

Raining Data Corporation

RELEASE NOTES

January 28, 2003

mvEnterprise Release 4.2.0 on AIX

© 1998–2003 Raining Data Corporation



This document refers to products that might yet be announced by Product Marketing or be generally available. Please consult your Raining Data Representative for more information.

Corrections, Restrictions and Enhancements

The following sections include corrected action items, restrictions, and enhancements for this release. Where appropriate, associated action item numbers are noted within parentheses.

In mvEnterprise 4.2.0

Corrections

Basic Corrections

- Corrected a problem with DELETESEQ which was leaving record locks after the BASIC program has ended. (27034)
- Corrected a problem with the BASIC SORT (), which caused intermittent workspace runaways. (27733)

Debugger Corrections

- Corrected a BASIC debugger abort when displaying the value of a BASIC variable whose offset was greater than 32k. This problem would also occur with regular common variables. (27542)

Index Corrections

- Corrected an abort when creating an index whose attribute contained more than 4095 multivalues. (26370)

Kernal Corrections

- Corrected the default permissions on UNIX file creation from 0660 to 0666. (27800)

Save/Restore Corrections

- Corrected message when using VERIFY-SAVE (A). (26298)
- Corrected problem with restores intermittently aborting or skipping files during a restore. This occurred when the file number on tape exceeded 65,535 and rolled over to zero. The error affected the ability to restore any file whose file number on tape was zero. (26031)

NOTE: This problem has affected the ability to perform a selective restore by specifying a file number greater than 65,535.

System Corrections

- Corrected the display on the `WHAT` command when using on a large system, which could cause the `maxfid` and available frame numbers to overlap. (26460)

TCL Corrections

- Corrected an intermittent problem with `COMI` not creating items in the `TERMIO` file. (27736)

Enhancements

Documentation Enhancements

- `mvEnterprise 4.2.0` documentation, including the Installation and Configuration Guide, Reference Manual, and Release Notes are now all available on the `mvEnterprise 4.2.0` CD-ROM.

Installation Enhancements

- Enhanced `mvEnterprise 4.2.0` by distributing it on CD-ROM. This requires a new installation procedure. Please refer to the Installation Guide for more information. (27427)

TCL Enhancements

- Enhanced by adding an `I` option to the `POKE` command for suppressing carriage return and linefeed when poking characters to a specified line. (27809)

Kernal Enhancements

- Enhanced by allowing an `mvEnterprise` process to remain after it has lost connection with its device. (27060).
- Enhanced the ability for `mvEnterprise` processes to respond to signals (such as a `LOGOFF`) after it has lost connection (turned off or otherwise disconnected) to its device. (27313) (24932)

In mvEnterprise 4.1.0

Corrections

Access Corrections

- Corrected an intermittent problem with F summation correlatives aborting during execution.

Basic Corrections

- Corrected sequential file handling when multiple sequential files are being used in a program. Typically, when multiple sequential files are opened and being updated, the program, for example, closes the first sequential file and opens a new sequential file and continues to perform updates and closes the remaining opened files, a garbage collect and subsequent workspace aborts were reported after the basic program stopped running.

Debugger Corrections

- Corrected a problem in the system debugger so that the G command no longer jumps to an incorrect address when a mode name and offset is specified.

ODA Corrections

- Corrected a problem that caused Pick/BASIC programs to abort when exiting if file indexes were used.
- Corrected a problem that caused Pick/BASIC programs to abort when using the SELECT statement with a remote file index variable.
- Corrected ODA Server aborts when clearing a remote file that had indexes defined.
- Corrected problems with update protection locks not working for remote files. ODA Servers will use the remote account's update and retrieval locks for checking permissions.
- Corrected aborts when attempting to open alternate data levels of remote files.

Enhancements

Kernel Enhancements

- Enhanced pick.lpd and pick.lpi to use local function code instead of using standard AIX libraries. This removed the 22-character path limit introduced in 4.0.9.

Virtual Tape Enhancements

- mvEnterprise tape handling capabilities have been enhanced by adding the ability to direct tape input and output to and from a virtual tape file. Previously, tape input and output could only reference a physical tape device or an ethertape connection.
 - Virtual tape files allow tape input/output to be sent to or received from:
 - a UNIX-level file.
 - a compressed UNIX-level file. (The administrator can change the compression program used in this operation.)
 - user-defined filter program(s) for special processing.
 - Virtual tape output files can be copied, moved, compressed, FTP'd, encrypted, archived, emailed, or burned to CD-ROM, just like any native UNIX-level file. This eliminates the need to send physical tapes from place to place to transfer mvEnterprise virtual data.
 - The mvEnterprise virtual tape capability supports all standard input and output processes such as filesaves and restores, T-DUMPS, transaction logs, and Pick/BASIC WRITET/READT statements.

Note: All currently valid T-ATT and tape device command formats remain legal and function as they did in previous releases.

Please see the attached section on Virtual Tape Files for more information and instructions on this feature.

In mvEnterprise 4.0.11

Corrections

Basic Corrections

- Corrected a line-hanging problem caused by stack corruption when using `OPENSER` on a UNIX file in AIX 4.2.1. If the UNIX file name and path excluded 22 characters, then stack corruption occurred. This problem was introduced in mvEnterprise 4.0.9 because standard AIX ‘`ttyunlock`’ was mistakenly used on non-tty devices.

Kernel Corrections

- Corrected `pick.lpd` and `pick.lpi` to allow longer printer names and resolve a random pipe-name error. The limit for a printer name is 64 characters including the path, but it is recommended to keep printer names short.
- Corrected a problem with handling the `ENODEV` condition for reading on a powered off Digi Etherlite.
- Corrected a problem with `serial_cnt` reading if no room was in the buffer, and the `serial_cnt` read going into the wrong buffer index position. This caused a problem with repeated calls to `GET.INP.PTRS` and `SYSTEM(14)` when input was being received, but there was no read operation between successive calls. This also resolved a problem with some Infolinke data transfers.
- Corrected a problem with timing-out the close function when a device appears to have hung. This ensures that non-line terminal writes (e.g. `OPENSER` devices) are performed synchronously and paced, so that output operations are made in the correct order.
- Corrected the hex display of kernel message 117 used in serial I/O (`termopen` error in `read/flush`) so that we know the value is hex.

mve_tel Corrections

- Corrected a problem with pty 0 byte reads disconnecting when using them with AIX 4.3.3.10.
- Corrected a problem with `mve_tel7024.script_sample` and `mve_tel7024.ports_sample`. These now allow connections based on remote IP address octets so that group connects are available, and give an option to supply the `NOCONNECT` string in the line option field, as a way to deny connections.

Spooler Corrections

- Corrected a problem with the printer not printing after being reassigned to a different queue. This happened because of the

mismatch pause and wake implementation for spooler introduced in a previous release.

Enhancements

Kernel Enhancements

- Enhanced picksync to default system wide sync time to 03:00 if entered incorrectly.
- Enhanced the log file or the display messages for picksync to be more descriptive.

System Enhancements

- Enhanced MD-UPGRADE to update MD items, which are reserved for the SYSPROG account, on other accounts if they are there.

CAUTION: If the account to be updated by MD-UPGRADE contains user specific MD items with the same name as the standard MD items, they will be over written.

In mvEnterprise 4.0.10

Corrections

Basic Corrections

- Corrected a problem introduced in 4.0.9 that could cause an open failure when using OPENSER to open a file object was fixed.

Logger Corrections

- Corrected a problem introduced in 4.0.9 and related to the OPENSER above that caused the TL.DISP command to fail as fixed.
- The logger sweeper and consumer components were modified to use shared memory segments rather than mapped file segments. This change totally eliminates the need for the `./tl_shared_mem` and `/tmp/tl_consumer...` files and closes a possible hole that might cause AIX crashes.

In mvEnterprise 4.0.9

Corrections

Access Corrections

- A problem that caused an overflow runaway has been fixed. This problem required more than 300 default numeric item-ids in the file dictionary and a LIST or SORT command with default output.
- A problem where a T-correlative after a B-correlative that opened a file dictionary failed was fixed.

Basic Corrections

- Exclusive locks were added to the OPENSER command so that only one process at a time can attempt to control a serial device. If a program attempts to open a busy device, it will take the ELSE clause. This is implemented with the UNIX standard 'ttylock' commands that write the lock records in the /etc/locks directory.
- A problem where DTR never dropped when using OPENSER and CLOSE was fixed. Now when the last user of a terminal device closes it the system will drop DTR.
- A problem where a READV that had an unassigned variable in the attribute number slot could leave a group lock locked for the duration of the program was fixed.
- Pick/BASIC function SUM(expression) was not returning a 0 when the expression is NULL (SPR 3563). It now returns 0.
- The Basic User Exit 'U61FC', used to get the CPU time of another line was corrected. This was broken in all previous 4.0 releases.
- The GET and GETX commands were changed to be more compatible with the mvBASE/Mentor products. GET may now echo character reads and honors the ECHO state of the line if and only if the form OPENSER `` TO . . . was used to open the local terminal. GET and GETX do *not* echo when the OPENSER uses an explicit device.

NOTE – This change may require adjustments in any application code that uses this function such as deleting SEND commands that do explicit echoing or setting the ECHO OFF state.

- The function SYSTEM(105) was added to provide easy access to a range of UNIX environment variables. Please see the attached manual page for this addition.
- An abort when using the MP (packed decimal) conversions was fixed in Release 4.0.8 but not documented there.

Kernel Corrections

- pick.lpd and pick.lpi now use standard AIX ‘ttylock’ and ‘ttyunlock’ to lock and unlock the pipe. The pipe name and its path name inclusively can’t be more than 22 characters.

System Corrections

- A problem with the CLEAR-PHANTOM command not deleting port record(s) for phantom(s) in ACC file was fixed. This caused LISTU to show them as being still logged on (SPR 4560).
- A problem with the single address cursor output in the VT-100 terminal type was fixed.
- The name for UNIX print driver log files has been changed. The old file had a name composed of lpd_log.[processid]. This caused numerous ‘dead’ log files and made it difficult to tie the log file back to a print queue. The new name is log_[printerpipename] where the printer pipe name by convention includes the machine name, the Unix print queue and the mvEnterprise line number that fully qualifies the log file. The location for the log file still defaults to /tmp, but can be moved by placing a -p path parameter on the pick.lpd command line.
- The name for the Consumer/Consumer Helper shared memory segment has been changed. The old file had a name composed of tlconsumer[processid]. This could cause numerous ‘dead’ consumer shared memory segments in the /tmp file. The new name is [machinename]_tlc[consumernumber]. The location for the shared memory file still defaults to /tmp, but can be moved by placing a -p path parameter on the consumer command line.

In mvEnterprise 4.0.8

Corrections

Access Corrections

- Using an F summation (F; . . .;S) correlative inside the THEN/ELSE of an A ; IF /THEN /ELSE yielded unpredictable results, usually aborts or infinite loops. This problem has existed since release 2.0. Note that the S(n) form of summation documented with the A correlative worked properly.
- An obscure problem with SAMPLE n where n also happens to be an attribute definition with an X on line 1 was fixed. NOTE – this fix returns behavior regarding X attribute definitions to its previous state – X is *only* allowed in the default numeric output lists (1, 2, 3,

...) to suppress an attribute without terminating the list. It is an error to explicitly specify an output attribute with an X on line 1.

Kernel Corrections

- A problem where the type-ahead cancel command canceled both pending input and pending output was fixed. This problem exhibited as a loss of output, particularly when in the system or Basic debugger, or during login. The problem was inadvertently introduced in the 4.0.5d release.
- A problem that could incorrectly convert a number containing two decimal points was resolved. This bug has always existed in the AIX releases.

Newac Corrections

- Deleted a bad record in newac that overwrote the logon proc for the ODA.ADMIN account. If you already have this problem, replace the ODA.ADMIN MD entry named ODA.ADMIN with the following:
PQ
HMENU ODA.MAIN
PX

Scribe Corrections

- The scribe modes were re-assembled for this release fixing an abort when attempting to run any Scribe commands including HELP.

In mvEnterprise 4.0.6

Corrections

Access Corrections

- Fixed a problem with multiple break-on values and a break-field in the header sometimes displaying the wrong break-field.
- Fixed problems with having the words LIST or SORT in the dictionary of a file causing Access to drop the first display field.
- Fixed a problem with having a data file named like a connective causing the Access compiler to fail. (IE file TEST,NE)
- Fixed a problem with display attributes with an X on line 1 stopping the list output instead of continuing to the next attribute.
- Fixed a problem with incorrect rounding when there is insufficient fractional data in the value to be rounded (IE value of '5' output under a 'MR02' mask could display '1' instead of '0'). Note – this error is not likely in Basic, but could happen if forcing implied decimals. This error has existed since Release 2.0 (SPR 4561).

Basic Corrections

- Fixed a problem where EXECUTE would turn off basic history.
- Fix a problem that didn't allow relative addressing in the OPENSEQ statement (IE ./scripts).
- Fixed a problem where reusing UNIX file handles that had previous errors for OPENSEQ files could open a file that cannot be written to causing the generation of 0 length output files.
- Fixed a problem of incorrect output when SUM() fields overflowed the binary accumulator – the instruction now correctly shifts to string math.
- Provide more descriptive compile-time error message when a GOTO target label is unreachable because of the size of the object code.

ODA Corrections

- Fixed a problem with REMX not working correctly and displaying garbage output when the “accept timeout” on the server was set to 0. Problem has existed since 3.0.
- Fixed an abort when a Clearfile was issued to a remote file that had records cached by the REMOTE_CACHE command. Problem has existed since 3.0.
- Fixed ODA headings and help files to indicate that Host ID numbers are mandatory and must be consistent across all client/server pairs.
- Fixed a problem where a rapid bounce (shutdown and restart) of ODA servers in less than 2 minutes would fail due to the previous socket not yet clearing. Problem existed since 3.0 on UnixWare systems.
- Replaced missing UNIX and Spooler hosts in the Hosts file. These host records are used to access UNIX files or spooler hold entries from Q-pointers.
- Fixed erroneous display in NETPS if old “stat:...” records existed in the dictionary of the HOSTS file.

Spooler Corrections

- Re-corrected spoolers failing to start correctly.
- Fixed a race problem that could prevent the printer driver from seeing a closed print file thus delaying printing of the output spool file.
- Fixed a problem with the SPOOLQ ODA functions that could cause an FLZ in RP.PEQS.RDHDR. This fix will permanently suppress the LINKS field, which never functioned as documented.
- Fixed a problem with extraneous output or forward link zero aborts when capturing the output from the forms processor or new Procs.

TCL Corrections

- Fixed problem with MESSAGE command ignoring breaks that occur during the message send time.

- Fixed several problems with BLOCK-PRINT and messages that contain quoted strings.
- Fixed POVf display of leading zeros on block fids.

In mvEnterprise 4.0.5d

Corrections

Basic Corrections

- Corrected a problem with Basic generating a 'Bad stack descriptor' when a pop string of a long string crosses a stack frame boundary. This problem has existed since 1.8 but is easier to generate on 3.1 and above due to the larger stack structures.

Kernel Corrections

- Corrected a problem with EXECing an alternate abs throwing a console error message or dropping the process to UNIX.
- Corrected several ODA problems - see the ODA section. These problems have existed since release 3.0.

ODA Corrections

- Corrected a problem with starting ODA servers when phantoms in the server target range have not been shutdown properly.
- Corrected a problem with not being able to quickly and cleanly shutdown ODA servers.
- Corrected a problem where a temporary I/O error could hang an ODA server and leave the service unavailable.

Spooler Corrections

- Corrected a problem with STARTSPOOLER declaring it failed to start the printer process when the spooler actually started. This was caused by not waiting long enough to before making the determination.

Enhancements

Redback Support

- Added library file SYS.CATALOG and made other Basic adjustments to enable and support the RedBack web-enabling product.

SYSTEM(105) Documentation

This Basic function call returns a multivalued list of useful UNIX environment variables. The contents of the returned variable are as follows:

Attribute	Contains
1	Machine node name
2	Current Process ID
3	Parent Process ID
4	Numeric UNIX User ID
5	Numeric UNIX Group ID
6	Multivalued list of alternate numeric Group ID's
7	Multivalued list of this commands arguments
8	Local console path
9	Current working directory
10	Multivalued list of all local UNIX environment variables in the form VARIABLE=VALUE

This function is somewhat expensive and required values should be gotten once and saved for future use. Note that all values are static during the execution of a mvEnterprise process!

There is a limit of 3999 characters returned. If the results are longer than this, they will be truncated.

Virtual Tape Files

The new mvEnterprise virtual tape feature allows users to direct input and output to a user-created file on a disk drive, instead of a physical tape or ethertape device. This new command format allows specification of the input/output disk file, filtering commands, and other elements of the new virtual tape interface. Virtual tape files can be fully navigated with the T-FWD/T-BCK/T-FSF/T-BSF commands.

Extended T-ATT Form for Virtual Tape Files (Disk Files and Pipes):

Format

```
T-ATT unit_number | device=unit_number      type=  
tape/ethertape/file/compressed/filtered  
[ blocksize=block_size ]  
[ path=file_path_name ]  
[ maxsize=maximum_file_size[K/M/G] ]  
[ infilter=input_filter_command ]  
[ outfilter=output_filter_command ]  
[ options=option_string ]
```

Parameter(s)

device=*unit_number*

Required. Specifies the logical tape unit to assign, given in decimal.

If given as the first parameter in the T-ATT command, may alternatively be specified as *unit_number* alone, without the preceding **device**= keyword.

type=*tape/ethertape/file/compressed/filtered*

Required. Specifies the type of device to attach.

If **type=tape** or **type=ethertape**: specifies standard tape device or ethertape I/O. The destination device name or IP address is taken from the *config.tape* or *config.ethernet* files. All command parameters other than **device**=,

blocksize=, and **options=** are ignored. If **type=ethertape**, you must specify in the **options=** parameter whether the device is the sender (S) or the receiver (R).

If **type=file**:

selects virtual tape input and output from/to a UNIX-level file, as specified by the file name/path in the required **path=** parameter.

If **type=compressed**:

the virtual tape file is filtered (piped) through the scripts *vtape_compress* and *vtape_decompress* for output and input compression/decompression, respectively. These scripts, normally contained in the mvEnterprise production directory, can use UNIX programs such as compress/uncompress and gzip/gunzip to perform compression and decompression. However, they may be freely altered to suit any site-specific needs. The source/destination file name supplied to the compression script will be the one specified by the **path=** parameter. However, depending on the command and options used, you may need to include the appropriate extension in the file name.

If **type=filtered**:

the virtual tape file is filtered (piped) through a user-specified UNIX-level script or command, as specified by the required **outfilter=** and **infilter=** parameters. The source/destination file name supplied to the filter command/script will be the one specified by the required **path=** parameter.

blocksize=*blocksize*

Optional. Tape blocksize.

If a blocksize is not specified, it defaults as follows:

- **type= tape** or **ethertape** defaults to the last attached blocksize
- **type= file, compressed, or filtered** default 16,384 bytes.

NOTE—File restores require a tape block size of 16,384 bytes.

options=*option_string*

Optional. Additional alphabetic or numeric options to be passed to the virtual tape driver. The meaning of these depends on the specific **type=** specifier being used. Currently, only options R (receiver) or S (sender) are recognized; one of these is required if **type=ethertape**.

The command parameters below are relevant only for tape types **file**, **compressed**, and **filtered**:

path=*file_path_name*

Required. An absolute or relative path to a UNIX file to be used

for virtual tape input/output. Relative paths are considered to be relative to the current mvEnterprise startup directory (usually /production).

maxsize=*max_file_size*[*K*/*M*/*G*]

Optional. Specifies the maximum virtual tape file output size. After this limit is reached, the virtual tape driver will “cascade” its output to a new, sequentially-numbered output file (*outfile*, *outfile-1*, *outfile-2*, and so on.). This is acted upon only during virtual tape output and has no effect during tape input (read) operations, during which cascading is handled automatically. If no **maxsize**= parameter is specified, the output limit for virtual tape operations will be set to a default value of 4 GB (the normal file size limit imposed by the underlying UNIX file system). The number given in the **maxsize**= parameter may optionally be suffixed by a single character **K**, **M**, or **G**, where K=Kbytes, G=Gbytes, and M=Mbytes. M is assumed if no suffix character is specified.

The command parameters below are relevant only for tape **type=filtered**:

outfilter="*output_filter_command*"

UNIX-level script or command(s) to filter virtual tape output through. This must be enclosed in quotes if it includes any embedded spaces. Required if **type=filtered**.

infilter="*input_filter_command*"

UNIX-level script or command(s) to receive virtual tape input from. This must be enclosed in quotes if it includes embedded spaces. Required if **type=filtered**.

Description

NOTE—You must have the correct permissions to read or write to and from the UNIX file/directory you are attempting to access.

NOTE—The MTU clause is not supported in the **device**= parameter of the new T-ATT format.

All currently valid T-ATT commands used to attach tape and ethertape remain legal and will continue to function as they do now.

Command parameters in the new form take the general form of **keyword=value**. These keywords may be given in any order, but there must be no space between the “=” sign and the keyword or value on either side of it. Values containing embedded spaces must be enclosed by quotes. Keywords can be entered in upper, lower, or mixed case.

In the new form of the T-ATT command, the *unit_number* (or *device=unit_number*) and the **type=** parameters are required.

The system supports up to 32 tape-type devices (0-31). These devices can be physical tape, ethertape, or virtual tape.

Network drives can be used for input and output of virtual tape files. However the T-FWD and T-BCK commands are not guaranteed to function in this case.

The new T-ATT command assigns a physical tape, ethertape, or virtual tape device to the current line. If the specified device is attached to another line or does not exist, an appropriate error message displays. If the specified device is available, it is attached and a message displays indicating the drive number and the current blocksize for the device.

The new T-ATT form automatically performs all SET-DEVICE initialization. This eliminates the need to use the SET-DEVICE command to specify the tape device type.

All tape manipulation processes on the system check for attachment, attach the tape if possible, generate the required message, and either perform the desired tape operation or stop if the tape is not available.

When attempting to attach virtual tape, the file must be explicitly specified in the command. There is no automatic assignment of the last virtual tape file used. This is done to protect virtual tape files from accidental overwriting.

WARNING –A write to an existing virtual tape file automatically overwrites the existing data!

The tape blocksize may be explicitly specified on the command line. If no tape blocksize is specified on the command line, the default tape blocksize for the device is used. For **Type=tape** or **ethertape**, the blocksize defaults to the last attached blocksize. For **Type=file**, **compression**, or **filtered**, the blocksize defaults to 16384 bytes. If reading labeled tapes, the tape blocksize specified in the tape label is transferred to the device's tape block specification. After a device's tape blocksize has been initialized, it will persist until the device is detached unless it is changed by subsequent T-ATT operations.

The maximum virtual tape file output size may be explicitly specified on the command line with the **maxsize=** parameter. This is the actual amount of bytes, before any filtering or compression occurs. If not specified, the default output limit is 4GB. Once the limit is reached, the virtual tape driver cascades its output to a new, sequentially-numbered output file. For example, when the virtual tape file *outfile* reaches its output limit, the new sequentially numbered file *outfile-1* is created. When *outfile-1* reaches its output limit, *outfile-2* is created, and so on.).

The number given in the **maxsize=** parameter may optionally be suffixed by a single character **K**, **M**, or **G**, where **K**=Kbytes, **G**=Gbytes, and **M**=Mbytes. **M** is assumed if no suffix character is specified.

When using the virtual tape **type=compressed** parameter, output is automatically compressed by the UNIX **compress** program before being sent to a disk. Conversely, input is automatically decompressed by the UNIX **uncompress** program before being read. The UNIX **compress** and **uncompress** programs are the default programs pointed to by the **vtape_compress** and **vtape_uncompress** scripts. However, users can specify the compression program they want to use to process tape input and output. The only requirement is that the same program used to compress the virtual tape file must also be used to decompress it.

NOTE—Slower compression programs (or higher degrees of compression) adversely impact the speed of tape input/output.

A compressed virtual tape file does not allow users to use the **T-BCK** and **T-BSF** commands. Instead of using these commands, users should use the **T-REW** and **T-FWD** or **T-FSF**. Furthermore, users cannot perform reads immediately after performing a write. You must first perform a **T-REW**. Once this is accomplished, users will be able to use the **T-FWD** and **T-FSF** commands.

When using the **maxsize=** parameter in conjunction with a **type=compressed**, the maximum file size is calculated using the uncompressed size. Therefore, users should adjust their maximum file size estimate based upon the compression ratio of the compression program used. For example, a three-to-one compression ratio is normal for the default compression program used by the **type=compressed** parameter. Therefore, a **maxsize=6G** would produce a series of 2 GB output files.

The virtual tape **type=filtered** parameter allows users to specify their own UNIX-level command or program to process (filter) the input and output. This can be a different compression program, encryption routines, custom programs, and so on. The filter specified to process input can be completely different from the filter used to process output.

When using the **type=filtered** parameter, the user-provided pathname in the **path=** parameter should specify the actual virtual tape input/output filename. However, for some types of special processing, the filter command may need to manage all of its own input and output (as well as undertake the management of cascaded tape files). In such a case, the path name/file name specified in the **path=** parameter can be set to a dummy value as it will be ignored.

Multiple commands can be specified with the input or output filters using the standard UNIX pipe symbol conventions.

Simple T-ATT Form for Tape Devices and Ethertapes:

This is the current T-ATT command format used in existing mvEnterprise releases. Any currently valid T-ATT command using these command formats is still legal and will continue to function as it does now.

Format

```
T-ATT blocksize [, unit_number]  
T-ATT unit_number [, blocksize]  
T-ATT blocksize [, MTU hexadecimal_unit_number]  
T-ATT unit_number (blocksize)
```

Note: The above command forms may be optionally suffixed by an (**R**) option if the attachment is an ethertape receiver or a (**S**) option if the attached device is an ethertape sender.

Description

In the simple T-ATT form, the default device type is tape, but the SET-DEVICE command can set a device to tape or ethertape.

If no device-number is specified when attempting to attach tape or ethertape, the system attempts to attach to the last device accessed from the current line. If the last device accessed is available, it is assigned to the line and an appropriate message displays.

It is important to note that a certain amount of flexibility is built into the syntax for the simple T-ATT command. Four simple formats for T-ATT are supported. In all formats, block sizes and devices are specified as decimal values. The only exception is the third format, where device numbers, as part of the MTU clause, are specified in hexadecimal. In all formats, options always appear at the end of the command line.

Regardless of which simple format is used, block sizes and device numbers can be specified in any order. If only one of these parameters is specified, the system will take it as a device number if it is within the range of valid device numbers (0-31). Otherwise, the parameter will be taken as a block size and the system will attempt to attach to the last device accessed on the line. If the parameter is within the range of valid block sizes (80-16384), the device will be set to the specified block size. If the parameter is not within the range of valid block sizes, the device will be set to the default block size for the device.

If both a block size and device number are specified, at least one of the parameters must be a valid device number. The system will take the first parameter that is specified as a valid device number as the device to attach. The other parameter will be used as a block size. If the block size is not a valid block size, the default block size for the device will be used.

If neither parameter is a valid device number, an appropriate message will be displayed.

The sender (**S**) and receiver (**R**) options allow mvEnterprise machines to be linked by an ethertape device. A link between mvEnterprise machines is established by setting the same device-number on both machines to ethertape and attaching one machine as a sender and one as a receiver.

@SYSTEM.RETURN.CODE Values

The @SYSTEM.RETURN.CODE variable is a command-related, read-only variable set by the system. It can be used in conjunction with certain statements and functions to return the status of their operation after execution, and is accessible from within paragraphs or mvEnterprise Pick/BASIC programs. The table below summarizes values returned after a T-ATT operation:

@SYSTEM.RETURN.CODE	DESCRIPTION
0	Indicates a successful T-ATT operation.
-1	Indicates an unsuccessful T-ATT operation.

T-ATT Examples

The table below provides several example simple and new ATT commands.

T-ATT (SIMPLE FORM)	T-ATT (NEW FORM)	EXPLANATION
T-ATT	T-ATT device=10 type=tape	Attaches the last device accessed to the line, if possible. Initializes the tape blocksize to the default blocksize. In the new form, device and type must also be specified.
SET-DEVICE T10 T-ATT 4000	T-ATT device=10 type=tape blocksize=4000	Attaches device 10 to the line, if possible. Initializes the tape blocksize to the recommended 4000 bytes. In the new form, device and type must also be specified.
SET-DEVICE T10 T-ATT 10	T-ATT device=10 type=tape	Attaches device 10 to the line, if possible. Initializes the tape blocksize to the default blocksize.

		<p>the default blocksize.</p> <p>In the new form, type must also be specified.</p>
<p>SET-DEVICE T2</p> <p>T-ATT 16384 2</p>	<p>T-ATT device=2</p> <p>type=tape</p> <p>blocksize=16384</p>	<p>Attaches device 2 to the line, if possible. Initializes the tape blocksize to 16,384 bytes.</p> <p>In the new form, type must also be specified.</p>
<p>SET-DEVICE E1</p> <p>T-ATT 1R</p>	<p>T-ATT device=1</p> <p>type=ethertape</p> <p>options=R</p>	<p>Attaches ethertape device 1 to the line if possible, designating it as a receiver. Initializes the tape blocksize to the default blocksize.</p> <p>In the new form, type must also be specified.</p>
<p>SET-DEVICE T10</p> <p>T-ATT 4000 MTU</p> <p>A</p>	<p>N/A</p>	<p>Attaches device 10 to the line if possible. Initializes the tape blocksize to the recommended 4000 bytes. Note that with the MTU clause, tape device numbers are specified in hexadecimal.</p> <p>Not available in the new form because the MTU clause is not supported.</p>

N/A	T-ATT device=30 type=file path=file1 maxsize=6 blocksize=6000	Not available in the simple form. Attaches virtual tape device 30 to the line if possible. Designates the path for the file as "file1". Maximum virtual tape file output size of 6 MB. Initializes the tape blocksize to 6,000 bytes.
N/A	T-ATT device=30 type=compressed path=/home/saves/file2	Not available in the simple form. In the new form, attaches virtual tape device 30 to the line if possible. Designates the path for the file as "/home/saves/file2". Compresses output and decompresses input using the UNIX compression programs. Uses the default virtual tape file output size of 4 GB. Initializes the tape blocksize to the default 16,384 bytes.
N/A	T-ATT device=30 type=filtered path=file3 outfilter="/usr/sbin/gzip -q -3 -c" infilter="/usr/sbin/gunzip -q -c"	Not available in the simple form. In the new form, attaches the filtered virtual tape device 30 to the line if possible. Designates the path for the file as "file3". Uses the listed infilter and outfilter programs to process the input and output. Uses the default virtual tape file output size of 4 GB. Initializes the tape blocksize to the default 16,384 bytes.

Command synonyms

The following synonyms for the new command parameters shown above are provided to allow a degree of backward compatibility with the D3/AP "chg-device" command. D3-familiar users can add the command **chg-device** to their master dictionaries as a synonym for T-ATT to take full advantage of these alternatives.

<i>Normal parameter</i>	<i>Alternate Synonym(s)</i>
device=	mtu=
device=	unit=
path=	name=
path=	pathname=
path=	file=
blocksize=	blksz=
blocksize=	block=

type=compressed density=pseudo
type=tape type=8mm | 4mm | sct | half
options= (options (standard TCL options at the command end)
maxsize=nnn volsz=nnn
(A **vol**sz= specification is assumed to be in bytes unless explicitly suffixed by **M** or **G**)

mvEnterprise Telnet Server for AIX (mve_tel)

The mve_tel binary program is supplied with new AIX versions of mvEnterprise for use as an alternative inbound Telnet services facility. The program provides simple to implement services for connectivity by remote Telnet Client packages or hosts for direct attachment to a specified host user name (uid) and to then subsequently execute a user specified binary program or user specified script. Executing a script, easily allows this service to attach a remote Client directly to one or more mvEnterprise virtual machines.

The mve_tel service provides the following additional benefits to those connections normally made by a regular inbound telnet connection (via the standard telnetd service, TCP port 23) originating from a remote Telnet Client or host:

- mve_tel requires only 2 AIX processes for each connection during mvEnterprise operations, whereas a normal telnet connection will generally require 3 AIX processes.
- Using mve_tel does not require a connection to have an AIX user license, whereas each normal telnet connection requires an AIX user license for login. The mve_tel service completely bypasses AIX login and runs as the selected user, subsequently executing the user specified binary program or script designated in the mve_tel service setup.
- mve_tel presents connection identification information in the form of both local (Server) and remote (Client) ip address and TCP port used to the specified binary program or script.
An appropriately written program or script is then able to perform specific actions for this connection if required. These may include attaching the connection to a particular port on an mvEnterprise virtual machine, attaching it to one or more selectable pools of ports, barring (or refusing) the connection entirely either short-term for system housekeeping or long-term for system security reasons, or applying specific run-time options to the connection (e.g. -ol -qb options).
All actions performed for a connection made via mve_tel are solely under the control of the appropriate binary program or script. The simple mve_tel service interface merely presents the connection to these environments for subsequent processing.
- mve_tel provides a much higher degree of flexibility (on a per port basis) when using TCP/IP Terminal Servers, especially those allowing multiple session capabilities.

- `mve_tel` removes any requirement for TCP/IP Terminal Servers to sometimes need AIX tty style drivers to provide specific methods of connectivity. TCP/IP Terminal Servers can connect to `mve_tel` services by using the standard telnet command and specifying a particular TCP port (service) to gain access to specific AIX user environments as if they were directly hard-wire attached.
- `mve_tel` fully participates in telnet option negotiations with the Telnet Client or host, and thus will have the same ‘feel’ as a normal standard telnet connection. Session disconnects will process the same as for regular telnet sessions. `mve_tel` connections use the same standard pts devices that regular telnet sessions use.

mve_tel Services Setup

`mve_tel` services are setup to be initiated by the AIX internet daemon super-server called **inetd**, which listens for network service requests and starts the appropriate daemon to process the request. The **inetd** daemon is started at AIX system boot time from an initialization file such as `/etc/rc`. When it is started, **inetd** reads it’s configuration from the `/etc/inetd.conf` file and the `/etc/services` file. These files contain the names and other information of the services that **inetd** listens for and starts. You can add or delete services by making changes to both these files.

Making /etc/inetd.conf File Entries

An example of a `/etc/inetd.conf` file entry is:

```
ps1_tel  stream  tcp  nowait  root
/production/mve_tel  mve_tel  -w  /production
-p  mve_script7024
```

The entry is contained completely on 1 line entry and the fields in the `/etc/inetd.conf` entry are from left to right:

<i>name</i>	Name of the service, as listed in <code>/etc/services</code> file. In the sample entry the value in this field is <code>ps1_tel</code> .
<i>type</i>	Type of data delivery service used, also called the socket type. Commonly used socket types are:
	stream The stream delivery service provided by TCP, i.e., TCP byte stream.
	dgram The packet (datagram) delivery service, provided by UDP.
	raw Direct IP datagram service.

The sample shows that `ps1_tel` uses a stream socket, this is the correct setting for all `mve_tel` services.

protocol Name of a protocol, as given in the `/etc/protocols` file. Its value is usually either “tcp” or “udp”. The sample shows that `ps1_tel` uses TCP as its transport layer protocol and so contains `tcp` in this field, this is the correct setting for all `mve_tel` services.

wait-status The value for this field is either “wait” or “nowait”. Generally, but not always, datagram servers require “wait”, and stream servers allow “nowait”. If the status is “wait” then **inetd** must wait for the server to release the socket before it begins to listen for more requests on that socket. If the status is “nowait”, then **inetd** can immediately begin to listen for more connection requests on the socket. Servers with a status of “nowait” use sockets other than the connection request socket for processing; i.e., they use dynamically allocated sockets. The sample shows that `ps1_tel` server uses “nowait”, this is the correct setting for all `mve_tel` services.

uid User name under which the server runs. This is any valid user name. This can be **root**, but for normal security reasons will most likely be a particular user name setup for mvEnterprise operations. The sample shows that `ps1_tel` server runs as the root user.

server Full pathname of server program started by **inetd**. The supplied `mve_tel` executable should be copied into the working directory of an appropriate mvEnterprise virtual machine for which the service intends to connect (ensure you use `cp` or `tar` with the `-p` option). To maintain consistency the executable should retain its name of `mve_tel` and should have the appropriate ownership and privileges for correct operation:

```
i.e.  chmod 111 mve_tel
      chmod u+s mve_tel
      chown root.sys mve_tel
```

The sample shows that `mve_tel` was copied into a particular working directory and thus has the full pathname of `/production/mve_tel` to the executable server program.

arguments These are any command line arguments that should be passed to the server program when it is invoked. This list always starts with argv[0] (i.e. the servers executable name). In the case of the mve_tel services this will normally always be mve_tel and should always match the name only of the executable as given in the full server pathname described above. The sample shows the server program name to be mve_tel, as per the executable.

Two additional arguments must be provided in order for an mve_tel service to 'target' a particular binary program or script.

These are the **-w** and **-p** options as follows:

-w Followed by a space and then the full path to the working directory where the binary program or script resides. The sample shows **-w /production** indicating that /production is the working directory for connections using this service. mve_tel will perform a chdir (change directory) to this directory immediately prior to executing the binary program or script.

-p Followed by a space and then the name only of the binary program or script to finally execute. The sample shows **-p mve_script7024** indicating that mve_script7024 is the name of the binary program or script to execute for connections using this service.

As can now be seen, an mve_tel service attaches connections to a particular user and then in turn executes a specified binary program or script which may then be used to subsequently 'attach' the user to a particular mvEnterprise virtual machine if desired.

If mve_tel services are required to attach to more than one virtual machine, or additional mve_tel services target the same virtual machine, then additional unique service entries must be made within the **/etc/inetd.conf** and **/etc/services** files.

Making /etc/services File Entries

An example of a **/etc/services** file entry is:

```
ps1_tel          7024/tcp
```

The entry is contained completely on 1 line entry and the fields in the **/etc/services** entry are from left to right:

<i>name</i>	Name of the service, as listed in /etc/inetd.conf file. In the sample entry the value in this field is <code>ps1_tel</code> .
<i>port</i>	TCP port number which the service uses. This must not conflict with any other services which the AIX system may wish to use and therefore must be unique for each <code>mve_tel</code> service required. Normally TCP port numbers below 4096 are already pre-assigned to standard existing services (regular telnet 23, rlogin 513, etc.). It is suggested that TCP port numbers are carefully selected by the Network Administrator to avoid conflict with other services. Power95 standardly uses TCP port 7023 for a service similar to <code>mve_tel</code> and therefore systems containing both Power95 and mvEnterprise should avoid using 7023 for <code>mve_tel</code> . It is suggested that TCP port 7024 onwards should be safe allocations for <code>mve_tel</code> services, but this should be checked rigorously before allocation. Some Telnet Servers may only be able to provide Telnet Client connections on certain TCP port allocations, or the manufacturer may make specific recommendations, please adhere to these if possible. The sample shows that service <code>ps1_tel</code> will use TCP port 7024 for inetd listening.
<i>/protocol</i>	The protocol that the service uses. <code>mve_tel</code> only uses tcp protocol. Thus the sample shows the full entry of <code>7024/tcp</code> for tcp protocol.

Implementing **/etc/services** and **/etc/inetd.conf** Changes

The complete configuration for **inetd** is located in the **/etc/services** and **/etc/inetd.conf** tables. These tables are mirrored by the **InetServ** object class in the AIX ODM. Any changes to the tables must be reflected in the ODM. Changes made to these tables through SMIT will automatically be incorporated in the table and the ODM. SMIT updates will also refresh **inetd**. If you want to maintain the tables using your favorite editor, you may use **inetimp** to subsequently copy table information to the ODM. e.g.:

```
# inetimp Import inetd.conf and services to InetServ ODM.
```

Any time updates are made to these tables, you will need to refresh **inetd**. This can be done with the SRC **refresh** command or by sending a **hangup signal** to the `inetd` process. e.g.:

```
# refresh -s inetd  
or  
# kill -HUP <inetd-pid>
```

```
or
# kill -1 <inetd-pid>
```

Example mve_tel Services Setup

The following example shows the complete `/etc/services` and `/etc/inetd.conf` line entries for a system that desires to have `mve_tel` services 7024 and 7026 connect to a root user in working directory `/production` and then subsequently execute a script named `mve_script702x` residing in that working directory, and `mve_tel` service 7025 to connect to a root user in working directory `/oldver` and then subsequently execute a script named `mve_script7025` residing in that working directory:

/etc/services entries:

```
ps1_tel    7024/tcp
ps2_tel    7025/tcp
ps3_tel    7026/tcp
```

/etc/inetd.conf entries:

```
ps1_tel stream tcp nowait root
/production/mve_tel mve_tel -w /production
-p mve_script702x
ps2_tel stream tcp nowait root
/oldver/mve_tel      mve_tel -w /oldver
-p mve_script7025
ps3_tel stream tcp nowait root
/production/mve_tel mve_tel -w /production
-p mve_script702x
```

The setup would be completed by entering these commands:

```
# inetimp
# refresh -s inetd
```

In these examples, Telnet Clients initiating telnet connections with TCP ports 7024 and 7026 will connect to the same user name and working directory and execute the same script, but with different service entries. TCP port 7025 will connect to the specified user name and other working directory and execute the specified script. It is assumed that different processing may be effected for ports 7024 and 7026 in this instance, even though they connect to the same user name and working directory and execute the same script.

Parameters Passed to Binary Program or Script

The binary program or script will be executed from mve_tel with connection identification information being handed as parameters. The passed parameter syntax is:

**AAA.AAA.AAA.AAA BBBB CCC.CCC.CCC.CCC
DDDD**

where

AAA.AAA.AAA.AAA is the dotted decimal ip address of the remote telnet client, e.g. 140.235.66.182 .

BBBB is the TCP port number used by the remote telnet client , e.g. 6512 . (see note below)

CCC.CCC.CCC.CCC is the dotted decimal ip address of the local host running mve_tel, e.g. 140.235.66.181 .

DDDD is the local host mve_tel service port number, e.g. 7024 .

Using these unique parameters it is possible for a suitably written program or script to effect specific connection requirements making access to some special look-up file or table or other scheme.

NOTE:

Systech RCS4000 Client TCP Port Number Designations:

The Client TCP/IP port number for a telnet connection made from a shell or configuration on an RCS based serial port has the form:

lppss

where

pp is the serial port running the telnet cmd (01-16)

ss is the session ID. The session ID is 00 if the port is not configured for multi-sessions or the value 00-03 if the port is configured for multi-sessions.

Digi PortServer II Client TCP Port Number Designations:

The Client TCP/IP port number for a telnet connection made from a PortServer II Terminal Server is dynamic and therefore it is not possible to "attach" via a known Client TCP port designation.

mve_tel Deliverables and Sample Components

The mvEnterprise delivery package includes the mve_tel binary program and also supplies a sample script for connecting to one or more mvEnterprise virtual machines together with a sample data file of mve_tel 7024 service port assignments for making these connections.

Both of these samples can be used immediately for this purpose with only little modification, both are listed below in their entirety. They should be copied to any working directory (and suitably renamed) that has been selected for mvEnterprise connection(s) using the mve_tel services. Ensure you use cp or tar with the -p option to preserve all module privileges and ownership.

The delivered mve_tel7024.script_sample file contains the following line entry to specify the name of the mvEnterprise monitor program to be executed within the local path (working directory):

```
mvecmd='./prodpick'           # This is
the program name (local path) to exec
```

The name of the mvEnterprise monitor program (prodpick) should be changed in any operational scripts if it differs from the sample script contents.

The eternal_sleep binary program is delivered with this version of the mve_tel package and is a support module for the OPENSER feature now included in the mve_tel7024.script_sample file.

mve_tel Deliverable Components for AIX

```
---s--x--x  1 root    sys      6568 Nov 16 15:04 eternal_sleep
---s--x--x  1 root    sys      58353 Nov 16 15:03 mve_tel
-rw-r--r--  1 root    sys      8197 Sep 01 09:38 mve_tel7024.ports_sample
-rwx--x--x  1 root    sys      5902 Sep 01 09:38 mve_tel7024.script_sample
#
```

mve_tel7024.script_sample

```
#!/usr/bin/ksh
# Sample korn shell script for use with mvENTERPRISE mve_tel Telnet Server
# Rev 01 BJC 03/03/99 Initial Release. General Automation, Inc.
# Rev 02 TJH 08/10/99 Modified for new mve_tel and quad address parameters
# Rev 03 BJC 09/23/99 Modified for Client TCP port processing
# Rev 04 BJC 05/22/00 Allow connections based on remote ip address octets so
#                   that group connects are available. Give option to supply
#                   NOCONNECT as the line option as means to deny connections
# Rev 05 BJC 08/30/00 Add OPENSER functionality to sleep a pts connection.
#                   Uses eternal_sleep binary added to mvEnterprise package.
# Rev 06 BJC 08/31/00 Add RECONNECT functionality for undetected disconnected
#                   output only socket for terminal server print type device.
#
# The script expects to be passed (by the mve_tel daemon) four parameters
# of the form: AAA.AAA.AAA.AAA BBBB CCC.CCC.CCC.CCC DDDD
```

```

# Where A and B are the remote IP address and port (Client)
#   and C and D are the local  IP address and port (Server)
# The script uses this information to obtain an appropriate line entry in an
# associated data file (e.g. mve_tel7024.ports) to exec a user defined
# executable with appropriate parameters.
# This executable will normally be a particular mvENTERPRISE virtual monitor
# program residing in the targetted working directory, but could also be ANY
# executable program or additional script used for other purposes.
#
# This sample script provides limited connection facilities and should be used
# as the basis for the user's requirements. It is recommended that this sample
# script remains intact, and the user works on an appropriately named copy.
#
Message()
{
reach=$(hostname)
[ ! -z "${*}" ] && echo "\n${reach}: ${*}"
echo "${reach}: $(date)\n${reach}: Pausing, Please wait ... \n"
sleep 5
echo
}

Bail()
{
Message ${*}
stty ${sttysv}          # allow breaks to happen again
exit 1
}

#####

export PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:.

# The ONLY 2 user changeable parameters are below:

access="allow"          # Used to control access to the system:
                        # A null string ("") denies system access.
                        # Any other string (e.g. "allow") allows
                        # system access to the user.

mvecmd='./prodpick'     # This is the program name (local path) to exec

# The ONLY 2 user changeable parameters are above.

stty sv=$(stty -g)     # save the current tty settings
stty intr ^@           # stop any break depressions from happening

if [ -z "${access}" ] ; then
    reach=$(hostname)
    echo "\n${reach}: Access currently denied, please attempt request later"
    echo "${reach}: $(date)\n${reach}: Pausing, Please wait ... \n"
    sleep 30
    stty ${sttysv}      # allow breaks to happen again
    exit 1
fi

ripaddr=${1}
ripport=${2}
lipaddr=${3}
lipport=${4}
ripap="${1}+${2}"
portfile='mve_tel'${lipport}'.ports'

[ -f ${portfile} ] || Bail "Connection aborted, local ${portfile} file not found"

mveline=$(grep "^${ripap} " ${portfile} | cut -f2 -d":")

if [ -z "${mveline}" ] ; then
    mveline=$(grep "^${ripaddr} " ${portfile} | cut -f2 -d":")
fi

```

```

if [ -z "${mveline}" ] ; then
    octet1=$(echo ${ripaddr} | cut -f1 -d'.')
    octet2=".$(echo ${ripaddr} | cut -f2 -d'.')"
    octet3=".$(echo ${ripaddr} | cut -f3 -d'.')"
    octets="${octet1}${octet2}${octet3}"
    mveline=$(grep "^${octets}" " ${portfile} | cut -f2 -d":")
fi

if [ -z "${mveline}" ] ; then
    octets="${octet1}${octet2}"
    mveline=$(grep "^${octets}" " ${portfile} | cut -f2 -d":")
fi

if [ -z "${mveline}" ] ; then
    mveline=$(grep "^${octet1}" " ${portfile} | cut -f2 -d":")
fi

if [ -z "${mveline}" ] ; then
    echo "\n$(hostname): No valid entry for ${ripaddr} in local ${portfile} file\c"
    Bail Please contact the System Administrator to gain correct access
fi

if [ "${mveline}" = "NOCONNECT" -o "${mveline}" = " NOCONNECT" ] ; then
    reach=$(hostname)
    echo "\n${reach}: Access is being denied by System Administrator"
    echo "${reach}: $(date)\n${reach}: Pausing, Please wait ... \n"
    sleep 10
    stty ${sttysv}                # allow breaks to happen again
    exit 1
fi

# Check for OPENSER= connect. If it is then we just need to drop the pts
# device name into the users special unique file and then infinitely
# sleep until we finally disconnect.
# Pass params to eternal_sleep to identify process in ps -elf cmd for admin
#
keyser=$(echo ${mveline} | cut -f1 -d"=")
if [ "${keyser}" = "OPENSER" -o "${keyser}" = " OPENSER" ] ; then
    filseq=$(echo ${mveline} | cut -f2 -d"=")
    who -m | read x namser x && echo "/dev/${namser}" > ${filseq}
    exec ./eternal_sleep ${filseq} ${mvecmd}
    exit 0
fi

quiet=1

# Check for RECONNECT= syntax. There should be NO spaces in front of 1st
# line option because ps removes all excessive white space in cmd line !!!
if [ "${keyser}" = "RECONNECT" -o "${keyser}" = " RECONNECT" ] ; then
    quiet=0
    mveline=$(echo ${mveline} | cut -f2 -d"=")
    ps -ef|grep -F "${mvecmd} ${mveline}"|grep -v grep|while read x pid x ; do
        kill -1 ${pid} >/dev/null 2>&1      # sometimes takes 2 shots to kill
        kill -1 ${pid} >/dev/null 2>&1      # sometimes takes 2 shots to kill
    done
    sleep 1
done
fi

${mvecmd} -oq                # check whether mvE virtual is active
if [ $? -eq 0 ]
then
    stty ${sttysv}          # allow breaks to happen again
    exec ${mvecmd} ${mveline}
else
    if [ ${quiet} -eq 1 ] ; then
        reach=$(hostname)
        echo "\n${reach}: System currently unavailable, please attempt request later"
        echo "${reach}: $(date)\n${reach}: Pausing, Please wait ... \n"
    fi
    sleep 30
    stty ${sttysv}          # allow breaks to happen again

```

```

    exit 1
fi
# Should never get here unless exec fails !!!
Bail "exec failure of ${mvecmd} ${mveline}"
exit 0

```

mve_tel7024.ports_sample

```

#####
# Sample file of mve_tel 7024 service port assignments
# Naming convention is mve_tel????.ports and must be placed in working directory
# and have appropriate read permissions and ownership.
# Where ???? is the mve_tel TCP service number (e.g. 7024).
#####
# Rev 01 BJC 02/22/99 Initial Release, GA Inc.
# Rev 02 BJC 09/23/99 Modified for mve_tel daemon quad parameter passing.
# Rev 03 BJC 05/22/00 Allow connections based on remote ip address octets so
#
#         that group connects are available. Give option to supply
#         NOCONNECT as the line option as means to deny connections
# Rev 05 BJC 08/30/00 Add OPENSER functionality to sleep a pts connection.
#
#         Uses eternal_sleep binary added to mvEnterprise package.
# Rev 06 BJC 08/31/00 Add RECONNECT functionality for undetected disconnected
#
#         output only socket for terminal server print type device.
#####
# Format is single line entries for each required line connection setup
# in the form of the following defined fields:
#
# "Client connection identifier" "comment":"line options"
#
# Where "Client connection identifier" is a sub-field of the form:
# "Client dotted decimal ip address"
# for those mve_tel Clients only effecting a single unique ip address connection
# e.g. 128.8.1.1          (See examples below).
# or
# "Client dotted decimal subnet address"
# for those mve_tel Clients effecting group connections based on subnet addr
# e.g. 128.8.1          To group apply against the 3 octet subnet
#     128.8            To group apply against the 2 octet subnet
#     128              To group apply against the 1 octet subnet
# or
# "Client dotted decimal ip address"+"Client TCP port"
# for those mve_tel Clients wishing to effect multiple unique ip address
# connections, normally TCP based Terminal Servers or multi-session Clients
# e.g. 128.8.1.5+10100  (See examples below).
# It is important to include the + character here to append the Client TCP port
# for scanning purposes, no other character is valid and a space will revert the
# entry back to a single unique ip address connection type.
#
# Where "comment" is an OPTIONAL sub-field containing any normal Ascii character
# sequence, except the character : which is used to define the line options area
#
# Where "line options" are standard mvENTERPRISE Virtual Monitor program
# connection options appropriate for operations. e.g. -qb -ol
# There must be at least 1 compulsory option entered into this field.
#
# Other "line options" available:
#-----
# NOCONNECT
# The value NOCONNECT may be placed in this field as a means to deny access to
# the entry, whether it be for a single ip address or group subnet connection.
# If the value NOCONNECT is used, it must be the only "line option" entered.
# Individual ip address entries will override group entries regardless of order
#-----
# OPENSER=filename
# Syntax required to setup a "sleeping" pts connection for OPENSER operations
# within an mvEnterprise virtual environment.

```

```

# The filename is a unique user selected filename (and path) to be used to
# contain the pts connection device information for this connection ONLY.
# Other OPENSER connects must also have their own unique filename (and path).
# e.g. OPENSER=/tmp/SERpts01 and OPENSER=/tmp/SERpts02, etc.
# This unique connection file is then used in a BASIC program to obtain the
# identity of the "sleeping" pts connection for OPENSER operations.
# Example usage BASIC coding:
# OPENSEQ EXISTING "/tmp","SERpts01" to SERDEV on ERROR ..fail user code..
# READSEQ DEVNAME FROM SERDEV
# CLOSESEQ SERDEV
# OPENSER DEVNAME TO REMDEV ELSE ..fail user code..
# SEND "some data" TO REMDEV ELSE ..fail user code..
# GET DUMMY FROM REMDEV UNTIL CR:LF RETURNING TERM.CHAR WAITING 30
# CLOSE REMDEV
#-----
# RECONNECT="line options"
# This is a special syntax to aid with the situation where a mvEnterprise
# virtual monitor program is unable to detect a detached socket, due to the
# remote terminal server device not supplying input characters to the socket.
# This is normally the case for a remote connected print type device acting
# as an output only process.
# This case will also attempt to kill any previous socket connection made to
# the same mvEnterprise virtual monitor line designation, to avoid the problem
# where the line is already in use and the connection is then refused.
# To further assist in this type of print connection, this syntax will also
# invoke (as best as possible) a "quiet" mode of operation so as to avoid
# issuing unnecessary messages to print devices with special forms loaded.
# "line options" are standard mvENTERPRISE Virtual Monitor program connection
# options appropriate for these operations. e.g. -lxxx -ot
# Reconnection rules demand at least the -lxxx option be used to specify the
# desired target line.
# When using this syntax there must be NO spaces between the RECONNECT= and
# the first line option entered.
#-----
#
# NOTES:
# 1: The space after the Client connection identifier is compulsory and
#    delimits the sub-field.
# 2: The : char delimits any comment or the space after Client connection
#    identifier if there is no comment field. This is compulsory.
# 3: There must be only one occurrence of the : character in any line entry
# 4: Blank lines are valid to make the file more readable.
# 5: Comment lines are valid and start with any non ip address text e.g. #.
# 6: Ordering of individual line entries is not important for correct
#    attachment scanning, assuming all information is correct.
#
#####
# Example entries (commented out with leading #):

#192.1.1.10 tom : -l4 -v192 -qb -ol
#192.1.1.11 kay : -l5 -qb -ol
#192.1.1.12 fred : -l99 -qb -ol

#128.8.1.1 brian PC: -l555 -v192 -qb -ol
#128.8.1.3 :-l88 -qb -ol
#128.8.1.4 :-qb -ol

#128.8.1.5+10100 Systech RCS4000 port 01 session 00 : -l150 -qb -ol
#128.8.1.5+10203 Systech RCS4000 port 02 session 03: -l151 -qb -ol
#128.8.1.5+11600 Systech RCS4000 port 16 session 00 : -l152 -qb -ol
#128.8.1.5 Systech RCS4000 any other port/session than above : -l153-165 -qb -ol

#128.8.1.6+10800 port 08 Interface :OPENSER=/tmp/SERpts01
#128.8.1.6+11200 port 12 Interface :OPENSER=/tmp/SERpts02
#128.8.1.6+11300 port 13 print device :RECONNECT=-l166 -ot
#128.8.1.6 Systech RCS4000 any other port/session than above : -qb -ol

#128.8.1.7+11400 port 14 Interface :OPENSER=/tmp/remote714
#128.8.1.7+11500 port 15 print device :RECONNECT=-l167 -ot
#128.8.1.7 Systech RCS4000 any other port/session than above : -qb -ol

```

```
#128.8.1.76 deny access to this addr :NOCONNECT
#128.8 any other addr on this subnet is dynamic : -qb -ol
```

```
#####
Notes on Systech RCS4000 Client TCP port number designations:
(Taken from Systech RCS4000 manual)
```

The Client TCP/IP port number for a telnet connection made from a shell or configuration on an RCS serial port has the form:

lppss

where

pp is the serial port that is running the telnet command (01-16)
ss is the session ID. The session ID is 00 if the port is not configured for multi-sessions or the value 00-03 if the port is configured for multi-sessions.

```
#####
Notes on Digi PortServer II Client TCP port number designations:
```

The Client TCP/IP port number for a telnet connection made from a PS II Terminal Server is dynamic and therefore it is not possible to "attach" via a known Client TCP port designation.

```
#####
Insert your live host remote connections after this line
```

This is the end of the file.

roll_log_pseudo Cron Script

The roll_log_pseudo cron script is delivered for AIX systems as a standalone application independent module which, when configured correctly, is automatically executed periodically by an AIX crontab entry at user selected times. The AIX crontab service has the same features found on other flavors of UNIX and generally operates in an identical manner, therefore it is a very simple exercise to adapt this script and methodology to other flavors of UNIX.

The roll_log_pseudo cron script facilitates 4 primary functions, as follows:

- Provides a configuration template cron script with a user site specific definable section for editing, which is then run with the install option to automatically create a real cron script (by another unique name, user defined) and crontab entry.
- Provides an uninstall option for the created real cron script which, when the option is exercised from the real cron script, will remove the appropriate crontab entry and thus terminate the execution of the real cron script. This real cron script is not removed from the system, and therefore can be restarted/reinstalled later.
- Optionally, scans a user described log file for all existing log messages containing a unique user defined text string. Log messages that are found are then emitted onto the network as specific E-mail alert notifications (using standard AIX mailx and sendmail SMTP services) to one or more named E-mail recipients. Normally most log files have time stamped log messages, in this (usual) case only log messages that are found time stamped since the previous scan are emitted as E-mail alert notifications to those named E-mail recipients.

The primary purpose of this function is generally in the case of a system that does not have system administration staff persistently monitoring such things as log files for major events or events that are critical to the continued normal operations of the system (or systems).

A particular example of use of this feature, for which it has been originally developed, is the case of mvEnterprise Transaction Logging overflow acquisition instances, which are critical to the normal operations of an mvEnterprise fault-resilient configuration. As supplied, the site parameter section of the initial delivered roll_log_pseudo cron script is specifically setup for correctly scanning the above example condition.

This E-mail alert function is not designed to completely remove the necessity of system administration staff conducting persistent system monitoring for these critical and important events, but is specifically designed to augment and assist in these normal and natural system orientated duties.

Currently, the E-mail alert layout contains additional information in keyword format that is more than sufficient to identify the source of the E-mail alert by machine automated capture techniques. It is envisaged that this E-mail alert feature will eventually become a major component of a much larger Remote Machine Monitoring plan that is currently being designed for implementation within the Raining Data Support Division, this plan will evolve into a “phone home” style of proactive system support facility.

- Optionally, the `roll_log_pseudo` cron script can also be configured to roll the log file over into additional daily log files at the midnight execution of the crontab entry, assuming the script has been configured to run at hour 0 minute 0 of each day.

This operation has a number of benefits, particularly for systems with long-term zero downtime. The most major benefit being that the log file is then packaged into more manageable and more easily accessible components. This has the added benefit, during the cron script scanning procedure, of only scanning the log file containing near recent log message entries, thus significantly reducing the periodic scan overhead on the system. Additionally, the log file history is also truncated at a suitable number of log files that will no longer burden the system with excessive and totally unnecessary usage of stale file system storage.

The `roll_log_pseudo` cron script is specifically termed “pseudo” in that it is not the real cron script that will be setup to run in the crontab entry. The pseudo script is used solely as a template during configuration and installation operations to define the unique real cron script that will be installed and used by the crontab entry execution.

The configuration settings are controlled by a user defined site-specific data section of the pseudo script, which is then run with the `install` option to create a new and unique real cron script and subsequent crontab entry.

Using this technique allows the single defined `roll_log_pseudo` cron script to be consistently used as a template to easily configure and install multiples of near similar unique cron scripts, each with it’s own `roll_log_pseudo` style, onto a system with exactly the same methodology.

To create a valid and usable script for the crontab entry a user must perform the following tasks:

1. Place the `roll_log_pseudo` cron script into the directory of choice.

2. Edit the roll_log_pseudo cron script with all the necessary information for creating a new unique named script to perform the desired tasks correctly.
3. Use the roll_log_pseudo script to automatically create and install this new unique named script and create the appropriate crontab entry.

roll_log_pseudo Cron Script Delivery and Placement

The roll_log_pseudo cron script for AIX is normally delivered in tar format, either already as part of a major package supplied from Raining Data, or separately via ftp (binary) download or via floppy disk media.

In all cases the resultant executable roll_log_pseudo cron script should be placed into the directory of choice before any editing and/or automatic install sequence process is effected.

In the case of floppy disk media supply this will entail the following operations as **root** user-id:

```
cd /directory_of_choice
tar -xvpf /dev/fd0
chmod 755 roll_log_pseudo
chown root.system roll_log_pseudo
```

Similar operations will also be required for an ftp binary download.

roll_log_pseudo Cron Script Deliverable Components for AIX (Subject to Change)

```
-rwxr-xr-x  1 root      system      9549 Jan 08
23:20 roll_log_pseudo
```

Editing the Changeable Parameters in the roll_log_pseudo Cron Script

The following user definable site specific parameters are based on requirements to monitor and alert for an mvEnterprise system and Transaction Logging overflow acquisition instances:

```
##### MAKE SITE SPECIFIC CHANGES BELOW THIS LINE #####
scrname="roll_pick_log"      # New unique name of script, DO NOT include
                             # the directory path. This script will be auto
                             # created during the install process and used
                             # for the crontab entry.
period="0,15,30,45 * * * *" # period to run as cron, usual cron rules.
                             # "0,15,30,45 * * * *" is every 15 mins, every
```

```

# hour, every day of month, every month of
# year, every day of week.

logfile="pick_log"      # log file, change if not same
logdir="/usr/stat"    # log file directory, change if not same

rollit="yes"           # "yes" to roll the log file each midnight
                       # (assumes cron will run at 0 hours 0 mins)
numrf=31              # No. of daily roll files to keep, can change
                       # Must be at least 2, if rollit="yes"
aperm="666"           # access permissions for new log file
owner="root.system"   # ownership for new log file

scanit="yes"          # "yes" to scan log file for user defined text entries

# Log entry date/time column defs for chronological
# tests, all column designations are base 1 for count.
# defines year as "yes" = ccy or "no" = yy, this used
# so we can pick off current TOD data from date cmd
# defines year ccy columns, "n-m" or "" for no year
century="yes"         # defines month as alpha Jan, Feb, etc, or mm in decimal
# defines month AAA/mm columns, "n-m" or "" for no month
colyr=""              # defines day dd columns, "n-m" or "" for no day
mthalpha="yes"        # defines hours hh columns, "n-m" or "" for no hours
colmth="1-3"          # defines minutes mm columns, "n-m" or "" for no mins
colday="5-6"          # defines seconds ss columns, "n-m" or "" for no secs
colhrs="8-9"
colmin="11-12"
colsec=""

gtext="Logger has acquired" # Unique message text to locate in log file
                           # this will also dictate the ekey type below

# IMPORTANT E-mail information
ekey="TYPE:N123456"      # alert type for receive processing
from="FROM:GAL999999"   # sender contract system id
site="CUST:Acme Co."    # sender customer company name
sware="SYST:mvEnterprise" # system type
mailme1="sysadmin@mail.acme.com" # mail 1st target, change to recipient
mailme2=""              # mail carbon copy recipients,
                           # separate multiples with spaces

```

DO NOT CHANGE BELOW THIS LINE

Description of Changeable Parameters

The following list describes all of the user changeable parameters available. All parameters are defined by a keyword= syntax which must be strictly adhered to, since these keywords are actually ksh (Korn Shell) variable allocations. All user changeable data must be defined within a pair of string style double quotes (“”) as per the examples, except for the parameter numrf= which takes a regular decimal count value only.

- **scrname="roll_pick_log"**
This is the new unique name of the real cron script to be automatically created. Do not include the directory path - when using the install option the new script will be created into the same directory path where roll_log_pseudo is stored and executed from.
- **period="0,15,30,45 * * * *"**
This is the period to run as a cron job and follows the usual crontab

rules.

Five fields are defined in this string, as follows:

“minute hour day_of_month month weekday”

The example “0,15,30,45 * * * *” will run the cron job every 15 minutes of the hour, every hour of the day, every day of the month, every month, every day of the week.

These fields accept the following values:

minute, 0 through 59; hour, 0 through 23; day_of_month, 1 through 31; month, 1 through 12; weekday, 0 through 6 for Sunday through Saturday

You must specify a value for each field. These fields can contain the following:

A number in the specified range:

To run a cron job in May, specify 5 in the month field.

Two numbers separated by a dash to indicate an inclusive range:

To run a cron job on Tuesday through Friday, place 2-5 in the weekday field.

A list of numbers separated by commas:

To run a cron job on the first and last day of January, you would specify 1,31 in the day_of_month field.

An * (asterisk), meaning all allowed values:

To run a cron job every hour, specify an asterisk in the hour field.

Note: Any character preceded by a backslash (including the %) causes that character to be treated literally.

The specification of days may be made by two fields (day of the month, and day of the week). If you specify both as a list of elements, both are adhered to:

For example, the following entry: “0 0 1,15 * 1” would run on the first and fifteenth days of each month, as well as every Monday.

To specify days by only one field, the other field should contain an *

- **logfile="pick_log"**

This is the name of the log file to optionally rollover or scan. Do not include the directory path in this definition.

- **logdir="/usr/stat"**

This is the directory path to where the log file is stored.

- **rollit="yes"**

Enter “yes” to roll the log file each midnight, else enter “no”. This assumes that the cron job will be scheduled to run at 0 hours 0 minutes of the day.

In the case of multiple scheduled cron jobs accessing the same log file, it is important that only one of these jobs is defined to midnight rollover the log file, if desired.

Candidate log files for midnight rollover should ONLY contain logged message text entries, they should not contain any embedded log file control structures.

This will most likely be the case with regard to application created log files with message text only entries, however particular care should be taken when selecting AIX operating system standard log files which may have embedded control structures (possibly such log files as the AIX errpt file /var/adm/ras/errlog, etc.)

- **numrf=31**

Defines the number of daily roll log files to keep, must be at least 2, if rollit="yes".

Each of these daily log files has the same original log file naming convention with the added suffix of a . (period) followed by an incrementing decimal counter, up to the decimal number described. For example the log file /usr/stat/pick_log will be rolled into daily historic log files of /usr/stat/pick_log.1 /usr/stat/pick_log.2 etc., where the decimal suffix counter will define a daily log file of that number of days preceding the current day "live" log file still retaining the name /usr/stat/pick_log (in our example). These roll log files will still retain the date/time stamp of the last update made to them prior to being rolled over, as normally seen in the standard AIX ls -al command.

Roll log files that eventually attain the maximum suffix counter defined by the numrf= parameter are lost at the next midnight rollover of the files, at which time all other rolled log files will increment their suffix counter.

- **aperm="666"**

Defines the AIX access permissions required for the new "live" log file that will be created as a null file (empty) during the midnight rollover process. This value should be defined to generate identical file permissions in force for the "live" log file prior to any midnight rollover procedure setup. The value defined must follow standard access permission value rules as normally used in the AIX chmod command.

- **owner="root.system"**

Defines the AIX owner and group required for the new "live" log file that will be created as a null file (empty) during the midnight rollover process. This value should be defined to generate identical file ownership in force for the "live" log file prior to any midnight rollover procedure setup. The value defined must follow standard ownership rules as normally used in the AIX chown command.

- **scanit="yes"**

Enter "yes" to scan the log file for user defined text entries, else enter "no".

If selected, the "live" log file will be scanned for the text string defined by the gtext= parameter. Any log file message entries found to contain this text string will be emitted as an E-mail alert notification to the defined recipients, assuming the message entry was inserted into the log file (with time stamp) since any previous scheduled scan process.

The following entries describe the log file message entry date and time column definitions. These are used for chronological tests to determine if a log file entry has been inserted into the log file since the last scan. If no time/date column entries are defined (i.e. all set as the null string ""), then each log message satisfying the scan criteria will be emitted as an E-mail alert notification each time the cron job is run. In this case it would be good practice to roll the log file over at each midnight to minimize the possibility of re-emitting historic log entries. All column designations are base 1 for counting.

- **century="yes"**
Defines the log message year style so that the correct year format can be picked off the current time of day data that the cron script retrieves by using the AIX date command with special format. This information is important for correct chronological tests.
"yes" will define the year as a 4-digit century style of ccyy
"no" will define the year as a 2-digit year only style of yy
- **colyr=""**
Defines the log message year ccyy or yy columns, "n-m" or "" for no year recorded.
In the case of no year information, the setting of century= will be irrelevant.
- **mthalpha="yes"**
Defines the log message month style.
"yes" will define the month as a 3 character alpha format Jan, Feb, Mar, etc., alpha months will be converted to 2-digit decimal values 01-12 for chronological tests.
"no" will define the month as a standard 2-digit decimal value 01-12.
- **= "1-3" colmth**
Defines the log message month AAA/mm columns, "n-m" or "" for no months.
In the case of no month information, the setting of mthalpha= will be irrelevant.
- **colday="5-6"**
Defines the log message day dd columns, "n-m" or "" for no days recorded.
- **colhrs="8-9"**
Defines the log message hours hh columns, "n-m" or "" for no hours recorded.
- **colmin="11-12"**
Defines the log message minutes mm columns, "n-m" or "" for no minutes recorded.
- **colsec=""**
Defines the log message seconds ss columns, "n-m" or "" for no seconds recorded.

- **gtext="Logger has acquired"**
Defines the unique user selected string text as the criteria for selecting log file messages as E-mail alert notification candidates. This will also dictate the ekey= type below.

The following entries describe the various information required that relates to the content and recipients of any emitted E-mail alert notifications. This information is structured to provide specific keywords within the text body of an E-mail, such that a machine automated capture technique may be able to identify and correctly forward process the alert. At present, the keywords embedded into the E-mail text body are TYPE:, FROM:, CUST:, SYST:, this minimal information should be sufficient for auto identification and processing requirements.

- **ekey="TYPE:N123456"**
Defines the TYPE: keyword alert notification class and code. The class is a leading alpha character describing the type of alert notification. In this example the class is N (notification), other classes may be designated such as E (error), I (information), etc. The code is a unique code that defines the type of alert notification within that class. In this example the code is 123456, which will have some meaning within the N class.
- **from="FROM:GAL999999"**
Defines the FROM: keyword, which should contain the customers Raining Data support contract ID, in this example GAL999999. This is the main method of identifying both a customer site and a particular system within that site.
- **site="CUST:Acme Co."**
Defines the CUST: keyword, which is used to describe the customer company name, in this example Acme Co. This is used solely to enhance the emitted alert notification for any human E-mail readability.
- **sware="SYST:mvEnterprise"**
Defines the SYST: keyword, which is used to describe the type of system from which the alert notification is being emitted, in this example mvEnterprise. This is used solely to enhance the emitted alert notification for any human E-mail readability. Other system types might be listed as: Power95, D3/AIX, etc.
- **mailme1="sysadmin@mail.acme.com"**
Defines the primary E-mail target recipient of the alert notification. The E-mail address contained in double quotes must be a valid reachable target.
This entry will normally represent the person internal to the customer company that has system administration responsibilities. It is important that the E-mail address entered here is a valid and reachable target. The cron script emits the E-mail messages by using the standard AIX mailx command. mailx uses the standard AIX

sendmail SMTP services to dispatch mail. The correct delivery of mail to a target from AIX is dependant upon the retrieval of correct domain resolution from an accessed Domain Name Server (DNS). It is important that the site network mail and DNS components are setup to provide these correct services.

To test whether an E-mail target recipient is reachable from the AIX mail system, the following steps can easily be performed:
Login to AIX as the root user-id and type the following command at the shell prompt, assuming the E-mail target recipient to test is sysadmin@mail.acme.com :

```
echo "TYPE:T911411\n\nTesting" | mailx -s 'RD911 TEST'  
sysadmin@mail.acme.com
```

If the target recipient is reachable then the E-mail test should be received correctly by the target recipient.
If the target recipient is not reachable then the AIX mail services will normally send a response E-mail to the root user-id (sender) with an embedded explanation of the delivery problem. Check the root user-id for mail by running the standard AIX mail command. Once mail has notified of any root messages, then entering a character n will read the next mail item and entering a character d will delete the mail item. Correct any problems signified by this mail and then re-issue the above mailx command until the test mail is finally received correctly.

- **mailme2=""**
Defines one or more carbon copy E-mail target recipients of the alert notification.
The E-mail addresses contained in double quotes must be valid reachable targets.
Multiple carbon copy recipient E-mail addresses should be separated by a single space within the confines of the double quote characters. This entry normally represents any additional persons that are required to be informed of the alert notification. If there are no further recipients then leave as the null string "".
It is important that each E-mail address entered here is a valid and reachable target.
Use the same method as above for testing correct delivery of a test E-mail to each of these carbon copy recipients.

Using the roll_log_pseudo Script to Create and Install the Script and Create the Crontab Entry

Once the roll_log_pseudo cron script has been edited with all the user defined data parameters, we need to use this edited script to create the new unique named script (as defined in the scrname= parameter) and create the appropriate crontab entry.

Using the full directory path, the script needs to be run with the install option to complete this task (assuming the roll_log_pseudo cron script was placed into our example /usr/mve_scripts directory). Login as the root user-id and then type:

```
/usr/mve_scripts/roll_log_pseudo install
```

This will create the new script /usr/mve_scripts/roll_pick_log (as per our example) and also create the crontab entry for this cron job.

Issuing the AIX ls -al command in the directory_of_choice, can check for the creation of the desired cron script.

Issuing the AIX crontab -l command, can check for the crontab entry insertion.

The crontab entry can be later removed (uninstalled) by executing the new real script (full directory path) with the uninstall option, if desired. This operation will still leave the real script stored on the system, which can then be used directly to re-install itself.

Using our example that created the new script /usr/mve_scripts/roll_pick_log and the crontab entry for this cron job, we can issue the following command to remove the crontab entry:

```
/usr/mve_scripts/roll_pick_log uninstall
```

Using our example, where we have uninstalled, we can now re-install the same script crontab entry (at a later time) by issuing the following:

```
/usr/mve_scripts/roll_pick_log install
```

Example Emitted E-mail message

From: root@yoursystem.acme.com
Sent: Friday, January 05, 2001 11:15 AM
To: sysadmin@mail.acme.com
Cc: another1@mail.acme.com; another2@mail.acme.com
Subject: RD911 REMOTE AUTO ALERT

TYPE:N123456
FROM:GAL999999
CUST:Acme Co.
SYST:mvEnterprise
DATE:20010105111500

Jan 05 11:02 ** 405bpick 10 null ** Warning..Logger has acquired 6006 overflow..Warning!!

Message auto sent by yoursystem using cron script
/usr/mve_scripts/roll_pick_log

The log content was extracted from /usr/stat/pick_log