UOS RTL Library (LIB$) Manual

Table of contents

Introduction

UOS RTL Library (LIB$) Manual

October 2023

Contents

Preface

Preface

Intended Audience

This manual is intended for application developers writing software for the UOS operating system that call LIB$ library routines.

LIB$

The part of the manual contains descriptions of the LIB$ runtime routines.

Overview of LIB$

Overview of LIB$

This manual describes the Run-Time Library (RTL) LIB$ routines that perform general-purpose functions.

Unless otherwise specified LIB$ routines use 64-bit addresses and 64-bit integer values.

LIB$ADD_RECALL

LIB$ADD_RECALL

Add Command to Recall Buffer

This function adds a command to the command recall buffer. If the added command causes the buffer to exceed the maximum number of commands, the oldest command is deleted from the buffer. Null commands are not added to the buffer.

Format

LIB$ADD_RECALL command

Arguments

command

The address of a TSRB structure that indicates the command string to add.

Required Privileges
None

Affected Quotas
None

Condition Values Returned
No condition code is returned.

LIB$CLEAR_RECALL

LIB$CLEAR_RECALL

Clear Recall Buffer

This function removes all commands from the command recall buffer.

Format
LIB$CLEAR_RECALL

Arguments
None.

Required Privileges
None

Affected Quotas
None

Condition Values Returned
No condition code is returned.

LIB$CVT_FROM_INTERNAL_TIME

LIB$CVT_FROM_INTERNAL_TIME

Converts a timestamp into human-centric values.

Format
LIB$CVT_FROM_INTERNAL_TIME result, operation {, time}

Arguments
operation
A pointer to a 64-bit integer value indicating the conversion to perform, as defined by the following:

Mneumonic	Value	Result range	Description
LIB_K_MONTH_OF_YEAR	0	1 to 12	Month: January=1
LIB_K_DAY_OF_YEAR	1	1 to 366	Day of year
LIB_K_HOUR_OF_YEAR	2	1 to 8,784	Hour of year
LIB_K_MINUTE_OF_YEAR	3	1 to 527,040	Minute of year
LIB_K_SECOND_OF_YE	4	1 to	Second of year

AR		31,622,400	
LIB_K_DAY_OF_MONTH	5	1 to 31	Day of month
LIB_K_HOUR_OF_MONTH	6	1 to 744	Hour of month
LIB_K_MINUTE_OF_MONTH	7	1 to 44,640	Minute of month
LIB_K_SECOND_OF_MONTH	8	1 to 2,678,400	Second of month
LIB_K_DAY_OF_WEEK	9	1 to 7	Day of week: Monday=1
LIB_K_HOUR_OF_WEEK	10	1 to 168	Hours since midnight of previous Monday
LIB_K_MINUTE_OF_WEEK	11	1 to 10,080	Minutes since midnight of previous Monday
LIB_K_SECOND_OF_WEEK	12	1 to 604,800	Seconds since midnight of previous Monday
LIB_K_HOUR_OF_DAY	13	0 to 23	Hour of day
LIB_K_MINUTE_OF_DAY	14	0 to 1,439	Minute of day
LIB_K_SECOND_OF_DAY	15	0 to 86,399	Second of day
LIB_K_MINUTE_OF_HOUR	16	0 to 59	Minute of hour
LIB_K_SECOND_OF_HOUR	17	0 to 3,599	Second of hour
LIB_K_SECOND_OF_MINUTE	18	0 to 59	Second of minute
LIB_K_NANOSECOND_OF_SECOND	19	0 to 999,999,999	Nanosecond of second
LIB_K_JULIAN_DATE	20		Days since 17-Nov-1858
LIB_K_DELTA_WEEKS	21		Number of whole weeks represented by delta time.
LIB_K_DELTA_DAYS	22		Number of whole days represented by delta time
LIB_K_DELTA_HOURS	23		Number of whole hours represented by delta time
LIB_K_DELTA_MINUTES	24		Number of whole minutes represented by delta time
LIB_K_DELTA_SECONDS	25		Number of whole seconds represented by delta time

The last five conversions interpret the time as a delta time; the rest interpret the time as absolute.

result
A pointer to a 64-bit integer to receive the conversion result.

time
A pointer to a 64-bit timestamp. If 0, the current system time is used.

Description

This service returns a number which is the conversion of an absolute or delta time using the specified operation.

Condition Values Returned

Value	Meaning
SS_NORMAL	Successful completion.

LIB_ABSTIM Absolute time required but delta time supplied.
REQ
LIB_INVOPE Invalid operation.
R

LIB$CVTIME

LIB$CVTIME

Convert Time

Format

result = LIB$CVTIM time, format, output, buffer

Arguments

time

The address of a TSRB that points to a valid time specification in ASCII form.

format

How to format the result.

Menumonic	Meaning
CVF_Absolute	Absolute date/time
CVF_Comparison	yyyy-mm-dd hh:mm:ss.cc format
CVF_Delta	Delta format

output

Which items to return.

Menumonic	Meaning
CVO_DateTime	Full date and time
CVO_Date	Full date
CVO_Time	Full time
CVO_Hour	Hour of the day
CVO_Second	Second of minute
CVO_Minute	Minute of hour
CVO_Hundredth	Hundreths of seconds
CVO_Day	Day of the month
CVO_Month	Month of the year
CVO_Weekday	Day of the week
CVO_Year	Year
CVO_DayofYear	Julian day
CVO_HourofYear	Hour of the year
CVO_MinuteofYear	Minute of the year
CVO_SecondofYear	Second of the year

buffer

The address of a TSRB that points to where the result is to be written. The length of the result is written

to the Length field, but never more than the passed length in the structure.

Description
The specified buffer is filled with the requested information in the requested format.

LIB$DAY_OF_WEEK

LIB$DAY_OF_WEEK

Returns the numeric day of the week for a supplied time stamp.

Format
LIB$DAY_OF_WEEK time, result

Arguments
time
A pointer to a 64-bit timestamp. If 0, the current system time is used.

result
A pointer to a 64-bit integer to receive the day number.

Description
This service returns the number of the day of the week corresponding to the passed time. If 0 is passed for the time, the current system time is used. The days are numbered 1 through 7, with Monday having the value 1.

Condition Values Returned
SS_NORMAL Normal completion of service.

LIB$GET_COMMAND

LIB$GET_COMMAND

Get Command from SYS$COMMAND

This function obtains a command from the default command source (SYS$COMMAND). Commands read by this service are added to the command recall buffer.

Format
LIB$GET_RECALL result, prompt, length

Arguments
result
Address of a TSRB structure which points to the buffer to receive the command input.

prompt
Address of a TSRB structure that points to the prompt string.

length
Address of a 64-bit integer that receives the count of bytes read.

Required Privileges
None

Affected Quotas

None

Condition Values Returned

SS$_NORMAL	The service completed successfully.

LIB$Get_Default_File_Protection

LIB$Get_Default_File_Protection
Get default file protection

Returns the default file protection mask for a given process.

Format
LIB$Get_Default_File_Protection pid

Arguments
pid

The process ID of the process whose default file protection mask is to be returned. A value of 0 indicates to return the current process default file protection mask.

Description

This service returns the default file protection mask for the specified process. This mask is a set of 4 fields, each 4 bits, which indicate the protection for newly created files. Each field indicates protection for a given type of user: owner, group, system, and world. The fields are laid out in the following order:

Type	Access Bit	Value (hex)
Owner	Read	1
Owner	Write	2
Owner	Delete/ Control	4
Owner	Execute	8
Group	Read	10
Group	Write	20
Group	Delete/ Control	40
Group	Execute	80
Syste m	Read	100
Syste m	Write	200
Syste m	Delete/ Control	400
Syste m	Execute	800
World	Read	1000
World	Write	2000
World	Delete/ Control	4000
World	Execute	8000

Condition codes returned

Code	Meaning
SS_NONEXPR	Specified process does not exist.
SS_NORMAL	Normal completion of service.

LIB$FAO and LIB$FAOL

LIB$FAO and LIB$FAOL

Format ASCII Output

FAO and FAOL format parameters consisting of strings and integer values, according to directives embedded in a control string. The output is the control string with substitutions made depending upon the embedded directives and the parameter values.

FAO can take up to 17 parameters in the function call. FAO is passed a pointer to an array of parameters.

Format

LIB_FAO control outlen outbuf {p1..p17}
LIB_FAOL control outlen outbuf parameters

Arguments

control

Pointer to SRB that points to the text to be output, together with one or more FAO directives. Each directive begins with an exclamation point (!). To include a literal exclamation point, the !! directive must be used. There is no limit to the size of the string or how many directives it contains. The valid directives are listed below.

outlen

Defines the address of the maximum output buffer size (an int64) on call. On return, the actual size of the data written to the output buffer is written to the address. Note that the output will never exceed the value at the time the function is called.

outbuf

Defines the address of the output buffer. The converted control string is written here.

p1..p17

Up to 17 64-bit integer values that can represent actual data or pointers to string data. There must be one value for each directive in the string. If the string requires more than are supplied, the missing parameters are assumed to be 0. Not all directives require a parameter, and some constructs may require up to three. Extra parameters are ignored. The parameters are processed sequentially as the control string is processed from left to right. If more than 17 parameters are required, use the LIB_FAOL function instead.

parameters

A pointer to an array of 64-bit integer values that can represent actual data or pointers to string data. There must be one value for each directive in the string. If the string requires more than are supplied, the behavior of the function is undefined, but will probably cause an error. Not all directives require a parameter, and some constructs may require up to three. Extra parameters are ignored. The parameters are processed sequentially as the control string is processed from left to right.

Description

FAO converts integer values into binary, octal, decimal, or hexadecimal values, and can insert strings, and

conditionally process directives. See the section below, describing the directives.

FAO Directives
FAO directives can appear anywhere in the control string and have the general form:
!ZZ
where the exclamation point (!) indicates the start of the directive and "ZZ" indicates a 1- or 2-character FAO directive. All alphabetic characters in a directive must be uppercase.

Width
FAO directives optionally can have a width, using this format:
!nZZ
where "n" is the decimal value specifying the width (in characters) for the value substituted for the directive. Example:
!3XB
This would display an integer byte values as hexadecimal (XB) with a width of 3 digits (it is zero-filled on the left).

Repeat
FAO directives optionally can have a repeat count, using this format:
!n(ZZ)
where "n" is the decimal value specifying the number of times that the directive is to be repeated. If the directive requires one or more parameters, sucessive parameters are used for each repetition - the same parameter is not reused for each repetition. Example:
!3(OB)
This would display 3 integer byte values as octal (OB).

Repeat with width
You can specify both a width and a repeat count, using this format:
!n(mZZ)
where "n" is the decimal value specifying the number of times that the directive is to be repeated and "m" is the decimal value specifying the with of the directive output, in characters. Example:
!5(10BB)
This would display five integer byte values as binary (BB), each of which is 10 characters wide.

Variable repeats and widths
You can specify either, or both, a width and a repeat count as variables by using a number sign (#) in place of the decimal value. When such a directive is processed, the next parameter is used in place of the number sign. Example:
!2(#BB)
This would display 2 integer byte values as binary, each of which is a number of characters wide that is defined by the next parameter. Note that even though the directive is repeated, only a single parameter is used for the width - the same width will be used for all iteraions.
!#(OB)
This would display a number of octal values equal to the next parameter.
!#(#OB)
This will read one parameter that will serve as the repeat count, and one more parameter for the width of each octal value output.

Indirect parameters
All string parameters are considered to be addresses of the data. All numeric parameters are assumed to be the actual value. A full 64-bits are required for each parameter value, even if less than 64-bits are required by the directive (the remaining bits are ignored). However, using the indirection symbol (@) in a directive, FAO can be made to treat a parameter as an address that contains the numeric value. Note that only the required number of bytes are read from that address. Example:
!@UQ
In this case, the next parameter is used as an address to a quadword (64-bit) value.

FAO Directives
String Directives:

Directive	Description
!AB	Inserts a string. The parameter is a pointer to a TSRB structure.
!AC	Inserts a string. The parameter is a pointer to a string whose first byte is the length of the string, followed immediately by that many bytes if text.
!AD	Inserts a string, with periods (.) substituted for all nonprintable ASCII codes. Two parameters are required: the first is the length of the string and the second is the address of the string data.
!AF	Inserts a string. Two parameters are required: the first is the length of the string and the second is the address of the string data.
!AS	Inserts a string. The parameter is the address of a string descriptor for a CLASS_S (static) or CLASS_D (dynamic) string.
!AZ	Inserts a string. The parameter is a pointer to a zero-terminated (ASCIZ) string.

Note: All string lengths indicate number of bytes, not number of characters.

Zero-filled Numeric Directives:

Directive	Description
!BB	Convert a byte value to the ASCII representation of that value in base 2. Only the low byte of the parameter is used.
!BW	Convert a word value to the ASCII representation of that value in base 2. Only the lower two bytes of the parameter are used.
!BL	Convert a longword value to the ASCII representation of that value in base 2. Only the lower four bytes of the parameter are used.
!BQ	Convert a quadword value to the ASCII representation of that value in base 2.
!OB	Convert a byte value to the ASCII representation of that value in base 8. Only the low byte of the parameter is used.
!OW	Convert a word value to the ASCII representation of that value in base 8. Only the lower two bytes of the parameter are used.
!OL	Convert a longword value to the ASCII representation of that value in base 8. Only the lower four bytes of the parameter are used.
!OQ	Convert a quadword value to the ASCII representation of that value in base 8.
!OA	Same as !OQ.
!OI	Same as !OL.
!OH	Same as !OQ.
!OJ	Same as !OQ.
!XB	Convert a byte value to the ASCII representation of that value in base 16. Only the low byte of the parameter is used.
!XW	Convert a word value to the ASCII representation of that value in base 16. Only the lower two bytes of the parameter are used.
!XL	Convert a longword value to the ASCII representation of that value in base 16. Only the lower four bytes of the parameter are used.
!XQ	Convert a quadword value to the ASCII representation of that value in base 16.
!XA	Same as !XQ.
!XI	Same as !XL.
!XH	Same as !XQ.
!XJ	Same as !XQ.
!ZB	Convert a byte value to the ASCII representation of that value in base 10. Only the low byte of the parameter is used.
!ZW	Convert a word value to the ASCII representation of that value in base 10. Only the lower two bytes of the parameter is used.
!ZL	Convert a longword value to the ASCII representation of that value in base 10. Only the lower four bytes of the parameter is used.
!ZQ	Convert a quadword value to the ASCII representation of that value in base 10.
!ZA	Same as !ZQ.

!ZI	Same as !ZL.
!ZH	Same as !ZQ.
!ZJ	Same as !ZQ.

Blank-filled Numeric Directives:

Directive	Description
!UB	Convert an unsigned byte value to the ASCII representation of that value in base 10. Only the low byte of the parameter is used.
!UW	Convert an unsigned word value to the ASCII representation of that value in base 10. Only the lower two bytes of the parameter are used.
!UL	Convert an unsigned longword value to the ASCII representation of that value in base 10. Only the lower four bytes of the parameter are used.
!UQ	Convert an unsigned quadword value to the ASCII representation of that value in base 10.
!UA	Same as !UQ.
!UI	Same as !UL.
!UH	Same as !UQ.
!UJ	Same as !UQ.
!SB	Convert a signed byte value to the ASCII representation of that value in base 10. Only the low byte of the parameter is used.
!SW	Convert a signed word value to the ASCII representation of that value in base 10. Only the lower two bytes of the parameter are used.
!SL	Convert a signed longword value to the ASCII representation of that value in base 10. Only the lower four bytes of the parameter are used.
!SQ	Convert a signed quadword value to the ASCII representation of that value in base 10.
!SH	Same as !SL.
!SJ	Same as !SL.

Other Directives:

Directive	Description
!/	Inserts a new line (carriage return and linefeed). It takes no parameters.
!_	Inserts a horizontal tab (ASCII 9). It takes no parameters.
!^	Inserts a form feed. It takes no parameters.
!!	Inserts an exclamation point. It takes no parameters
!%S	Inserts the letter S if the most recently converted numeric value is not 1. If the character before the directive is upper case, an upper case S is inserted, otherwise a lowercase s is inserted.
!%T	Inserts the system time. The parameter is the datetime stamp. If the parameter is 0, the current time is inserted.
!%U	Same as !UQ.
!%I	Converts a UIC to the account name. If an invalid UIC is specified, the directive is treated as !UQ.
!%D	Inserts the system date and time. The parameter is the timestamp. If the parameter is 0, the current date/time is inserted.
!n%C	Conditional. See discussion of conditionals below.
!%E	Else portion of conditional. See discussion of conditionals below.
!%F	End of conditional. See discussion of conditionals below.
!n<	See next directive.
!>	The preceeding directive and this one are used together to define an output field that has a width of n. Within this field are displayed all directives between the !n< and !> directives. The field is blank-filled on the right to make it n characters wide if necessary. All directives within this field are left-justified and blank-filled. Note that these can be nested.
!n*c	Repeats the character c in the output n times.
!-	Reuse the most recently used parameter value.

!+ Skip the next parameter value.

Conditionals
!%nC, !%E, and !%F are used together to insert values depending upon parameter values. This is primarily for use with plurals. The general format is:
!%nCa!%Eb!%F
If n matches the last parameter value, then a is inserted, otherwise b is inserted. Example:
!ZB !%1Cchild!%Echildren!%F
In this example, if the first parameter is 1, the output would be:
1 child
But if the first parameter is not 1, the output would be:
n children
where "n" is the value of the first parameter.

The following table illustrates how the directives interact with width and filling.

Directive Type	Default output width	When explicit width is greater than default	When explicit width is less than default
!BB	8	Right justify and blank fill	Result truncated on left
!BW	16	Right justify and blank fill	Result truncated on left
!BL	32	Right justify and blank fill	Result truncated on left
!BQ	64	Right justify and blank fill	Result truncated on left
!OB	3	Right justify and blank fill	Result truncated on left
!OW	6	Right justify and blank fill	Result truncated on left
!OL	11	Right justify and blank fill	Result truncated on left
!OQ	22	Right justify and blank fill	Result truncated on left
!HB	2	Right justify and blank fill	Result truncated on left
!HW	4	Right justify and blank fill	Result truncated on left
!HL	8	Right justify and blank fill	Result truncated on left
!HQ	16	Right justify and blank fill	Result truncated on left
Unsigned zero-filled decimal	As many characters as are necessary	Right justify and blank fill	Field completely filled with asterisks (*)
Signed or unsigned decimal	As many characters as are necessary	Right justify and zero-filled	

Strings	As many characters as in the string	Left justify and blank fill to specified length	Truncate on right

LIB$GET_FOREIGN

LIB$GET_FOREIGN

Returns the text of the command line that invoked the current program, minus the program's name.

Format
LIB_GET_FOREIGN result {, prompt} {,len} {,flags}

Returns
64-bit integer Status code.

Arguments
result
The address of a SRB structure that defines the location where the command line is to be written, and the maximum size of that location.

prompt
Optional user-supplied prompt for text that LIB_GET_FOREIGN uses if no command-line text is available. This is the address of an SRB structure that defines the prompt text. If this is 0 or the prompt string is null, and there is no command-line text available, a zero-length string is returned.

len
Optional address of where to write the 64-bit length of the returned command line text. This will be the size of the string actually returned.

flags
The address of a 64-bit integer flags value. If provided, and the low bit is set in the flag, the user is prompted unconditionally. Otherwise, the user is only prompted if there is no command-line available. If the prompt is omitted or null, and there is no command-line, a null string is returned.

Description
LIB_GET_FOREIGN returns the contents of the command line that was used to activate the current image, minus the program name. Optionally, the user can be prompted for data if there is no command line text available. The service can be called multiple times to retrieve multiple lines of data. Data returned due to prompting is read from SYS$INPUT. It can be called once to get the command line and then again to get additional parameters from the user.

The command line is set by the shell when it begins execution of a program.

Condition Values Returned

Code	Meaning
SS_NORMAL	Normal completion.
LIB_INPSTRTRU	The result buffer was too small to hold the command-line. Only the characters that fit are returned.

LIB$GET_INPUT

LIB$GET_INPUT

Get Command from SYS$INPUT

This function obtains a command from the default input source (SYS$INPUT). Commands read by this service are added to the command recall buffer.

Format

LIB$GET_RECALL result, prompt, length

Arguments

result

Address of a TSRB structure which points to the buffer to receive the command input.

prompt

Address of a TSRB structure that points to the prompt string.

length

Address of a 64-bit integer that receives the count of bytes read.

Required Privileges

None

Affected Quotas

None

Condition Values Returned

SS$_NORMAL	The service completed successfully.

LIB$GET_RECALL

LIB$GET_RECALL

Get Command from Recall Buffer

This function obtains a command from the command recall buffer.

Format

LIB$GET_RECALL Index, OutLen, OutBuf

Arguments

Index

The offset of the command to recall. 0 returns the oldest command in the buffer, 1 returns the next-to-oldest command, and so on. If negative, it indicates an offset from the end of the command buffer. For instance, -1 returns the most recent (last) command in the buffer, -2 returns the next-most recent command, and so on.

OutLen

Address of a 64-bit integer that indicates the size of the buffer pointed to by OutBuf. On return it receives the actual length of the command, in bytes. If the command is larger than the buffer size, only the specified number of bytes is returned.

OutBuf

Address of a buffer large enough to hold the command.

Required Privileges
None

Affected Quotas
None

Condition Values Returned

SS$_NORMAL	The service completed successfully.

LIB$GET_RECALL_LENGTH

LIB$GET_RECALL_LENGTH
Get Length of Command in Recall Buffer

This function obtains a command from the command recall buffer.

Format
LIB$GET_RECALL_LENGTH Index, OutLen

Arguments
Index
The offset of the command to recall. 0 returns the oldest command in the buffer, 1 returns the next-to-oldest command, and so on. If negative, it indicates an offset from the end of the command buffer. For instance, -1 returns the most recent (last) command in the buffer, -2 returns the next-most recent command, and so on.

OutLen
Address of a 64-bit integer to receive the size of the command at the indicated offset. If the offset is out of range, 0 is written to this address.

Required Privileges
None

Affected Quotas
None

Condition Values Returned

SS$_NORMAL	The service completed successfully.

LIB$POP_RECALL

LIB$POP_RECALL
Pop Command from Recall Buffer

This function deletes the most recent command from the command recall buffer.

Format
LIB$POP_RECALL

Arguments
None.

Required Privileges
None

Affected Quotas
None

Condition Values Returned
None

LIB$PUT_FORMATTED_OUTPUT

LIB$Put_Formatted_Output

Writes HTML formatted output.

This service writes UHTML text to a file, representing it in a way appropriate for the output device. See the Utility Library Reference Manual for a description of UHTML.

Format
LIB$Put_Formatted_Output file output

Parameters
file
A pointer to a UOS file to which the data will be written.

output
A pointer to an SRB that points to the string to write to the file.

Description
Put_Formatted_Output writes UHTML text to the file in such a way to most closely match the UHTML formatting on the output device.

Condition codes returned:

Code	Meaning
SS_BUFFER OVF	indicates that the result was larger than the provided buffer
SS_NORMAL	Successful completion.

LIB$RECALL_COUNT

LIB$RECALL_COUNT

Return Length of Recall Buffer

This function returns the number of commands in the command recall buffer.

Format
LIB$RECALL_COUNT

Arguments
None.

Required Privileges

None

Affected Quotas
None

Condition Values Returned
None

LIB$RUN

LIB$RUN

Executes a program in the context of the current process.

Format
LIB_RUN programname, commandline, flags

Arguments
programname
The address of a SRB that points to the name of the program file to execute. The following rules determine which program is executed if the path and/or extension are omitted:
If a path isn't specified, the program must exist in the user's execution path.
The first instance of the program found in the execution path is the one that is executed, so if programs with the specified name exist in multiple directories in the execution path, you must specify the path in order to execute one that is later in the path.
If a file extension is not provided and multiple matching programs are found in a given directory, the one executed depends upon the current order of execution for file extensions. If a file with a higher extension priority exists in the path after a file with a lower extension priority, the one with the higher extension priority is run.

commandline
The address of a SRB that points to the command line to be passed to the program when execution begins.

flags
The address of a 64-bit value containing the flags that apply to program execution. The flags are:

Flag	Meaning
RUN_ACCNT	Run with accounting, otherwise program quota usage is not applied to user's account. If not set, the user must have the ACNT privilege.
RUN_AUTH	Run with normal authorizations, otherwise run as a logged-out user. If not set, the user must have the IMPERSONATE privilege.
RUN_DEBUG	Run the program in the debugger.
RUN_DUMP	Create a dump file if the program ends abnormally.

Quotas
None for the call, any/all for the running program.

Condition Values Returned

Code	Meaning
SS_NORMAL	Normal completion.
UOSErr_File_Not_Found	No matching program could be found

UOSErr_Infinite_Symbol_Recursion	Extension priorities (sys$extensions symbol) contains an infinite recursion.

LIB$SEEK_FILE

LIB$SEEK_FILE

Set File Position

This service sets the current file position of an open file.

Format

LIB$SEEK_FILE handle, position

Arguments

handle

Handle of file to affect.

position

Byte offset of the new file position.

Required Privileges

None

Affected Quotas

None

Condition Values Returned

SS$_NORMAL	The service completed successfully.

LIB$SPAWN

LIB$SPAWN

Spawns a subprocess.

Format

LIB$SPAWN {command} {,input} {,output} {,flags} {,name} {,pid} {,status} {,eventflag} {,AST} {,ASTP} {,prompt} {,shell} {,reserved}

Arguments

command

A pointer to an SRB which points to the name of the program to execute in the subprocess. If this parameter is 0, the default or specified shell is executed in interactive mode. This string can also contain parameters after the program name in order to pass those parameters to the program.

input

A pointer to an SRB which points to the name of the file to be assigned to sys$input in the subprocess. If this is 0, the subprocess uses the same sys$input as the calling process.

output

A pointer to an SRB which points to the name of the file to be assigned to sys$output in the subprocess. If this is 0, the subprocess uses the same sys$output as the calling process.

flags

A pointer to a 64-bit integer containing options for subprocess creation, as follows. If this parameter is 0, the flags are considered to all be unset.

Mnuemonic	Value	Description
CLI_M_NOWAIT	1	If set, the calling process continues executing asynchronously. Otherwise the calling process is blocked until the subprocess finishes.
CLI_M_NOCLISYM	2	The spawned subprocess does not inherit symbols.
CLI_M_NOLOGNAM	4	The spawned subprocess does not inherit symbols. This has the same meaning as CLI_M_NOCLISYM.
CLI_M_NOKEYPAD	8	The keypad symbols and state are not passed to the subprocess.
CLI_M_NOTIFY	16	A message is broadcast to sys$output when the subprocess completes or aborts.
CLI_M_NOCONTROL	32	No carriage-return/line-feed is prefixed to any prompt string on the subprocess.
CLI_M_TRUSTED	64	Indicates a SPAWN command on behalf of the application. If not set, captive accounts cannot spawn a subprocess
CLI_M_AUTHPRIV	128	The subprocess inherits the caller's authorized privileges.
CLI_M_SUBSYSTEM	256	The spawned process inherits protected subsystem IDs for the duration of the process creation.
CLI_M_NONRANDOM	H10000000	A non-random process name prefix is used. See the name parameter for details.

name

A pointer to an SRB which points to the name to assign to the subprocess. If the specified name is already in use by another process, an error occurs and the subprocess is not created. If this parameter is 0, the name of the process will be the user name, an underscore, and a number. Normally a unique random number is used, but if the CLI_M_NONRANDOM flag is specified, the number will be "1", unless that process name is in use, in which case "2" is tried, and incremented by 1 until a unique process name is generated. For instance, if the user is "System", and there are already existing process names of "System_1", "System_2", and "System_4", then the new process will be named "System_3".

pid

A pointer to a 64-bit integer which receives the PID of the subprocess after it is created. If this parameter is 0, the process ID is not returned.

status

A pointer to a 64-bit integer which will receive the completion status of the subprocess. This is updated asynchronously if the the NOWAIT flag is used. If the subprocess completes without error, the result set to 0. If this parameter is 0, the completion status is not returned.

eventflag

A pointer to a 64-bit integer which contains the event flag to be set when the subprocess completes. If this parameter is 0, no event flag is set.

AST

A pointer to an AST to call when the subprocess completes. If this is 0, no AST routine is called. AST routines should be used if NOWAIT is specified in the flags and the calling process needs to know when the subprocess finishes.

ASTP

A pointer to a 64-bit integer containing the parameter value to pass to the AST routine. This is only meaningful if an AST address is provided.

prompt
A pointer to an SRB which points to the text to be used as a prompt in the shell of the subprocess. Depending upon the shell, this may have no effect.

shell
A pointer to an SRB which points to the name of the shell for the subprocess. If not specified, the default system shell is used.

reserved
This parameter reserved for future use. It is ignored, but 0 should be passed.

Description

This service creates a subprocess. The created subprocess inherits the following attributes from the calling process, although some of these can be modified by the passed flags:
Process symbols
Default device and directory
Process privileges
Process nondeductible quotas

The subprocess does not inherit process-permanent files nor execution or image context.
The set of authorized privileges in the subprocess is inherited from the caller's current privileges. If the calling image is installed with elevated privileges, these privileges are not available to the subprocess until a SETPRV call is performed in the subprocess to enable them. If the calling image is installed with elevated privileges, it should disable those privileges before the call to LIB_SPAWN unless the environment of the subprocess is strictly controlled. Otherwise, elevated privileges may accidentally be made available to the user.

If neither command nor input is specified, command input is taken from the parent process' terminal. If both command and input are specified, the subprocess first executes command and then reads from input. If only command is specified, the program is executed, and the subprocess is terminated when the program exits. If input is specified, the subprocess is terminated by either a LOGOUT procedure or an end-of-file.

The LOGIN utility is not run and no LOGIN.COM file is executed.

Unless the NOWAIT flags bit is set, the caller's process is put into hibernation until the subprocess finishes. Because the caller's process hibernates in supervisor mode, any user-mode ASTs queued for delivery to the caller are not delivered until the caller reawakes. Control can also be restored to the caller by a suitable call to LIB_ATTACH from the subprocess.

Condition Values Returned

SS_NORMAL Normal completion of service.
SS_DUPLNAM A process name was specified, but a process with that name is currently running.
INVARG A flag outside of the valid values was specified.
INVOPER Invalid operation.

LIB$Substitute_Wildcards

LIB$Substitute_Wildcards

Substitute wildcards in a file specification

Returns a file specification substituting wildcards with defaults and values from a non-wildcard specification.

Format

LIB$Substitute_Wildcards source target defaults result length

Arguments

source
The address of an SRB that points to the source file specification.

target
The address of an SRB that points to the target file specification. This is the specification that can contain wildcards. Note that empty fields are implied "*" wildcards.

defaults
The address of an SRB that points to the default file specification. Any fields missing (null) in the target specification are filled from this specification before the wildcard substitution is done.

result
The address of an SRB that points to the buffer to receive the resulting file specification. The length indicates the maximum size of the buffer, in bytes.

length
The address of a 64-bit integer to receive the actual size of the result. If 0, no size is returned.

Description
This service constructs a result file specification from a source specification, a target (containing wildcards) and a default specification. The default specification is first used to fill any null fields in the target. Wildcard fields are not filled from the default. Next the source specification is used to fill any remaining null fields or wildcards in the target. The result is returned to the specificed buffer. If the buffer is too small to contain the result, as much as will fit is returned and the service returns a SS_BUFFEROVF condition. Note that null fields that are not specified in any of the three specifications will be null in the result. This service does not perform any validation of the file specifications' syntax nor does it check to see if the resulting specification refers to an existing file.

Condition codes returned

Code	Meaning
SS_BUFFEROVF	Result buffer was not large enough to fit the entire result specification.
SS_NORMAL	Normal completion of service.

LIB$SYS_ASCTIM

LIB$SYS_ASCTIM
Convert Binary Time to ASCII

This function converts a 64-bit internal timestamp to an ASCII string.

Format
LIB$SYS_ASCTIM length, result, timestamp, flags

Arguments
length
Address of where to write the length of the result, in bytes.

result
Address of the SRB structure indicating the address of the buffer to receive the ASCII time.

timestamp
Internal 64-bit UOS timestamp. A negative value indicates a delta time. Otherwise it is considered to

be an absolute time. If this is 0, the current date and time is returned.

flags
Conversion indicator specifying what values to return.

Value	Meaning
0	Date and time
1	Time only
2	Date only

Required Privileges
None

Affected Quotas
None

Condition Values Returned

SS$_NORMAL	The service completed successfully.
SS$_ACCVIO	The address to receive the time cannot be written to.

LIB$SYS_FILESCAN

LIB$SYS_FILESCAN
Scan File Specification

This function searches a string for a file specification and parses it into its fields.

Format
LIB$SYS_FILESCAN filespec list fldflags auxout retlen

Arguments
filespec
String to be searched. This is the address of an SRB structure that points to the string.

list
Item list specifying which components of the file specification are to be returned. This argument is the address of the first descriptor in the list. The last descriptor in the list must be all 0's.

fldflgs
The address of where a bitmask is written that indicates which fields of the file specification were specified. If this value is 0, this is ignored. The fields are indicated by the following flag values:

Symbol name	Description
FSCN_V_DEVICE	Device name.
FSCN_V_DIRECTORY	Directory name.
FSCN_V_NAME	File name.
FSCN_V_NODE	Node name.
FSCN_V_NODE_ACS	Access control string of primary node.
FSCN_V_NODE_PRIMARY	Primary (first) node name
FSCN_V_NODE_SECONDARY	Secondary (additional) node information
FSCN_V_ROOT	Root directory name string
FSCN_V_TYPE	File type
FSCN_V_VERSION	Version number

auxout
Auxillary output buffer. This argument is the address of an SRB structure which indicates where the complete file specification (as provided) is written. Any secondary node information is stripped from the output and quotations are reduced and simplified.
If this value is 0, it is ignored. If provided, the values written to the item list are addresses within this auxillary buffer.

retlen
Auxillary output buffer length. This is the address of an 8-byte integer where the length of the auxillary output buffer is written. If this is 0, no length is written.

Descriptors

Byte offset	Byte length	Description
0	4	Item code
4	4	Length
8	8	Address

Description
The FILESCAN service searches a string for a file specification and parses the fields of that specification. The length and starting addresses of the fields requested are returned. If a field was requested in the item list but not found in the file specification, a length and address of 0 are written to the descriptor. The descriptor list is terminated with a descriptor that has an item code of 0.

The information returned describes the entire contiguous file specification. For example, to extract only the file name and type from the full string, you can use the address of the file name, for the length of the sum of the name and type to obtain the full file name. However, FSCN_NODE_PRIMARY and FSCN_NODE_ACS items contain no double colon (::), so you would have to add 2 to the sum of the lengths of those two fields to obtain the entire node specification.

FILESCAN does not check all aspects of validity in the specification. For instance, it does not verify that the node name specified corresponds to a valid node. Nor does it validate the access control string contents. Nor does it verify the existence of the path or specified file. It treats wildcard characters as any other valid character. It doesn't validate lengths either. Finally, multiple whitespace characters are not collapsed to a single space, nor trimmed from the beginning or end of the string. However, spaces, tabs, and delimiting characters must be enclosed in quotes if they are part of the file name or type, otherwise the character is treated as a terminator for the specification. Quotes used to indicate a node access control string require that the node name be enclosed in quotes and thus the quotes delimiting the access control string must be doubled (""). For example, the node specification:
abcd"efg"
would need to be specified as:
"abcd""efg"""
FILESCAN does not assume default values for missing fields or perform logical name translations.

Here are the item codes that can be used in the passed descriptors:

Code	Description
FSCN_DEVICE	Returns length and starting address of the device name, including the colon (:).
FSCN_DIRECTORY	Returns the length and starting address of the path, including all backslashes (\).
FSCN_FILESPEC	Returns the length and starting address of the full file specification.
FSCN_NAME	Returns the length and starting address of the file name, including no syntactical elements.
FSCN_NODE	Returns the length and starting address of the node, access control string, and double colon (::).

FSCN_NODE_ACS	Returns the length and starting address of the node access control string.
FSCN_NODE_PRIMARY	Returns the length and starting address of the primary node name. It doesn't include the double colon (::) or access control string.
FSCN_NODE_SECONDARY	Returns the length and starting address of the secondary node string.
FSCN_ROOT	Returns the length and starting address of the root directory of the path, including backslashes (\).
FSCN_TYPE	Returns the length and starting address of the file type, including the leading dot (.).
FSCN_VERSION	Returns the length and starting address of the version, including the leading semicolon (;).

Required Privileges
None

Affected Quotas
None

Condition Values Returned

SS$_NORMAL	The service completed successfully.
SS$_ACCVIO	The address to receive the time cannot be written to.

LIB$SYS_PARSE

LIB$SYS_PARSE
Parse a File Specification

The PARSE service parses a file specification string and fills in various NAML fields.

Format
LIB$SYS_PARSE fab err suc

Returns
The result of the operation is stored in the FAB_L_STS item in the FAB structure.

Arguments
fab
Pointer to a FAB block whose contents are to be used as arguments for the PARSE call.

err
Address of a user-written routine to be called if there was an error. If 0, no routine is called. The called routine is assumed to take no parameters and return void.

suc
Address of a user-written routine to be called if there were no errors. If 0, no routine is called. The called routine is assumed to take no parameters and return void.

Description
This function is automatically called as part of the OPEN, CREATE, and ERASE services. It is also used to prepare the FAB and NAML blocks for use in the SEARCH service. The following FAB and NAML fields are potentially read and/or written by this service.

Block	Field	R/W	Description
FAB	FAB_L_DNA	Read	Default file specification string.
FAB	FAB_L_DNS	Read	Default file specification string length, in bytes.

FAB	FAB_B_FNA	Read	File specification string address.
FAB	FAB_L_FNS	Read	File specification string address length, in bytes.
FAB	FAB_L_NAM	Read	Address of NAML block.
FAB	FAB_L_STS	Write	Completion status code.
NAML	NAML_B_NOP	Read	Processing flags. If the NAML_V_SYNCHK flag is set, only a syntax check is performed. Otherwise, the node, device, and path are checked for validity.
NAML	NAML_L_LONG_EXPAND	Read	Address of output expanded string value.
NAML	NAML_L_LONG_EXPAND_ALLOC	Read	Maximum size of expanded output buffer.
NAML	NAML_L_LONG_EXPAND_SIZE	Write	Length of output expanded string value.
NAML	NAML_L_LONG_DEFNAME	Read	Address of default file specification. If FAB.FAB_L_DNS is -1, this is used as the default.
NAML	NAML_L_LONG_DEFNAME_SIZE	Read	Length of default file specification.
NAML	NAML_L_LONG_FILENAME	Read	Address of file specification. If FAB.FAB_L_FNA is -1, this is used as the file specification.
NAML	NAML_L_LONG_FILENAME_SIZE	Read	Length of file specification.
NAML	NAML_L_LONG_DEV	Write	Address of device name, or 0 if none.
NAML	NAML_L_LONG_DEV_SIZE	Write	Length of device name.
NAML	NAML_L_LONG_DIR	Write	Address of the path, or 0 if none.
NAML	NAML_L_LONG_DIR_SIZE	Write	Length of the path specification.
NAML	NAML_L_LONG_NAME	Write	Address of the name portion of the file name, or 0 if none.
NAML	NAML_L_LONG_NAME_SIZE	Write	Length of the name portion of the file name.
NAML	NAML_L_LONG_NODE	Write	Address of the node name, or 0 if none.
NAML	NAML_L_LONG_NODE_SIZE	Write	Length of the node name.
NAML	NAML_L_LONG_TYPE	Write	Address of the type portion of the file name, or 0 if none.
NAML	NAML_L_LONG_TYPE_SIZE	Write	Length of the type portion of the file name.
NAML	NAML_L_LONG_VER	Write	Address of the version portion of the file name, or 0 if none.
NAML	NAML_L_LONG_VER_SIZE	Write	Length of the version portion of the file name.
NAML	NAML_L_LONG_RESULT_SIZE	Write	Set to 0.
NAML	NAML_W_FID	Write	Set to 0.
NAML	NAML_L_FNB	Write	Filename flags.

Condition Codes

The following condition values can be returned:

RMS_FAB	FAB block has invalid format.
RMS_BLN	FAB or NAM block have invalid length(s).
RMS_DNF	Directory not found.

LIB$SYS_GETMSG

LIB$SYS_GETMSG

Get System Message

This function obtains the text of a message from the system messages file.

Format

LIB$SYS_GETMSG msg, reslen, res, flags, outadr

Arguments

msg

Address of a 64-bit integer containing the ID of the message to retrieve.

reslen

Address of a 64-bit integer where the length of the retrieved text is written.

res

Address of a TSRB structure that points to the buffer to receive the message text.

flags

Flags indicating what text to return.

Bit	Meaning
1	Include text of message
2	Include message identifier
4	Include severity indicator
8	Include facility name

outadr

Reserved for future use. Address of a 64-bit integer.

Required Privileges

None

Affected Quotas

None

Condition Values Returned

SS$_NORMAL	The service completed successfully.