execution of a script. Setting this variable to `TRUE`, causes a message to be sent to standard error, which identifies the location of the failure when a script terminates abnormally. This message is similar in form to compiler diagnostic messages. ◆

C H A P T E R 3

The RShell Command

This chapter explains how you enable communication between ToolServer and the MPW Shell residing on different machines, describes the syntax of the `RShell` command, which is used to send a command line to be executed by ToolServer in the background or on a remote machine, includes two sample scripts, and discusses the limitations on scripts that are run on remote machines.

You need to read this chapter if

- you are using version 3.3 of the MPW Shell, and want to use the `RShell` command to have the MPW Shell on a local machine send a script to be executed by ToolServer in the background on your local machine or on a remote machine
- you need to make use of Apple events to have ToolServer run a tool or script from a development system other than MPW and are interested in an example of a command-line interface to the Apple event mechanism

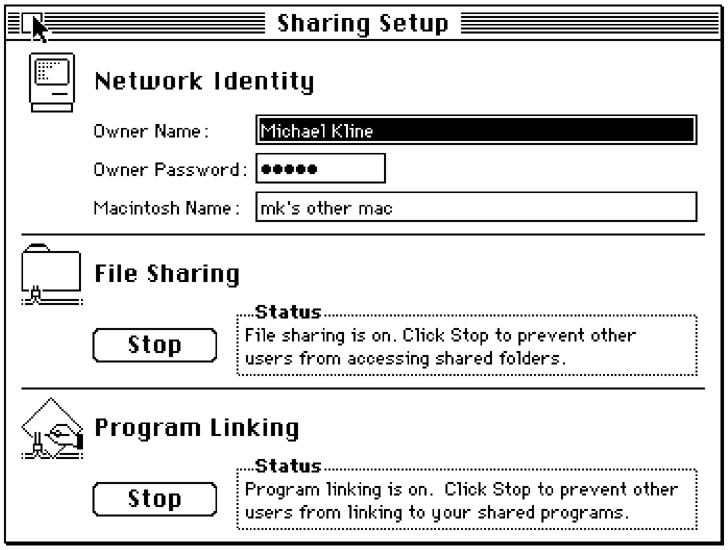
Version 3.2 of MPW does not support the `RShell` command.

Communicating With ToolServer on a Remote Machine

Before you can use the `RShell` command to execute a script on a remote machine, you must enable ToolServer or MPW on a local machine to communicate with ToolServer on a remote machine. The following tutorial describes the procedure you need to follow to do this. For additional information about remote linking and file sharing, see the user documentation provided for System 7.

1. **Install ToolServer on the remote machine.**
2. **Open the Sharing Setup control panel on the remote machine and turn program linking on. Figure 3-1 shows the Sharing Setup control panel. The status panel for program linking indicates that it is turned on. The status panel for File Sharing also shows that File Sharing is on, which you may also want to enable.**

Figure 3-1 Sharing Setup control panel


Sharing Setup

Network Identity

Owner Name: Michael Kline

Owner Password: ●●●●●

Macintosh Name: mk's other mac

File Sharing

Status

File sharing is on. Click Stop to prevent other users from accessing shared folders.

Stop

Program Linking

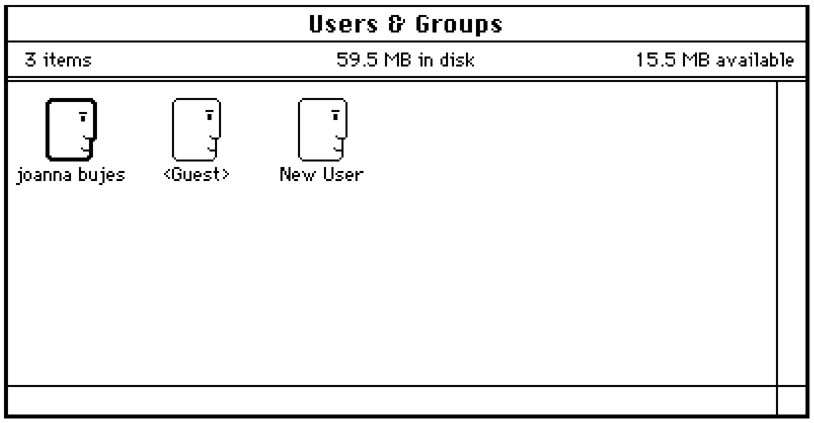
Status

Program linking is on. Click Stop to prevent other users from linking to your shared programs.

Stop

3. **Open the Users & Groups control panel on the remote machine and choose New User from the File menu. A New User icon is added to the Users & Groups control panel as shown in Figure 3-2.**

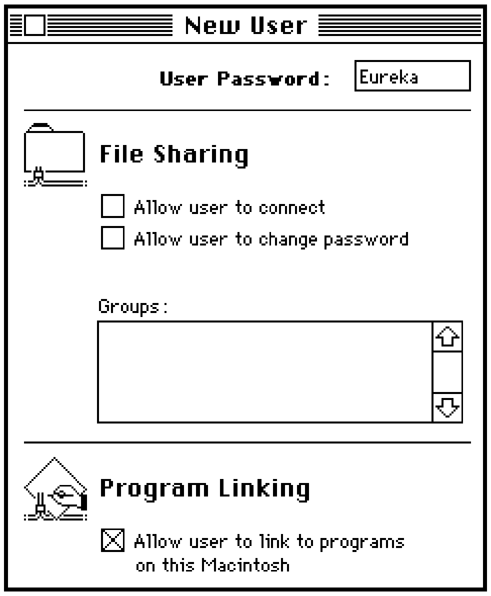
Figure 3-2 Users & Groups control panel


Users & Groups

3 items 59.5 MB in disk 15.5 MB available

4. **Double click the New User icon. When the New User document appears, enter a user password and enable program linking by clicking the Program Linking check box. See Figure 3-3 for an example of an open New User document.**

Figure 3-3 New User document



New User

User Password: Eureka

File Sharing

☐ Allow user to connect

☐ Allow user to change password

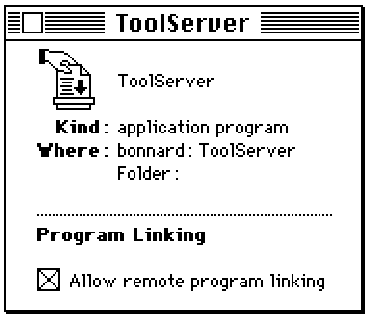
Groups:

Program Linking

☒ Allow user to link to programs on this Macintosh

5. **Select the ToolServer application and choose Sharing from the File menu. A dialog like the one shown in Figure 3-4 is displayed. Make sure the check box "Allow Remote Program Linking" is checked.**

Figure 3-4 Enabling program linking to ToolServer



ToolServer

ToolServer

Kind: application program

Where: bonnard: ToolServer Folder:

Program Linking

☒ Allow remote program linking

6. **Launch ToolServer on the remote machine by double-clicking the ToolServer icon.**

If you have followed the preceding steps, you are now ready to send a script from the local machine to be executed by ToolServer on the remote machine.

7. **Return to the local machine. Launch MPW if you have not already done so and create a new file called `RemoteScript`. Type the following commands into the new file and save the file.**

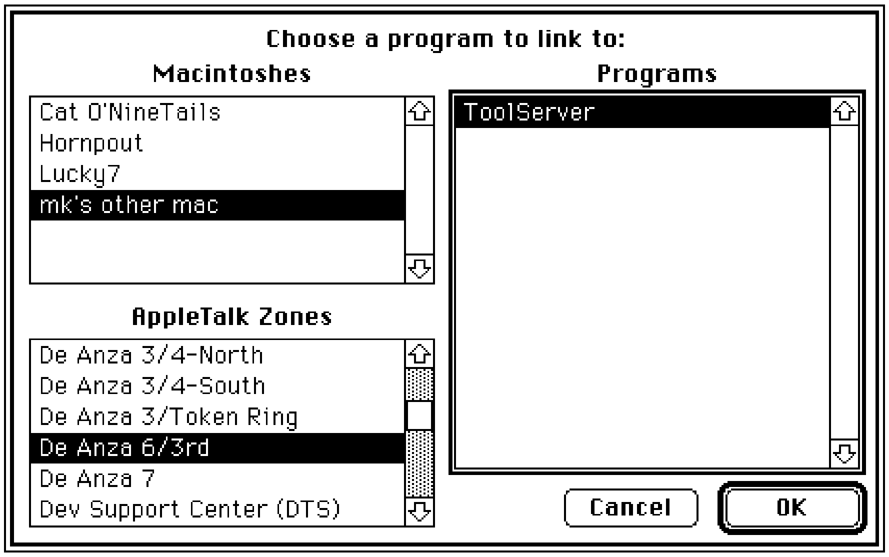
```
beep
beep
beep
echo "done"
```

8. **Enter the following command to send the contents of `RemoteScript` to be executed by ToolServer on the remote machine.**

```
RShell < RemoteScript
```

9. **Because no target was specified with the `RShell` command, the PPC Browser dialog shown in Figure 3-5 is displayed. When invoked by the `RShell` command, the PPC Browser displays all items that respond to ToolServer Apple events and that have been launched. Select the zone, machine, and application to which the script is to be sent and click OK.**

Figure 3-5 PPC Browser dialog


Choose a program to link to:

Macintoshes
- Cat O'NineTails
- Hornpout
- Lucky7
- mk's other mac

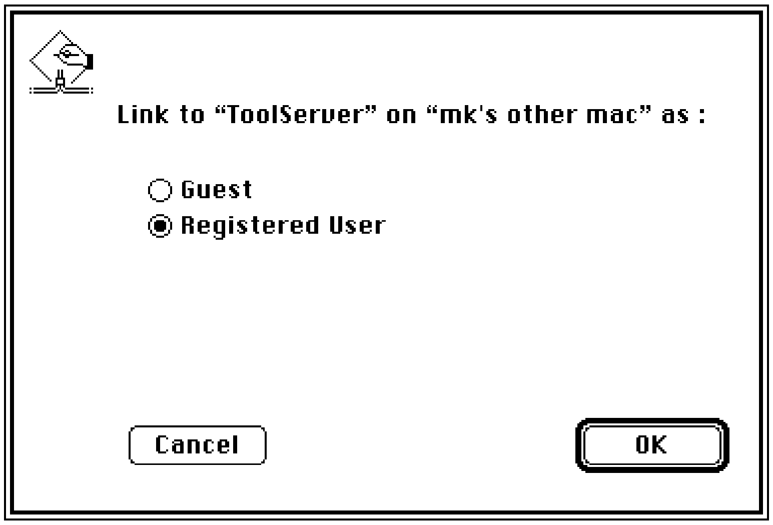
AppleTalk Zones
- De Anza 3/4-North
- De Anza 3/4-South
- De Anza 3/Token Ring
- De Anza 6/3rd
- De Anza 7
- Dev Support Center (DTS)

Programs
- ToolServer

Note that the PPC Browser dialog only displays the ToolServer on a specific machine, not all ToolServers in the zone. If you want to install multiple ToolServers on one machine, you must rename them so that they have unique names; for example, ToolServer1, ToolServer 2, and so on.

10. **When the dialog box shown in Figure 3-6 is displayed, click Registered User, specify your password, and then click OK.**

Figure 3-6 Program linking dialog


Link to “ToolServer” on “mk's other mac” as :

11. **The script is now executed on the remote machine. If you are close enough to the remote machine, you should hear three beeps. You should also see the “done” message echoed back to the currently active MPW window on your local machine.**

Using Files on the Local Machine

The preceding tutorial assumes that any source files needed by the executing script are located on the remote machine. If the source files are located on the local machine, you will also need to do the following.

1. Use the Users & Groups control panel to turn file sharing on for the local machine.
2. Use the Sharing item in the Finder File menu on the local machine and specify the appropriate privileges for the folder(s) containing the files you need to access.
3. Include the `Choose` command in the executing script to log on to these folders (treated as AppleShare volumes). The `Choose` command must precede any command that access these files. For example,

   ```
   Choose LocalZone:LocalMachine:Volume -u Joanna Bujes -pw boo
   ```

   The `-u` option makes the remote user a registered user of your local machine. The `-pw` option specifies the user’s password.

If it is possible for you to respond to user dialogs on the remote machine because the two machines are physically close, you can also use the Chooser to log on to the folders on the local machine.

Using the RShell Command

This section describes the syntax and use of the `RShell` command.

Command Syntax

The syntax of the `RShell` command is:

`RShell` [[`-f`|`-b`|`-r` *target]* [`-q`] [*command* | < *script*]] |
[`-status`|`-c` *id* |`-k` *id*] [> *output*] [≥ *error*]

`-f`
This option sends the Script event to the MPW Shell (version 3.3 or later) running on the same machine where the `RShell` command is executed. If the `RShell` command is included in a script that is executed by ToolServer, and the MPW Shell has not been launched, this option launches it.

`-b`
This option sends the Script event to be executed in the background by ToolServer running on the same machine where the `RShell` command is executed. If the `RShell` command is executed by the MPW Shell and ToolServer has not been launched, this option launches it.

`-r [`*target* `]`
This option sends the command to the application specified by *target*. The application must already be running. The *target* parameter takes the form [[*zone*:] *server*:] *ApplicationName.* The application is either ToolServer or the MPW Shell. The application name is case sensitive.

`-q`
This option quits ToolServer. If *command* follows, the specified command is executed first.

command
This parameter is a string specifying the command to be executed. The command you specify is syntactically equivalent to an MPW command line. If you omit this parameter, `RShell` reads from standard input.

If you use the `-r` option, the specified script or tool is assumed to reside on the remote machine. Note that ToolServer does not support all MPW built-in commands; see Table 2-3, in Chapter 2 for a list of valid commands.

script
This parameter specifies the name of a script on the local machine whose contents are redirected to ToolServer. If the script name contains a space, it must be quoted.

`-status`
This option displays information about scripts that are currently running or about pending RShell requests—scripts waiting to be run. Only those RShell requests that you have initiated are displayed in the status report. Standard input is ignored.

`-c` *id*
This option closes the files associated with the request whose transaction ID is *id.* Standard input is ignored. See "Transaction ID of a Request" on page 3-9 and "Redirecting Output and Abnormal Termination" on page 3-10 for more information.

`-k` *id*
This option aborts the script whose transaction ID is *id*. Standard input is ignored. See "Transaction ID of a Request" on page 3-9 and "Redirecting Output and Abnormal Termination" on page 3-10 for more information.

output
This parameter is the name of the file to which standard output is redirected. See "Input, Output, and Diagnostic Files" in Chapter 2 and "Redirecting Output and Abnormal Termination" on page 3-10 for additional information.

error
This parameter is the name of the file to which standard error is redirected. See "Input, Output, and Diagnostic Files" in Chapter 2 and "Redirecting Output and Abnormal Termination" on page 3-10 for additional information.

Note
The `RShell` command reads from standard input if neither *command,* `-k`, `-c`, `-status`, nor `-q` is specified.

If you don't specify any of the `-f`, `-b`, or `-r` options to select a target application, an interactive window called the PPC Browser is opened to allow you to select the application to receive the command. Figure 3-5 shows a sample PPC Browser. ◆

Examples

The following examples illustrate the use of the `RShell` command:

```
RShell -q                  # quit ToolServer
RShell "beep"              # execute the beep command
RShell "beep" -q           # execute the beep command and quit
RShell < myscript          # execute the contents of the file
myscript
```

The command

```
RShell < myscript
```

or

```
RShell < "my script"
```

redirects the contents of a file named `myscript`, to be found in the current directory for the local (sending) machine, to the `RShell` command. The command

```
RShell MyScript
```

or

```
RShell "My Script"
```

sends the specified script name to ToolServer, which locates the script in ToolServer's current directory.

Variable Substitution in the RShell Command Line

The type of quotes (single or double) you use to delimit the command string you specify for the `RShell` command determines where variable substitution occurs.

If you specify the command

```
RShell -b "echo {BackgroundShell}"
```

the sending application (usually the MPW Shell) evaluates `{BackgroundShell}` and sends the resulting command to ToolServer. If you specify the command, because of the hard quotes ('), the command line is sent as is to ToolServer, which evaluates `{BackgroundShell}` and performs variable substitution.

```
RShell -b 'echo {BackgroundShell}'
```

Transaction ID of a Request

When you specify a script, tool, or command to be sent by the `RShell` command, the command assigns each request a transaction ID and places it in a queue. Multiple requests to one or more applications might be outstanding at any time. If you use the `-status` option to the `RShell` command, it displays information about requests still in the queue that were issued by your application. Listing 3-1 shows the output of the `-status` option.

Listing 3-1 Output of RShell -status option

```
rshell -status
ID       Request            Current Command       Server
--       -------            ---------------       ------
 1       makescript         link                  ToolServer
 2       BuildScript        duplicate             AZone:AMachine:ToolServer
 3       TestScript                               ToolServer
```

In the status shown in Listing 3-1, two requests are active on two different ToolServers. The current command column is empty for request 3; a blank current command field usually indicates that the script is queued, waiting for ToolServer to complete a previous script.

The transaction ID of a script appears in the first column of the status display. You can use this ID with the RShell `-k` or `-c` option as described in the next section.

Redirecting Output and Abnormal Termination

When you use the `RShell` command to send a command or script, the `RShell` command keeps open its standard output and standard error files in case the command or script directs output to `Dev:Output` or `Dev:Error` respectively. Thus files are kept open for each outstanding RShell request. For example, the following command opens `outputfile` and `errfile`, and then writes output and diagnostic output to these files. The sending application (ToolServer or the MPW Shell) closes these files when the script completes.

```
RShell 'myScript > dev:output ≥ dev:error' > outputfile ≥ errfile
```

- If you do not specify output redirection with the `RShell` command, and the sending application is ToolServer, output is sent to wherever the issuing ToolServer's `Dev:Output` and `Dev:Error` are currently directed.
- If you do not specify output redirection with the `RShell` command, and the sending application is the MPW Shell, output is sent to the window in which the `RShell` command (or the script containing it) was executed.

  This can cause a problem when more than one process attempts to write to the same window or file. For example, you execute an `RShell` command from the Worksheet and do not specify redirection. You then choose the Check Out item from the Projector menu and find that you are not able to. The following error message is displayed:

  ```
  ###ExperShell-a15 - Unable to open "Athena:MPW:Worksheet"
  # The read/write permission of only one access path to a file
  can allow writing (OS error -49)
  ```

  The cause of the error message is that both the `RShell` command and the Check Out item from the Projector menu redirect output to the same window, in this case the Worksheet window. If this happens to you, you can either choose the menu item from another window or you can use the `-k` *id* option of the `RShell` command to kill the process.The same situation can arise if you execute a number of `RShell` commands from the same window without specifying redirection. If you find yourself in this situation, you can use the `-k` *id* option to kill the contending processes. Best of all, you can avoid trouble by always redirecting `RShell` command output to a file.

If a script terminates abnormally—due, for example, to a broken network connection or to a tool that crashes—RShell's standard output and standard error files might remain open. In this case, you need to close the standard output and standard error files associated with the request by using the `-c` *id* option of the `RShell` command, where *id* specifies the transaction ID of the request.

The `-c` *id* option is not intended for use with an executing script; it is provided to allow you to cleanup output files when the connection between the client (the MPW Shell or the ToolServer that ran the `RShell` command) and the server (the MPW Shell or the ToolServer that received and processed the command) has been broken. If a script is running, do not use the `-c` *id* option. Kill the process instead with the `-k` *id* option.

▲ **WARNING**
It is strongly recommended that you do not use the `-c` *id* option on a running ToolServer process or on a ToolServer process awaiting execution. Doing so makes it impossible to kill the process from the sending application; it can only be stopped (it must start executing) from the ToolServer executing it by aborting the process with a Command-period. ▲

Sample Scripts

This section describes the use of the `RShell` command to execute scripts. It describes two scripts: one that sends compilations to ToolServer running in the background and one that sends compilations to ToolServer running on a remote machine.

A Script That Compiles in the Background

The script shown in Listing 3-2 sets the current directory, searches the directory for files terminating in `.c`, and compiles each such file.

If the compile does not terminate normally, an `RShell` command within the script sends MPW an `Alert` command that specifies the name of the file that did not compile successfully. When the script is done, another `RShell` command sends MPW an `Alert` command that displays the message “Done.”

The `RShell` command used to send the script shown in Listing 3-2 to ToolServer is

```
RShell -b "Background Compile"
```

▲ **WARNING**
Under low memory conditions, using the `-b` option to the `RShell` command may launch ToolServer in a partition smaller than the one requested in ToolServer’s size resource. Subsequent operations performed within ToolServer might then fail because of insufficient memory. The notification given varies with the particular tool that has failed. In extreme cases the Apple events mechanism might not have enough memory to reply to the MPW Shell. In that case the `-status` option to the `RShell` command will show that operations are still pending. ▲

Listing 3-2 Background compile script

```
Set Exit 0
Directory "hd:ToolServer Demo"
For file In ≈.c
      (Evaluate "{file}" =~ /(≈)®1.c/) > Dev:Null        # set {®1} to the
                                                         # part of the name
                                                         # that is to the
                                                         # left of ".c"
      c "{file}" -o "{®1}".o ≥ "{®1}".err
      If {status} != 0
            RShell -f "Alert 'Errors in ∂"{file}∂"'"
      End
End
RShell -f "Alert Done"
Set Exit 1
```

A Script That Compiles on a Remote Machine

Before you can use the `RShell` command to communicate with ToolServer on a remote machine, you need to configure the local and remote machines according to the instructions given in "Communicating With ToolServer on a Remote Machine" on page 3-2.

Like the script shown in Listing 3-2, the script shown in Listing 3-3 sets the current directory, searches the directory for files ending in `.c`, and compiles them.

The `RShell` command used to send the script shown in Listing 3-3 to ToolServer is

```
rShell -r "TSZone:TSServer:ToolServer" < "Remote Compile" ∑ MyOut
```

where `Remote Compile` is the script to be executed. `TSZone` and `TSServer` are respectively the zone and server where ToolServer resides.

If you compare the scripts shown in Listing 3-2 and Listing 3-3, you will notice that the `RShell` commands in Listing 3-2 have been replaced with `Echo` commands in Listing 3-3. This is because `RShell` commands in a remote script will cause program linking dialogs to be generated on the remote machine. This will cause problems if the user cannot see the screen of the remote machine and cannot respond to the dialogs. In such a case, you must pass all status information back to the local machine using standard output or standard error.

Listing 3-3 Remote compile script

```
Set Exit 0
For file In "ToolServer Demo:"≈.c   # find all files whose names end in
                                    # ".c"
   (Evaluate "{file}" =~ /(≈)®1.c/) > Dev:Null  # set {®1} to the part of
                                                # the name that is to
                                                # the left of ".c"
   c "{file}" -o "{®1}".o ≥ "{®1}".err
   If {status} != 0
      echo "Errors in  ∂"{file}∂""
   End
End
   echo "Done!"
Set Exit 1
```

The output of the `RShell` command used to send the script to the remote machine can be directed to a window on the local machine as follows:

```
RShell -r "TSZone:TSServer:ToolServer" < "Remote Compile" ∂
            ∑ Remote.log
```

▲ **WARNING**
If you are issuing several `RShell` commands at the same time, it is best to redirect output from these commands to separate log files or windows. The problems that can arise if you do not are described in "Redirecting Output and Abnormal Termination" on page 3-10. ▲

Shortcuts

Though the `RShell` command provides a great deal of power and flexibility to script writers, it can be somewhat complicated to use. An MPW Shell startup file named `Userstartup•ToolServer` (in the `Examples` folder) simplifies using ToolServer from the MPW Shell. The startup file provides

- a command-key definition (Command-\) that sends the current line or selection to ToolServer for execution
- extensions to the build menu for doing background builds
- a sample script that synchronizes variables and aliases between the MPW Shell and ToolServer

A `Quit•ToolServer` script, also in the `Userstartup•ToolServer` file, checks for outstanding RShell requests before exiting MPW.

C H A P T E R 4

Using Apple Events to Communicate With ToolServer

This chapter describes ToolServer's support of Apple events. An Apple event is a high-level event that adheres to the Apple Event Interprocess Messaging Protocol (AEIMP). By using the routines of the Apple Event Manager your application can use Apple events to communicate with ToolServer in a standard way.

A transaction involving Apple events is initiated by a client application, which sends an Apple event to request a service. The application providing the service is called a server application. For complete information about sending and processing Apple events, see the Apple Event Manager chapter of *Inside Macintosh,* Volume VI.

You should read this chapter if you need to extend the functionality of your application by making use of ToolServer's execution environment. See the following examples.

- A client application has a text editor but no development environment. The client can use Apple events to send text input (like a `BuildProgram` or `Link` command) to ToolServer, which can execute the command and return output as the client directs.
- A client application consists of a development environment that has no resource compiler or decompiler. The client can use Apple events, triggered perhaps by a menu selection, to send `Rez` and `DeRez` commands to ToolServer.

Apple events can be sent locally or across a network. The Users & Groups control panel provides an interface for controlling Apple events access across the network. Multiple users can use a single ToolServer running on a remote server; the users share the ToolServer state such as shell variables, aliases, and the working directory. Complete information about the configuration required to enable remote communication is provided in the section "Communicating With ToolServer on a Remote Machine" in Chapter 3.

ToolServer as a Server Application

This section introduces the three kinds of Apple events that are understood by ToolServer. Some of these events are documented in other publications; this chapter supplies additional information about how ToolServer handles these events. Events that are specific to ToolServer and MPW are only documented in this chapter.

ToolServer supports the following kinds of events:

- required events

  The four events that an application must handle to support Apple events are Open Application, Open Documents, Print Documents, and Quit. These events are defined in the *Apple Event Registry: Standard Suites*.

- miscellaneous events

  ToolServer supports the 'dosc' (do script) event, which is defined in the Miscellaneous suite of the *Apple Event Registry: Standard Suites*.

- MPW custom events

  ToolServer uses the Script ('scpt'), Output ('outp'), Diagnostic ('diag'), Status ('diag'), and Abort ('abrt') events to control script execution.

Figure 4-1 shows the possible flow of events between a client application and ToolServer. Because you cannot use a high-level event to launch ToolServer, it must already be running when your application sends an Apple event.

The Apple Event Manager defines a standard reply that includes an error number ('errn') and an error string ('errs'). Events may also include additional parameters in the reply. These parameters are defined along with the events in the following sections.

Note
You can send Apple events to the MPW Shell as well as to ToolServer, but only ToolServer returns output. For additional information, see Appendix A, "Apple Event Support in the MPW Shell." ◆

Figure 4-1 Apple events

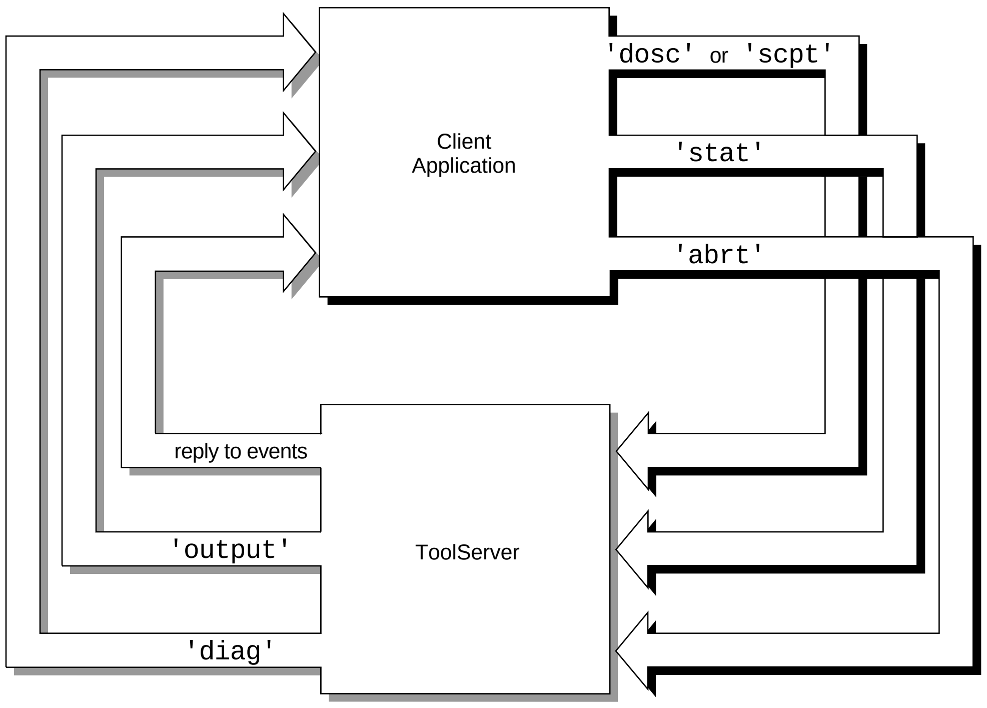


Table 4-1 lists the events supported by ToolServer. The sections “Required Events” on page 4-6 and “Script Events” on page 4-7 describe in detail the attributes and parameters for the events listed in Table 4-1.

Table 4-1 Apple events

ID	Class	Effect
`'oapp'`	`'aevt'`	Finder sends the Open Application (`'oapp'`) event when ToolServer is launched and no documents are selected to be executed or printed. The reply contains `'errn' == noErr.` This event is included because it is a required event. It would normally be used by an application that opens an untitled document in response to the user’s double-clicking the application. Because ToolServer does not create new documents, it is not affected by this event.
`'odoc'`	`'aevt'`	The Open Documents (`'odoc'`) event contains a list of files that ToolServer is to open. ToolServer treats these documents as scripts and executes them. Because the client application does not expect parameters in the reply beyond an error number and error string, scripts executed by using this event do not use Apple events to return output. All input and output must be through files. For additional information see the section “I/O With Apple Events” on page 4-5.
`'pdoc'`	`'aevt'`	The Print Documents (`'pdoc'`) event causes ToolServer to execute the print tool to print the specified documents. As in the Open Documents case, the event returns no output to the client application.
`'quit'`	`'aevt'`	The Quit (`'quit'`) event terminates ToolServer. If ToolServer is executing a tool or script, the Quit event is queued until the script or tool completes. As required by the Apple Event Manager, ToolServer replies to the event and then proceeds with the Quit event. Because quitting involves a number of steps, including the execution of a script, it is possible for the Quit event to fail even though a good status has been returned.
`'dosc'`	`'misc'`	The Do Script (`'dosc'`) event sends an MPW command line to be executed by ToolServer. This event causes ToolServer to buffer script output and return it as the direct parameter of the reply. For additional information see the section “I/O With Apple Events” on page 4-5.
`'scpt'`	`'MPS '`	The Script (`'scpt'`) event sends an MPW command line to be executed by ToolServer. This event returns a series of events containing the text written to `Dev:Output`, `Dev:Error`, or Dev:Console. See “I/O With Apple Events” on page 4-5 for additional information.
`'outp'`	`'MPS '`	The Output (`'outp'`) event returns standard output to the client application.
`'diag'`	`'MPS '`	The Diagnostic (`'diag'`) event returns diagnostic output (standard error) to the client application.
`'stat'`	`'MPS '`	The Status (`'stat'`) event is used by the client application to determine the current status of ToolServer.
`'abrt'`	`'MPS '`	The client application uses the Abort (`'abrt'`) event to abort the currently executing script.

Executing a Script by Using Apple Events

In general there are two ways to execute a script by using Apple events:

- by sending the Open Documents event to ToolServer

  If you use this method, ToolServer sets the working directory to the one containing the script.
- by sending ToolServer either the Script event or the Do Script event

  These events are designed so that development environments such as the MPW Shell or other applications can send MPW commands or scripts to ToolServer and receive the output back. ToolServer's working directory is not changed before running the script in this case.

Rather than receiving a specific file or document, ToolServer gets a buffer of text to be interpreted as an MPW Shell command line or script. ToolServer interprets the text as if it had been entered interactively to the MPW Shell. Since the MPW Shell is designed to "read" scripts to execute them, ToolServer creates a temporary file to contain the text and saves this file in the folder specified by the `{TempFolder}` variable, which is set to the system's temporary folder by default. You can have ToolServer save the file elsewhere by resetting the value of this variable in your `UserstartupTS` file.

ToolServer responds to one script event at a time. `'odoc'`, `'pdoc'`, `'quit'`, `'dosc'`, and `'scpt'` events received while ToolServer is running a script are queued until that script completes. ToolServer responds to `'stat'`, `'abrt'`, and `'oapp'` events even if it is handling a script or print event.

I/O With Apple Events

If you use the Script event or the Do Script event, you can have the output returned via the Apple event mechanism by directing output to the pseudofilename `Dev:Output` and diagnostic messages to `Dev:Error`. ToolServer treats anything directed to `Dev:Console` (specifically or by default) as if it were written to `Dev:Output`. The pseudofilenames `Dev:Output` and `Dev:Error` are used exclusively by ToolServer.

How ToolServer returns output to an application by using an Apple event *and* directing output to `Dev:Output` and `Dev.Error` depends on the event used to execute a script:

- If the application uses the Script event (`'MPS ' 'scpt'`), the sender should use the send mode `kAEQueueReply`. The sender is then able to receive events while waiting for the ToolServer reply. ToolServer generates a series of output and diagnostic output (error) events. These events contain the characters written to `Dev:Output` and `Dev:Error` respectively. The reply to the Script event itself contains the final MPW status (value of the MPW {`status`} variable).
- If the application uses the Do Script event (`'misc' 'dosc'`), ToolServer buffers output and diagnostic output. When the event is completed, ToolServer returns

these buffers as part of the reply to the event. Output is placed in the direct object parameter of the reply, and diagnostic output is placed in the '`diag`' parameter of the reply.

Because Apple event messages are currently limited to 64 KB bytes, the combined total of these buffers must be slightly less than this. The reply to the `'dosc'` event also contains the final MPW status value and a system error value. If the message buffer overflows, ToolServer returns the first 64 KB bytes of data and the system error message indicates `bufferIsSmall (osErr = -607)`.

Required Events

The *Apple Event Registry: Standard Suites* defines four required events: Open Application, Open Documents, Print Documents, and Quit. This section describes the attributes and parameters used by these events.

Open Application Event

`Message class:`	`'aevt'`
`Message ID:`	`'oapp'`
`Parameter Keyword:`	`no parameters`

The reply contains `'errn' == noErr`. You cannot use this event to launch ToolServer.

Open Documents Event

`Message class:`	`'aevt'`
`Message ID:`	`'odoc'`
`Parameter Keyword:`	`'----'`
`Parameter Type:`	`'list'`
`Parameter Data:`	`a list of documents to open`

Scripts executed via the Open Documents event do not return output via Apple events. All input and output must be through files. Refer to the section “I/O With Apple Events” on page 4-5 for additional information.

Print Documents Event

`Message class:`	`'aevt'`
`Message ID:`	`'pdoc'`
`Parameter Keyword:`	`'----'`
`Parameter Type:`	`'list'`
`Parameter Data:`	`a list of documents to print`

As in the Open Documents case, no output is returned to the sender.

Quit Event

```
Message class:                  'aevt'
Message ID:                     'quit'
Parameter Keyword:              no parameters
```

Script Events

You use script events, which include one miscellaneous event and four MPW events, to execute a script, return status information, return output and diagnostic information, and abort a script. The following sections describe the attributes and parameters of these events as well as the reply returned by ToolServer.

Script and Do Script Events

The Script and Do Script events are used to execute scripts. The Do Script event is defined in the *Apple Event Registry: Standard Suites*; the Script event is an MPW-specific event. Both events are shown below:

```
Message class:                  'misc'
Message ID:                     'dosc'
Parameter Keyword:              '----'
Parameter Type:                 'TEXT'
Parameter Data:                 string containing command line

Message class:                  'MPS '
Message ID:                     'scpt'
Parameter Keyword:              '----'
Parameter Type:                 'TEXT'
Parameter Data:                 string containing command line
TransactionID:                  optional
```

Both events contain a single parameter of type `'TEXT'` that is a string containing a command line to be executed. This line can specify a single tool or script, or it can contain a series of command lines separated by semicolons or return characters.

When sending the '`scpt`' or `'dosc'` event, the sender can specify an optional transaction ID to identify the event. The sender can use the transaction ID later to obtain the status of this script event, or to abort it.

The Script event ('`MPS `', '`scpt`') can return `'diag'` and `'outp'` events containing the text written to `Dev:Output`, `Dev:Error`, or `Dev:Console`. These events are defined in the next section, "Output and Diagnostic Output Events." The Do Script event ('`misc`', '`dosc`') will buffer this data, and return it in the Apple event reply.

The Apple event reply contains the parameters shown in Table 4-2 and Table 4-3. The 'errn' parameter contains any system errors reported while handling the event or constructing the reply. The 'stat' value contains the MPW status variable set as a result of running the script. The direct ('----') and 'diag' parameters are set in the Do Script event to contain standard output and standard error (diagnostic output) respectively. If no output or error data is produced, these parameters are not set.

Table 4-2 Parameters to Script reply

Parameter	Type	Description
`'errn'`	`typeLongInteger`	system errors
`'stat'`	`typeLongInteger`	MPW status value

Table 4-3 Parameters to Do Script reply

Parameter	Type	Description
`'errn'`	`typeLongInteger`	system errors
`'stat'`	`typeLongInteger`	MPW status value
`'----'`	`typeChar`	standard output
`'diag'`	`typeChar`	diagnostic output

Output and Diagnostic Output Events

These events are used to return standard output ('outp') and standard error ('diag') to the sender of a Script event. The transaction ID of the event matches the transaction ID specified in the Script event that resulted in these events.

```
Message class:          'MPS '
Message ID:             'diag'
Parameter Keyword:      '----'
Parameter Type:         'TEXT'
Parameter Data:          a stream of characters
TransactionID:           same as that of scpt event
```

```
Message class:          'MPS '
Message ID:             'outp'
Parameter Keyword:      '----'
Parameter Type:         'TEXT'
Parameter Data:          a stream of characters
TransactionID:           same as that of scpt event
```

The libraries and device handlers might buffer output, so these events can contain an arbitrary amount of text (up to 64 KB bytes). The C stdio library allows you to change the size of its buffers.

Abort Event

The Abort event aborts the currently executing script.

```
Message class:                   'MPS '
Message ID:                      'abrt'
Parameter Keyword:               no parameters
TransactionID:                   optional
```

If the sender of the event does not specify a transaction ID, ToolServer aborts the currently executing script as if the user had entered a Command-. from the keyboard. If a transaction ID is specified, this ID and the sender of this event must match the ID and sender of the currently executing script. The script is aborted only if these two parameters match. The error number `ProcNotFound` is returned if no abort occurred.

Status Event

The Status event determines the current status of the ToolServer.

```
Message class:                   'MPS '
Message ID:                      'stat'
Parameter Keyword:                no parameters
TransactionID:                    optional
```

If the transaction ID and sender of the status event match those of the currently executing script, or if no transaction ID is specified, the reply '`errn`' is set to `noErr(0)`, and the four additional parameters listed in Table 4-4 are returned.

Table 4-4 Parameters to Status event reply

Parameter	Type	Description
`'what'`	`typeChar`	file name of the script in progress
`'pos '`	`typeLongInteger`	number of characters read from the file
`'size'`	`typeLongInteger`	the size of the file
`'who '`	`typeChar`	the name of the current command or tool

If the transaction ID does not match that of the currently executing script, the error number `ProcNotFound` is returned. If the shell is not running a script, ToolServer sets the error number to `noErr`, and returns no other parameters.

Apple Event Support in the MPW Shell

Version 3.3 or later of the MPW Shell supports the Apple events described in Chapter 4 of this manual. However, due to the interactive nature of the MPW Shell and the design of the MPW Shell and the editor, this support is somewhat limited compared to the support provided by ToolServer. The differences and limitations are:

- The MPW Shell's reply to an Apple event is merely an acknowledgment that the event was received properly; it does not indicate the success or failure of the request. The reason for this is that the MPW Shell replies to the event before it processes the event.
- Because the MPW Shell replies immediately to script events, the Abort and Status events function only when the transaction ID is not set (specified as `kAnyTransactionID`). The MPW Shell does not match the sender of this event with the sender of the script.
- Because the MPW Shell has a console, it does not support the `Dev:Output` or `Dev:Error` devices nor return console output to the sender of a script event.
- The MPW Shell supports the four required events (Open Application, Open Documents, Print Documents, and Quit). However, the Open Documents event opens the files in the editor and does not execute a script, as is the case with ToolServer.
- The MPW Shell accepts the Script and Do Script events. However, it replies immediately and then executes the script as if the user had run the script via a user-defined menu item. Standard output and standard error default to the console, which is generally the frontmost window. The error number in the reply indicates that the script was successfully received, not the success or failure of the script itself.

Version 3.3 of MPW can also communicate with ToolServer by using the `RShell` command that has been added to both the MPW Shell and ToolServer. This command is described in Chapter 3, "The RShell Command."

A P P E N D I X   B

ToolServer Files and Variables

This appendix provides summary information about the files created by ToolServer and the variables used by ToolServer.

ToolServer Files

Table B-1 describes the files included with ToolServer and the files ToolServer creates during execution.

Table B-1 ToolServer files

File	Use
`StartupTS`	The contents of this script are executed at launch time. The script sets the default value of ToolServer variables, calls the script `UserstartupTS`, and then calls any additional user-defined startup scripts named `UserStartupTS•`*name*.
`UserStartupTS`	User-defined script that specifies commands to be executed after the `StartupTS` script is executed. This script can be used to set new default variables or to change the values of variables set in the `StartupTS` script.
Scriptname`.err`	File created by ToolServer when a script is executed using the Execute Script menu item and the execution of the script results in text written to standard error.
Scriptname`.out`	File created by ToolServer when a script is executed using the Execute Script menu item and the execution of the script results in text written to standard ouput.
`QuitTS`	File executed as part of the quit process. Commands included in this file search for additional scripts to be run at quit time. Such scripts must be named `QuitTS•`*name* and be located either in the ToolServer folder or in the MPW Preferences folder. By default, ToolServer overwrites the `ToolServer.log` file each time it runs, the `QuitTS` script includes a command which you can uncomment to save a copy of this file.
`ToolServer.log`	File created by ToolServer to which messages are directed about errors that occur during the creation of the output or error files or the processing of Apple events. By default, ToolServer overwrites the `ToolServer.log` file each time it runs, the `QuitTS` script includes a command which you can uncomment to save a copy of this file.
`Userstartup•ToolServer`	File containing scripts that provide shortcuts in the use of ToolServer from MPW 3.3.
`Dev:Output`	ToolServer pseudodevice to which standard output is sent if a script does not specify a file to which standard output is redirected.
`Dev:Error`	ToolServer pseudodevice to which standard error is sent if a script does not specify a file to which diagnostic output is redirected.
`Dev:Null`	ToolServer pseudodevice to which standard output or standard error may be redirected. Standard output or standard error information redirected to this device is then discarded.

ToolServer Variables

The following variables are set in ToolServer's `StartupTS` file.

Table B-2 ToolServer variables

Variable	Use and default value
`BackgroundShell`	True for ToolServer.
`Boot`	The boot disk.
`SystemFolder`	The directory that contains the System and the Finder.
`ShellDirectory`	The directory that contains ToolServer.
`Status`	The result of the last command executed.
`User`	Automatically defined to the name that appears in the Chooser.
`MPW`	The volume or folder containing ToolServer. If you place ToolServer outside this folder, you should redefine this variable to the location of the ToolServer folder.
`Commands`	Directories to search for commands. The default setting is `":{MPW}Tools:, {MPW}Scripts:"`.
`PrefsFolder`	The directory to search for `UserStartupTS` and `QuitTS` scripts: by default `{SystemFolder}Preferences:MPW`. You can change the value in the Startup file or create an alias named MPW in the preferences folder to reference the folder where you keep your scripts.
`TempFolder`	The directory for temporary items. ToolServer may create some temporary items here. The default value is `"{Boot}Temporary Items"`.
`AIncludes`	The directories to search for assembly language include files. The default value is `"{MPW}Interfaces:AIncludes: "`.
`Libraries`	The directory that contains shared libraries. The default value is `"{MPW}Libraries:Libraries "`.
`CIncludes`	The directories to search for C include files. The default value is `"{MPW}Interfaces:CIncludes "`.
`CLibraries`	The directory that contains C libraries. The default value is `"{MPW}Libraries:CLibraries "`.
`PInterfaces`	The directories to search for Pascal interface files. The default value is `"{MPW}Interfaces:PInterfaces "`.
`PLibraries`	The directory that contains Pascal libraries. The default value is `"{MPW}Libraries:PLibraries "`.
`RIncludes`	The directory that contains Rez include files. The default value is `"{MPW}Interfaces:RIncludes "`.

Table B-2 ToolServer variables (continued)

Variable	**Use and default value**
`WordSet`	Character set that defines words for searches and double-clicks. The default value is `'a-zA-Z_0-9'`.
`PrintOptions`	Determines the options used by the print tool. The default `-h` setting, prints headers containing the name of the file, the page, and the time.
`Exit`	Determines whether command files terminate after first error: 0 not to terminate; non-zero to terminate. The default value is `1.`
`Echo`	Determines whether commands are echoed before execution: 0 not to echo; 0 to echo. The default value is 0.
`Test`	Determines whether tools and applications are executed: 0 to execute; non-zero not to execute. The default value is 0.

Index

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software. Proof pages and final pages were created on an Apple LaserWriter IINTX printer. Line art was created using Adobe™ Illustrator. PostScript™, the page-description language for the LaserWriter, was developed by Adobe Systems Incorporated.

Text type is Palatino® and display type is Helvetica®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Apple Courier.

WRITER
Joanna A. Bujes

EDITORS
Loralee Windsor and Wendy Krafft

ILLUSTRATOR
Dave Olmos

PRODUCTION EDITOR
Teresa Lujan

Special thanks to Judy Ketenhoffen, Mark Cleveland, Brian Strull, Keithen Hayenga, and Herb Kanner.