Release Notes

MPW Shell, ToolServer, SourceServer

v. 3.6b2

Contents

General Information

- It is recommended that MPW Shell, ToolServer, and SourceServer be run using StdCLib 3.5d9 or later. This will ensure that a serious bug in the library's signal handling mechanism does not exist in the version of the library being used by the shell and tools. (Note: Mac OS 9.0 includes StdCLib 3.5) While an earlier version of the applications (3.5b2) was built to require the updated library, that requirement has been backed out for 3.5 and later, due to the confusion it caused when the Mac OS refused to launch MPW without the update library in place.

Enhancements in v. 3.6d4 (Feb. 2001 Dev CD)

- Added support to allow the MPW Shell to run in the MultiUser environment, thereby removing the restriction that was added late in the 3.5 development effort. The shell now registers itself with the MultiUser Control Panel, notifying MultiUser to allow all File Manager calls originating from within the MPW environment to proceed without any oversight or alteration on the part of MultiUser.

Note: This implementation completely bypasses the file privileges enforced by the MultiUser environment. When a non-owner user is using MPW, they will have complete access to the file system, as if MultiUser were turned off. If you are the administrator of such a system, use extreme caution in determining who to allow to run MPW.

Projector

Enhancements in v. 3.6d4 (Feb. 2001 Dev CD)

- Changed logic used to initialize the {User} environment variable. MPW will now recognize the MultiUser environment, and set the variable appropriately.

Enhancements in v. 3.6d2 (Web-only release)

- When option-clicking the CheckOut button in the CheckOut window, or when the -open option is used with the CheckOut command, the filetype of the file being checked out is examined. If the file is a TEXT file, the file is opened in an editor window, as it has been in the past. If the file is any other type, an AppleEvent will be passed to the Finder to open the document with the appropriate application (as determined by the Finder).

- Expanded -noWarn option of the CheckOut command to suppress warnings of the form 'File "foo" is not part of project "bar"' when a revision name is provided in the command, but no matching revision name is found in the project.

Enhancements in v. 3.6d1 (Aug. 2000 Dev CD)

- The root project list in the CheckOut window is now sorted in alphabetical order.

- Opening a ProjectorDB file from the Finder now mounts that project in the shell.

Bug fixes in v. 3.6d5 (Web-only release)

- Implemented progress messages in ObsoleteProjectorFile and UnobsoleteProjectorFile.

Bug fixes in v. 3.6d3 (Web-only release)

- Fixed uninitialized variable that was causing the resizing of Projector windows to malfunction.

Bug fixes in v. 3.6d2 (Web-only release)

- Added logic to the Projector list creation functions to check the size of the List Manager DataArray as it's being created. If it's about to exceed the List Manager limit (32000 bytes), a dialog is presented to the user to notify her of the condition, and no additional items are added to the list.

- Added check in the NameRevisions command for a failure to find a FileRec for a given file, and report the error if encountered. Previously, the logic would use a left-over FileRec if the proper FileRec wasn't found, and so ended up using an invalid revision number for a given file. This usually happened when the FileRecs in a ProjectorDB were not ordered in descending fileID order.

Bug fixes in v. 3.6d1 (Aug. 2000 Dev CD)

- Fixed problem in DuplicateNameRevisions command, in which an invalid entry could be produced in the symbolic nametable if the new revision name is alphanumerically "less than" the original name. The problem was that the logic was creating a pointer to the fileid/revid pair list that was in the original name entry, then passing that pointer to the function that adds a name to the list. Since the list must be sorted, the addName function was moving table entries down in the list, thereby invalidating the passed pointer. Fixed by first making a copy of the fileid/revid pair list, then passing the handle to that copy in.

- When performing a CheckIn of a file that had previously been checked out non-modifiable by a new Author (but had been MRO'd), the userID of 0 that was stored in the ckid resource is now treated as a wildcard, and will match the userID that's created at CheckIn time. Projector will no longer complain that another user had checked out the file (via a dialog), and SourceServer will no longer emit an error message for this case.

- Command-clicking on the title of a Projector window no longer crashes.

Editor

Clarifications

- Regarding Setkey: it is possible to assign commands to the main ‘-‘ key and the keypad ‘-‘ key, although it may not be completely obvious how to do so. Normally, if you use the ‘-‘ key in the SetKey command, it will be interpreted as either an option specifier, or as a separator character between two key names. To specify one of the ‘-‘ keys as the key name, use the name “dash” for the main keyboard ‘-‘ key, and use “kpminus” for the keypad ‘-‘ key.

- Regarding SetKey: the length of the command string that can be assigned via the SetKey command is limited to a maximum of 255 characters. This is because the command string is stored in an ‘STR#’ resource in memory, and the limit is inherent in the structure of the resource. To attach a longer command sequence to a particular key, define an ‘alias’ with the command string, and then assign the alias to the key with the SetKey command.

Enhancements in v. 3.6b2 (Web-only release)

- Added new editor primitives: MoveSubWordLeft, MoveSubWordRight, SelectSubWordLeft, and SelectSubWordRight. A subword boundary is defined as:

  - a lower-case character followed by an upper-case character within a word

  - an alphabetic character next to a numeric character

  - a word break character (i.e. a character not in {wordset})

  - an embedded underscore character

  These editor primitives are initially mapped to Control-LeftArrow, Control-RightArrow, Control-Shift-LeftArrow, and Control-Shift-RightArrow.

- Added control-double-click behavior, to allow the selection of a subword with a word. (See above for definition of subword boundaries.)

Enhancements in v. 3.6a1 (Web-only release)

- Changed behavior of Editor. Now, a file of any type (or no type at all) can be dragged onto the Shell icon, and opened by the Editor. If the file is not a TEXT file, the editor will not write out any of the Editor state resources.

Enhancements in v. 3.6d7 (Aug. 2001 Dev CD)

- Added "Replace, then Find" item to Find menu, visible when the Option key is held down. It's an additional behavior on top of the "Replace Again" menu item. Key equivalent is Command-Option-T.

- Added logic to support proportional scroll bars on Mac OS systems that also have proportional scroll bar support. (Mac OS 8.5 and later.)

- Added ability to click on the "Deferred Save" icon in a window's info bar to toggle the SaveOnClose setting for that window.

Enhancements in v. 3.6d5 (Web-only release)

- A new menu item, "Set Find String", has been added to the Find menu, with a command-key equivalent of Command-E. This item will copy the current selection in the active window into the internal variable used to cache the string used for the last Find command. Option-Command-E will copy the current selection of the active window into the variable used to cache the string from the last Replace command. Using these menu commands allows the user to “pre-cache" these strings, so that they're available for the "Find Same" "Replace Same" menu commands.

Enhancements in v. 3.6d4 (Feb. 2001 Dev CD)

- When user items in Shell-defined menus are enabled/ disabled, the shell now performs a CountMenuItems() to determine exactly how many items are in the menu, and then limits its loop to that many items. (Previously, would always operate on items up to number 32.)

- Both NavServices and StdFile Open dialogs now recognize a new environment variable, {OpenSelectAllDocuments}, that, if set, will default the dialog to open with all documents selectable. The default behavior (based on the setting of the new environment variable) is reversed if the Option key is down.

- Added .xml, .plist suffixes to HTML colorizer resource.

- Added 'Tee' command to colorizer resource for MPW commands.

Enhancements in v. 3.6d3 (Web-only release)

- Added support to recognize a new environment variable - DefaultLineTerminator. This variable controls the ethnicity of a new file window created by the MPW editor. If the variable is set to "Unix", a linefeed will be inserted into the file when the user presses the return key at the end of a line. If the variable is set to "Dos", a carriage return/linefeed combination line terminator will be inserted into the file when the user presses the return key. If the variable is set to any other value, or is not set at all, the editor will behave as it has in the past, using a carriage return as the line terminator character.

Enhancements in v. 3.6d2 (Web-only release)

- When the user attempts to open a file for which they don't have write privileges (such as on Mac OS X), and the system returns a -5000 error, the shell will now attempt to open the file read-only automatically.

Enhancements in v. 3.6d1 (Aug. 2000 Dev CD)

- Implemented logic to support the kMustBeginLineFlag option of the syntax colorization, to better support Fortran syntax colorization.

Bug fixes in v. 3.6b2 (Web-only release)

- Fixed problem with auto indent behavior in DOS files. When a newline was inserted at the beginning of the line, the contents of the new line were not being indented properly. (Existing indentation was actually being removed.)

- Fixed crash that occurred when the PopUpFuncs menu was activated with the Shift key depressed.

Bug fixes in v. 3.6b1 (Web-only release)

- Changed file saving logic in Editor to use the Temporary Items folder to hold the temporary file (i.e. destination of the copyFile() operation), and then use FSpExchangeFiles() to swap the contents of the original file and the temporary file. This was done to be better behaved when running on Mac OS X. (Previously, saving a file when the file's parent directory is displayed in the X Finder would cause the file to "disappear" from the Finder's display, until some other modification was made to the directory.)

Bug fixes in v. 3.6a1 (Web-only release)

- Fixed problem with paste operation. When the entire contents of a window was selected with "Select All" (Cmd-A), and then a Paste (Cmd-V) operation was

performed, the editor re-used a stale pointer to a text block that had been deleted. Fixed by resetting the block pointers after the textual replacement operation.

- Added logic to adjust for differing window title bar heights depending on whether or not the Appearance Manager is available.

Bug fixes in v. 3.6d8 (Web-only release)

- Added/fixed Objective-C keywords in colorizer resource.
- Fixed problem with tracking the mouse position while the user moves proportional scroll bar thumbs while live scrolling is active.
- Fixed problem with calling FixMenuItem() for non-existent menu items while initializing the Mark menu.
- Fixed problem in window management code when a NULL window ref was returned by FindWindow(). Found by running with DebuggingCarbonLib.

Bug fixes in v. 3.6d7 (Aug. 2001 Dev CD)

- When an alias file is selected in the file list of the Browser window, the Browser will now resolve the alias to its target, and examine the target for any existing marks to display in the Mark list.

Bug fixes in v. 3.6d6 (Web-only release)

- Fixed problem introduced in 3.6d5 when a file was saved. The buffered information for the file's directory was being written to the directory entry for the file, rather than restoring the information for the directory itself, thereby obliterating the file type and creator information (along with the other catalog info).

Bug fixes in v. 3.6d5 (Web-only release)

- Fixed a problem in Undo where an Undo operation would undo too many steps when AllowMultipleUndo was turned on. Reported by Geza Fabry.

- If ® is included in a replace string, without a following digit, then the ® character will be treated as a literal character. Also added logic to recognize an escaped ® character, and treat it as a literal also.

- If ®<digit> is included in a replace string, and the tagged selection represented by that combination is undefined, the ®<digit> tag will be replaced by a null string.

Bug fixes in v. 3.6d2 (Web-only release)

- Changed regular expression parsing to be consistent with command line parsing in the interpretation of escaped character sequences. RE parsing now uses the same conversion function to translate escaped characters into the actual characters, so that sequences of the form '0Xdd' work.

- Changed regular expression scanner to recognize both carriage return and linefeed as line terminators.

- Fixed initialization to properly handle startup volumes > 4 Gb.

- Copy/Paste of a large block of text FROM a DOS file resulted in a loss of data. Problem was that, when the line terminators in the clipboard text blocks were converted from CR/LF to just CR, the internal state fields were not updated properly. Now they are.

Bug fixes in v. 3.6d1 (Aug. 2000 Dev CD)

- If the screen resolution was changed, the shell would continue to constrain window growth based on the size of the screen in effect when the shell was launched. The shell now installs a notification procedure so that the Display Manager will notify the shell when the screen size changes.

- Text dragging now translates line terminators in the dragged text to match those of the target window, if necessary.

- Fixed problem in Format dialog where the current language setting in the list of languages wasn't being set correctly if the language happened to be the last one in the list. (Reported by Graham Hinton, who's doing a lot of work with Z80 assembly language.)

- When a non-active window was modified somehow, and then brought forward to be the active window, the MPButtons support logic was changing the mod date back to it's value when the window was opened. First of all, reading resources out of the file

should never change the mod date, so there should be no reason to ever reset the mod date. I also modified the resource-changing functions so that the mod date of the affected file is only reset if the data in the file is not dirty (i.e. is unmodified). If the file is already dirty, then the mod date shouldn't be reset unless the file is reverted by the user.

- Changed Lex extension to ".l" from ".y" in the 'Odds' resource.

- Fixed potential problem with various popup menus using a menu id that's already in use by a user-defined menu. All menu identifiers are now defined in a single location, and are defined so that popup menu id's will always be greater than the maximum number of user-defined menus allowed.

- When a new file is created via the File menu, the file specified in the StdFile/NavSvc dialog is explicitly deleted before being created. This is done so that the new file doesn't inherit any old, stale, left-over resources from the pre-existing file, if any.

- Fixed problem in the low level window file i/o handler that was getting the editor confused. When a block of text was written to an editor window which contained different line terminator conventions than the source block, the window file mark was not being set properly. Previously, it was set based on the number of characters that were requested to be written, not the ACTUAL number of characters that were written (which would be different due to the addition/removal of the second DOS line terminator character).

- Added logic in the shell's GrowZoneProc to call WaitNextEvent periodically. This is most apparent to the user when the Bulldozer icon appears.

- Merged in changes to the Fortran syntax colorization resources suggested by Brian Helinski at Absoft Corporation.

- Added a missing '&' in a conditional statement which was causing a possible NULL dereference.

Shell

Enhancements in v. 3.6d7 (Aug. 2001 Dev CD)

- Extended behavior of AllowCmdsOnlyInWorksheet. When set to 2, it allows the Return key to be used like the Enter key while executing commands in the worksheet. Command-Return will allow the user to insert a carriage return instead.

(i.e. 'Set AllowCmdsOnlyInWorksheet 2' will switch the behavior of Return and Command-Return when used in the Worksheet window.) In addition, the Enter key will behave like the former Return key, inserting a carriage return in the text.

- Enhanced application launching so that the process name of an already-running application can be specified as the command. (The process name can be determined by examining the Application/Process menu at the right end of the menu bar.)

Enhancements in v. 3.6d3 (Web-only release)

- Added support for pathnames with >255 characters.

Enhancements in v. 3.6d2 (Web-only release)

- Changed application launching to allow launching and quitting of application extensions (file type 'appe').

Enhancements in v. 3.6d1 (Aug. 2000 Dev CD)

- Opening an MPW tool from the Finder now causes the shell to display the Commando dialog for that tool.
- Dragging a folder to the application icon will now set the current working directory to that folder.
- Added 'r' to list of characters that will be recognized as special characters on the command line. It will be converted to a linefeed character (0x0a) (Currently, the only other special characters are 'n', 't', and 'f'.)
- Added logic in application launching to be able to handle application "packages" properly.
- Built with the new 3.6a4 ToolLibs, so the updated ErrMgr is available to interpret the contents of the updated SysErrs.err file.

Bug fixes in v. 3.6b2 (Web-only release)

- Fixed problem introduced in 3.6b1. When quitting, if a window other than the Worksheet was active, the MPButtons Commenter module was using a stale pointer, and somehow creating a bogus resource file in the same directory as the file in the active window. Fixed by ensuring that said pointer is pointing to a valid data structure when MPButtons is called.

- Altered creation of MPW.Scratch file so that if it can't be created in the MPW folder for whatever reason (like launching MPW from a read-only volume), the file is created in the system's Temporary Items folder.

Bug fixes in v. 3.6b1 (Web-only release)

- Reverted behavior of continuation lines in scripts back to what it was prior to version 3.6d7. At the time, I thought it was fine to replace the escaped line terminator with a space, and then delete any leading whitespace on the following line. After encountering several problems as a result of this scheme, I now realize that it probably wasn't a good idea in the first place. The behavior now is that an escaped line terminator will simply be removed from the input stream as if it didn't exist at all.

- When 'Set Echo 2' is active (for script debugging purposes), the shell will now quote the script pathname, if necessary.

Bug fixes in v. 3.6a1 (Web-only release)

- Fixed problem (again) with parsing of a command on multiple lines, with leading whitespace on the continued portions of the command.

Bug fixes in v. 3.6d8 (Web-only release)

- Fixed problem introduced in 3.6d7 in which continuation lines in a multi-line command would not have leading whitespace stripped from the beginning of the continued portion of the line.

Bug fixes in v. 3.6d7 (Aug. 2001 Dev CD)

- Changed {"Parameters"} quoting so that special characters within the parameter strings are quoted appropriately, in addition to the enclosing double quotes. Tabs, Returns, Formfeeds, and Linefeeds are escaped properly. '∂', '"', '{', and '`' characters are also preceded by the Escape character ('∂').

- Fixed problems in quoting the contents of expanded variables if those variables contained escape characters ('∂'). The new logic will also quote tabs, returns, formfeeds, linefeeds, '"', '{', and '`' characters.

- Fixed the same problem with expanding the results of an embedded command.

- An Option-click in the DoIt box (aka the Status panel) of a window now generates the same behavior as Option-Enter - Commando will be invoked for the command in the current selection.

- Fixed application launching logic to allow folders to be specified as application parameters sent to the launched application.

- Fixed script processing to properly recognize lines when said lines are terminated with linefeed characters.

Bug fixes in v. 3.6d5 (Web-only release)

- The shell no longer gets itself into an infinite loop when it encounters the special "HFS+ Private Data" folder on a volume touched by Mac OS X.

- A command alias followed immediately by a comment (with no intervening space character) now works as expected.

Bug fixes in v. 3.6d4 (Feb. 2001 Dev CD)

- Fixed the Commando dialog for the Open command to include the two new Finder-related options. (Thanks to Dan Allen for pointing out the omission.)

Bug fixes in v. 3.6d3 (Web-only release)

- Fixed problem in Files command in which the 'o' and/or 'g' parameters to the -x option would return no information about folders on file servers.

Bug fixes in v. 3.6d2 (Web-only release)

- Fixed problem with receiving an 'odoc' on a directory before the shell has completed its startup process. Previously, the current directory would be set to the parent directory of the directory that was specified in the 'odoc', rather than the directory itself.
- Fixed problem with receiving an 'odoc' event on a tool before the shell has completed it's startup process.
- Fixed problem with receiving an 'odoc' event on a tool that existed on a volume with a space in the pathname. The pathname is now quoted, if necessary, before being inserted into a command line.

Bug fixes in v. 3.6d1 (Aug. 2000 Dev CD)

- Fixed application launching logic to report the name of the file that it was unable to retrieve information on, instead of reporting the filename as "".
- Fixed possible NULL dereference during shell initialization.

Built-in Commands

Changes in MPW Shell 3.6b2 (Web-only release)

Files

- Changed output format of the Size field of a Files listing. The size value is now scaled to K/M/G as appropriate for the file/folder size, with the appropriate suffix. (This is similar to what the Finder has been doing for years.)

ResolveAlias

- Fixed bug in ResolveAlias which would produce an invalid name when an alias resolved to a volume name.
- Removed unimplemented -v option from ResolveAlias command.

Changes in MPW Shell 3.6b1 (Web-only release)

Rshell/RProj

- Changed behavior of RShell and RProj commands. If the target application of the command must be launched automatically as a result of the command, the command will first attempt to launch a copy of the application that exists in the same directory as the client application. If there is none, then a copy of the application that exists on the same volume as the client will be searched for and launched if found. If none, the command will search all volumes for the proper target application.

Changes in MPW Shell 3.6d8 (Aug. 2001 Dev CD)

Duplicate

- Corrected message in dialog that is presented when the Duplicate command is used to copy a source file data fork to a destination file resource fork, or vice-versa. When

the -swap option was used, the dialog identified the incorrect destination fork, by basing its decision on whether the -r or -d option was used (which specifies the source fork).

OSACompile

- Added OSACompile built-in command, to match the behavior supplied by the same-named BSD executable supplied with Mac OS X.

Syntax

```
OSACompile [-l language] [-e command] [-o name] [-d] [-r type:id]
[-t type] [-c creator] [-x] [file ...]
```

Description

OSACompile compiles the given files, or standard input if none are listed, into a single output script. Files may be plain text or other compiled scripts.

If no options are specified, OSACompile produces a classic Mac OS format script file, that is, type “osas”(compiled script), creator “ToyS” (Script Editor), with the script data in the scpt:128 resource and nothing in the data fork. This format is compatible with all Mac OS and Mac OS X systems.

The -d and -r options are not exclusive. If exactly one is specified, the script is written only to that fork. If both are specified, the script is written to both forks.

Output

OSACompile writes the compiled script to a file named “a.scpt”. This may be overridden with the -o option.

Diagnostics

Errors and warnings are always written to the diagnostic output.

Status

0 Normal termination.
1 Parameter or option error.
2 Command or system error.

Parameters

[*file*]…

Specifies one or more input script files.

Options

-l language

Override the language for any plain text files. Normally, plain text files are compiled as AppleScript.

-e command

Enter one line of a script. Script commands given via -e are prepended to the normal source, if any. Multiple -e commands may be given to build up a multi-line script. Because most scripts use characters that are special to many shell programs (e.g., AppleScript uses single and double quote marks, “(“, “)”, and “*”), the command will have to be correctly quoted and escaped to get it past the shell intact.

-o name

Place the output in the file name. If -o is not specified, the resulting script is placed in the file “a.scpt”.

-d

Place the resulting script in the data fork of the output file.

-r type:id

Place the resulting script in the resource fork of the output file, in the specified resource.

-t type

Set the output file type to `type`. Type is a four-character code. If this option is omitted and the output file does not exist, the type is set to “osas”, that is, a compiled script.

-c creator

Set the output file creator to `creator`. Creator is a four-character code. If this option is omitted and the output file does not exist, the creator is set to “ToyS”, that is, Script Editor.

-x

Save the resulting script as execute only.

OSALang

- Added OSALang built-in commands, to match the behavior supplied by the same-named BSD executable supplied with Mac OS X.

Syntax

```
OSALang [-d] [-l] [-i]
```

Description

OSALang prints information about installed OSA languages. With no options, it prints an unadorned list of language names to standard output. These names can be passed to the -l options of `OSACompile` and `OSAScript`.

Output

OSALang writes the output to standard output.

Diagnostics

Errors and warnings are always written to the diagnostic output.

Status

0 Normal termination.
1 Parameter or option error.

Options

-d

Only print the default language.

-l

List in long format. For each language, osalang will print its component subtype, manufacturer, and capability flags. There are eight groups of

optional routines that scripting components can support. Each flag is either a letter, meaning the group is supported, or '-', meaning it is not. The letters map to the following groups:

c compiling scripts.

g getting source data.

x coercing script values.

e manipulating the event create and send functions.

r recording scripts.

v "convenience" APIs to execute scripts in one step.

d manipulating dialects.

h using scripts to handle Apple Events.

For descriptions of the groups and the APIs in each of them, see <http://developer.apple.com/techpubs/mac/IAC/IAC-361.html>.

-i

Same as -l, but also prints the description of each component after its name.

OSAScript

- Added OSAScript built-in command, to match the behavior supplied by the same-named BSD executable supplied with Mac OS X.

Syntax

```
OSAScript [-l language] [-e command] [-s flags] [programfile]
```

Description

OSAScript executes the given script file, or standard input if none is given. Scripts may be plain text or compiled scripts. osascript was designed for use with AppleScript, but will work with any Open Scripting Architecture (OSA) language. To get a list of the OSA languages installed on your system, use OSALang. For documentation on AppleScript itself, see <http://www.apple.com/applescript>.

Output

OSAScript writes the output to standard output.

Diagnostics

Errors and warnings are always written to the diagnostic output.

Status

0 Normal termination.
1 Parameter or option error.
2 Command or system error.

Parameters

[*programfile*]

Specifies the OSA script file to execute. May be either plain text source, or a compiled script file.

Options

-e command

Enter one line of a script. If -e is given, osascript will not look for a filename in the argument list. Multiple -e commands may be given to build up a multi-line script. Because most scripts use characters that are special to many shell programs (e.g., AppleScript uses single and double quote marks, “(“, “)”, and “*”), the command will have to be correctly quoted and escaped to get it past the shell intact.

-l language

Override the language for any plain text files. Normally, plain text files are compiled as AppleScript.

-s flags

Modify the output style. The flags argument is a string consisting of any of the modifier characters e, h, o, and s. Multiple modifiers can be concatenated in the same string, and multiple -s options can be specified. The modifiers come in exclusive pairs; if conflicting modifiers are specified, the last one takes precedence. The meanings of the modifier characters are as follows:

h Print values in human-readable form (default).

s Print values in recompilable source form.

OSAScript normally prints its results in human-readable form: strings do not have quotes around them, characters are not escaped, braces for lists and records are omitted, etc. This is generally more useful, but can introduce ambiguities. For example, the lists ‘{"foo", "bar"}’ and ‘{{"foo", {"bar"}}}’ would both be displayed as ‘foo, bar’. To see the results in an unambiguous form that could be recompiled into the same value, use the ‘s’ modifier.

e Print script errors to stderr (default).

o Print script errors to stdout.

OSAScript normally prints script errors to stderr, so downstream clients only see valid results. When running automated tests, however, using the ‘o’ modifier lets you distinguish script errors, which you care about matching, from other diagnostic output, which you don't.

Set

- Fixed crash in the Set command caused by examining the first command line parameters when no parameters were supplied on the command line.

Changes in MPW Shell 3.6d7 (Aug. 2001 Dev CD)

Delete

- Added -l option to the Delete command, to automatically unlock files and directories before they're deleted. ()
- Removed -ay, -an, and -ac options from the Delete command. While the command will still accept the options without complaint (for compatibility), they no longer have any effect. The Delete command no longer treats alias files any differently than any other file.

Duplicate

- Fixed crash in Duplicate that occurred when a server volume unexpectedly disappeared, and an error message that attempted to report the problem passed a NULL string pointer to strlen().

Help

- Enhanced Help command to look for the requested help text in multiple locations. If the -f option is used (with or without additional parameters), only the specified file will be searched. If no parameters are supplied at all, the command will display the default portion of the default MPW.Help file, located in the {ShellDirectory}.

  In the normal case of one or more parameters on the command line (that being the command for which to retrieve the help text), the Help command will first attempt to locate a tool or script by that name, as if the tool or script is to be executed (i.e. using the normal {Commands} search algorithm). If found, a 'MPSR'(1024) resource is retrieved from the file. If the resource is successfully loaded, the contents of the resource are written to standard output. This resource can be most easily created by having the help text stored in a standard text file, and then using Rez to read the text file into the resource as the tool/script is being built:

  echo "Read 'MPSR'(1024) ∂"myTool.help∂";" | Rez -a -o myTool

  Using this storage method, the help text for a given tool/script can be easily synchronized to the tool/script itself.

  If the tool/script cannot be found, or does not contain a 'Help' resource, the Help command will look for a file with the same name as the parameter in a new ":Help Items:" folder, stored in the same folder as the MPW Shell. (i.e. the {ShellDirectory}) If the file is found, the contents of the file are written to standard output.

  If no dedicated help file for a command is found in the :Help Items: folder, the Help command will perform its standard search through the default MPW.Help file, located in the {ShellDirectory}.

  If the help entry is not found in MPW.Help, the help command will then enumerate any files in the :Help Items: directory with names ending in '.help', and will search those files.

Open

- When the -s or -f options are used with the Open command, the shell will now switch the Finder to be the active process.

Quote

- Fixed Quote command (and string quoting in general) to eliminate extraneous hard quote characters when possible.

Set

- Fixed command to treat the -e option as a true option. Code review and experimentation demonstrated that if the -e option was used with only a variable name (without an associated value), the command would define a variable with the name '-e', set to the value of the supplied variable name.
- Fixed quoting in the output of the Set command when the value string is a regular expression.

Unset

- Added -e option to the Unset command. This is intended to mirror the -e option in the Set command. When used, any variables unset will also be unexported. Note that this only operates at the current variable scope. Variables exported from previous scopes will remain exported.

Changes in MPW Shell 3.6d6 (Web-only release)

Evaluate

- Added -l option. This option is used to have the value displayed as a 4-character literal (i.e. an OSType).

Move

- Fixed problem introduced in 3.6d4. An attempt to move a file into a directory containing a busy file having the same filename would end up with the original file moved, but renamed to 'MPW.UniqueX' (with X being some digit), rather that having the move operation fail.

Version

- Added -s option, to report the version number of the system.

Changes in MPW Shell 3.6d5 (Web-only release)

CheckOut

- Fixed crash that occurred when the -cancel option was used with the Checkout command (without a corresponding -y/-n/-c).

Duplicate

- A `-swap` option has been added to the Duplicate command. This option is only valid when used in conjunction with the `-d` or `-r` options. Its effect is to cause the duplication of the specified fork of the source file (data or resource) into the opposite fork of the destination file. For example:

  ```
  Duplicate -r -swap sourcefile destinationfile
  ```

  will copy the resource fork of `sourcefile` into the data fork of `destinationfile`. It is intended to be used in producing Mac OS X packages.

Equal

- The Equal command now reports offsets of mismatched bytes from a 0-based offset in the file. (Previously, the command would report the first byte of the file as byte 1, which doesn't match the C convention, nor the convention used by the DumpFile command.)

Changes in MPW Shell 3.6d4 (Feb. 2001 Dev CD)

Duplicate

- Fixed possible infinite loop in the Duplicate command when duplicating just the data fork of a file into a file open in a window in the editor on Mac OS 9.0 and later.

- Fixed Duplicate's inability to duplicate a file to a target file that is opened in a window in the editor.

RunApplescript

- The -s option has been removed, and the Open Scripting Architecture (OSA) is queried to determine if the compiled script should be saved. This will allow scripts to have persistent properties.

- A new -sub option has been added, to allow the user to specify the name of an Applescript subroutine to invoke, and to specify the parameters to be passed to the subroutine. Syntax is:

  -sub 'name(parm1, ..., parmN)'

  The named subroutine will be called, with the positional parameters supplied to the Applescript in the order supplied on the command line. The parameters must be supplied within parentheses, and separated by commas. Whitespace is optional around any of the required characters. Parameters may be quoted with either single or double quotes (but said quotes must match). There is no evaluation performed on any of the supplied parameters. Whatever is supplied as a parameter will be passed along to the Applescript - no character substitution, escape sequences, etc. are recognized or acted upon.

- Any result returned by the Applescript will be extracted and written to standard output.

- Custom ActiveProc and SendProc procedures are supplied to the OSA, so that update/activate events are handled properly, and to allow the user to cancel lengthy Applescripts.

TransferCkid

- Fixed TransferCkid to delete any existing 'ckid' resource from the target file prior to copying the new 'ckid' resource from the source file. This will eliminate the problem of having duplicate 'ckid' resources in a file.

Changes in MPW Shell 3.6d2 (Web-only release)

OrphanFiles

- Enabled OrphanFiles as a built-in command in MPW Shell.

RunApplescript

- Added RunApplescript Commando dialog.
- Added RunApplescript to list of colorized shell keywords.

TransferCkid

- Enabled TransferCkid as a built-in command in MPW Shell.

Changes in MPW Shell 3.6d1 (Aug. 2000 Dev CD)

Files

- Fixed problem in Files command with retrieving Finder comments for a file from the Desktop Database.
- Fixed crash in Files command that was caused when the command attempted to display a full pathname containing more than 256 characters. The problem was a statically-sized buffer that would allow the code to corrupt the stack when a long pathname was built up. Fixed by bumping the size of the buffer to 512 bytes, and to check the length of the pathname string so that a properly sized buffer can be dynamically allocated if necessary.

RunApplescript

- Added RunApplescript command. Syntax is:

```
RunApplescript [-c] [-d | -i] [-s] [-p] (file | <script)
        -c      # syntax check only, don't run the script
        -d      # delete any existing compiled script resource in the file
        -i      # ignore any compiled script resource in the file
        -s      # save the compiled script in the input file
        -p      # show progress of execution
```

Unmount

- Fixed the Unmount command to call the File Manager's Eject() function if the driver is marked as having ejectable media.