SysWorks

Callable Routines Reference Manual

Order Number SWRK-RRM-35

August 2009

This manual provides reference information about the various callable routine interfaces to SysWorks™.

Revision/Update Information:	This manual supercedes the SysWorks™ Callable Routines Reference Manuual V3.4-1
Operating System:	OpenVMS VAX V7.2 or higher; OpenVMS Alpha V7.2 or higher; DECwindows/Motif V1.2-3 or higher
Software Version:	SysWorks™ V3.5

Corpita Pty Ltd

Printed in Australia

This document was prepared using DECdocument V3.3.

Contents

Part III SWRKDCL_ Reference Section

Part IV Macros Reference Section

Index

Send Us Your Comments

We welcome your comments on this manual or any SysWorks manual. If you have suggestions for improvements or find any errors, please indicate the chapter, section and page number (if available). Your input is valuable in improving future releases of our documentation.

You can send comments to us in the following ways:

- Email—`http://www.sysworks.com.au/contact.php`
- Phone—`+61 (03) 9411 4411`
- FAX —`+61 (03) 9411 4499`
- Postal service

```
SysWorks
Corpita Pty Ltd
15 Bedford Street
Collingwood VIC 3066
Australia
```

Preface

This manual provides reference information about the various callable routine interfaces to SysWorks™.

Intended Audience

This manual is intended for SysWorks™ application developers who have a working knowledge of the underlying Digital products.

Conventions

The following conventions are used in this document:

Conventions	Description
Ctrl/X	A sequence such as Ctrl/X indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
[ ]	In format descriptions, brackets indicate that whatever is enclosed is optional; you can select none, one or all of the choices.
	In system prompts indicates the default value which will be assumed if the Return key is pressed without first entering a value.
{}	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.
|	In format descriptions, vertical bars separate the options. If the options are enclosed in brackets (i.e. [ ]) you can select none, one or all of the choices. If the options are enclosed in braces (i.e. {}) you must choose one of the options listed.
( )	In system prompts, parenthesis indicate the list of values one of which may be entered. The values are separated by a forward slash "/"
...	An elipsis indicates that a value within a range may be chosen or a syntax repeated. A range may be indicated by a pair of end values, or an end value and an end keyword. For example `Disk quota (0..unlimited)` indicates that the keyword `unlimited` may be used to represent the highest possible disk quota.
italic text	Italicized words and letters indicate that you should substitute a word or value of your choice.
UPPERCASE TEXT	Uppercase letters indicate the name of a command or routine.
`monospace text`	Normal monospace text indicates system prompts and output.
`bold monospace text`	Bold monospace text indicates user responses to system prompts.
`bold monospace italic text`	Bold monospace italic text indicates user responses to system prompts which need approriate value substitution.

Conventions	Description
mouse	The term *mouse* is used to refer to any pointing device such as a mouse, a puck or a stylus.
MB1, MB2, MB3	MB1 indicates the left mouse button, MB2 indicates the middle mouse button, and MB3 indicates the right mouse button. (The buttons can be redefined by the user.)

Unless otherwise noted, all numeric values are represented in decimal notation.

Part I

General Information

This section contains general information about SysWorks™ programming interfaces.

1

Callable Routines Reference

This chapter provides the reference information for the callable routine interfaces provides by SysWorks™.

Part II

SWRK_ Reference Section

This section contains detailed discussions about routines provided by the SysWorks SWRKSHR runtime library.

SWRK_ADD_OBJECT Template

The SWRK_ADD_OBJECT routine ...

Format

status=*SWRK_ADD_OBJECT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_ADD_OBJECT routine ...

See also

SWRK_ALLOCATE_BITS Template

The SWRK_ALLOCATE_BITS routine ...

Format

status=*SWRK_ALLOCATE_BITS(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_ALLOCATE_BITS routine ...

See also

SWRK_ALLOCATE_MEMORY Template

The SWRK_ALLOCATE_MEMORY routine ...

Format

status=*SWRK_ALLOCATE_MEMORY(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_ALLOCATE_MEMORY routine ...

See also

SWRK_APPEND_ITEM_TO_LNKLST Template

The SWRK_APPEND_ITEM_TO_LNKLST routine ...

Format

status=*SWRK_APPEND_ITEM_TO_LNKLST(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_APPEND_ITEM_TO_LNKLST routine ...

See also

SWRK_APPEND_LNKLST_TO_LNKLST Template

The SWRK_APPEND_LNKLST_TO_LNKLST routine ...

Format

status=*SWRK_APPEND_LNKLST_TO_LNKLST(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_APPEND_LNKLST_TO_LNKLST routine ...

See also

SWRK_APPEND_STRING_TO_LNKLST Template

The SWRK_APPEND_STRING_TO_LNKLST routine ...

Format

status=*SWRK_APPEND_STRING_TO_LNKLST(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_APPEND_STRING_TO_LNKLST routine ...

See also

SWRK_BACKUP_FILE Template

The SWRK_BACKUP_FILE routine ...

Format

status=*SWRK_BACKUP_FILE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_BACKUP_FILE routine ...

See also

SWRK_BINARY_SEARCH
Search an ordered keyword list

The SWRK_BINARY_SEARCH routine searches an ordered array of keywords.

Format

status=*SWRK_BINARY_SEARCH(index, num, table, keyword)*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

index

VMS Usage: **index**
type: **integer (unsigned)**
access: **write only**
mechanism: **by reference**

Specifies the address of the returned index of the matching keyword. If there was no match, this value is undefined.

num

VMS Usage: **length**
type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

Specifies the number of keywords (or rows) in the keyword table array.

table

VMS Usage: **structure**
type: **array**
access: **read only**
mechanism: **by reference**

Specifies the table of keywords. Each element consists of three longwords. The first is the length of the keyword. The second is the address of the keyword. The third is the keyword value.

keyword

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the keyword which will be searched for.

Description

The SWRK_BINARY_SEARCH routine searches for the keyword in the ordered keyword table using a binary search. It allows the use of unambiguous abbreviations.

Return Values

SWRK__ABRKEYVAL	Keyword was found and was an unambiguous abbreviation. The index argument received the index of the keyword within the table.
SWRK__AMBKEYVAL	Keyword was found but was an ambiguous abbreviation.
SWRK__EXAKEYVAL	Keyword was found as an exact match. The index argument received the index of the keyword within the table.
SWRK__UNKKEYVAL	Keyword was not found.

SWRK_BITMAP_OPERATION_2 Template

The SWRK_BITMAP_OPERATION_2 routine ...

Format

status=*SWRK_BITMAP_OPERATION_2(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_BITMAP_OPERATION_2 routine ...

See also

SWRK_BITMAP_OPERATION_3 Template

The SWRK_BITMAP_OPERATION_3 routine ...

Format

status=*SWRK_BITMAP_OPERATION_3(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_BITMAP_OPERATION_3 routine ...

See also

SWRK_BUILD_INFO_RECORD Template

The SWRK_BUILD_INFO_RECORD routine ...

Format

status=*SWRK_BUILD_INFO_RECORD(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_BUILD_INFO_RECORD routine ...

See also

SWRK_CALL_REMOTE Template

The SWRK_CALL_REMOTE routine ...

Format

status=*SWRK_CALL_REMOTE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CALL_REMOTE routine ...

See also

SWRK_CANCEL_JOB Template

The SWRK_CANCEL_JOB routine ...

Format

status=*SWRK_CANCEL_JOB(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CANCEL_JOB routine ...

See also

SWRK_CANCEL_TIMER Template

The SWRK_CANCEL_TIMER routine ...

Format

status=*SWRK_CANCEL_TIMER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CANCEL_TIMER routine ...

See also

SWRK_CHECK_RESOURCE Template

The SWRK_CHECK_RESOURCE routine ...

Format

status=*SWRK_CHECK_RESOURCE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CHECK_RESOURCE routine ...

See also

SWRK_CLEAR_CONTEXT Template

The SWRK_CLEAR_CONTEXT routine ...

Format

status=*SWRK_CLEAR_CONTEXT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CLEAR_CONTEXT routine ...

See also

SWRK_CLOSE_AND_DETACH_JOB Template

The SWRK_CLOSE_AND_DETACH_JOB routine ...

Format

status=*SWRK_CLOSE_AND_DETACH_JOB(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CLOSE_AND_DETACH_JOB routine ...

See also

SWRK_CLOSE_AND_SPAWN_JOB Template

The SWRK_CLOSE_AND_SPAWN_JOB routine ...

Format

status=*SWRK_CLOSE_AND_SPAWN_JOB(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CLOSE_AND_SPAWN_JOB routine ...

See also

SWRK_CLOSE_AND_SUBMIT_JOB Template

The SWRK_CLOSE_AND_SUBMIT_JOB routine ...

Format

status=*SWRK_CLOSE_AND_SUBMIT_JOB(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CLOSE_AND_SUBMIT_JOB routine ...

See also

SWRK_CLOSE_FILE Template

The SWRK_CLOSE_FILE routine ...

Format

status=*SWRK_CLOSE_FILE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CLOSE_FILE routine ...

See also

SWRK_CLOSE_INPUT Template

The SWRK_CLOSE_INPUT routine ...

Format

status=*SWRK_CLOSE_INPUT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CLOSE_INPUT routine ...

See also

SWRK_CLOSE_OUTPUT Template

The SWRK_CLOSE_OUTPUT routine ...

Format

status=*SWRK_CLOSE_OUTPUT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CLOSE_OUTPUT routine ...

See also

SWRK_COMMIT Template

The SWRK_COMMIT routine ...

Format

status=*SWRK_COMMIT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_COMMIT routine ...

See also

SWRK_CONVERT_DATE Template

The SWRK_CONVERT_DATE routine ...

Format

status=*SWRK_CONVERT_DATE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CONVERT_DATE routine ...

See also

SWRK_CONVERT_DATE_TIME Template

The SWRK_CONVERT_DATE_TIME routine ...

Format

status=*SWRK_CONVERT_DATE_TIME(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CONVERT_DATE_TIME routine ...

See also

SWRK_CONVERT_TIME Template

The SWRK_CONVERT_TIME routine ...

Format

status=*SWRK_CONVERT_TIME(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CONVERT_TIME routine ...

See also

SWRK_COPY_FILE Template

The SWRK_COPY_FILE routine ...

Format

status=*SWRK_COPY_FILE(src-file-spec, trg-file-spec [,dflt-file-spec] [,rltd-file-spec] [,flags] [,user-success-clb] [,user-error-clb] [,user-confirm-clb] [,user-arg] [,src-rslt-file-spec] [,trg-rslt-file-spec] [,context] [,extended-flags] [,ufsa-type] [,log-flags])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

src-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

File specification of the files to be copied. The src-file-spec argument is the address of a descriptor pointing to the source file specification. The specification may include wildcards, in which case each file that matches the specification will be copied. The string must not contain more than 255 characters. Any string class is supported.

trg-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

File specification for the target file names. The trg-file-spec argument is the address of a descriptor pointing to the target file specification.

This specification need not be complete; fields omitted or specified by using the wildcard character (*) will be filled in from the existing file's name using the same rules as for the equivalent DCL command. The string must not contain more than 255 characters. Any string class is supported. [element-or-template]...

dflt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**

mechanism: **by descriptor**

Default file specification of the files to be copied. The dflt-file-spec argument is the address of a descriptor pointing to the default file specification. This is an optional argument; if the argument is omitted, the default is the null string. Any string class is supported.

See the OpenVMS Record Management Services Reference Manual for information about default file specifications.

rltd-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Related file specification of the files to be copied. The rltd-file-spec argument is the address of a descriptor pointing to the related file specification. Any string class is supported. This is an optional argument; if the argument is omitted, the default is the null string.

Input file parsing is used. See the OpenVMS Record Management Services Reference Manual for information on related file specifications and input file parsing.

The related file specification is useful when you are processing lists of file specifications. Unspecified portions of the file specification are inherited from the last file processed.

flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating optional behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

user-success-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied success routine that is called after a file is successfully copied.

The success routine can be used to display a log of the files that were copied.

user-error-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied error routine that is called when an error is detected.

The error routine returns a success/fail value that is used to determine if more files should be copied.

user-confirm-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied confirm routine that is called before each file is copied. The value returned by the confirm routine determines whether or not the file will be copied. The confirm routine can be used to select specific files for copy based on criteria such as expiration date, size, and so on.

user-arg

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

User supplied argument that is passed to the error, success, and confirm routines each time they are called. Whatever mechanism is used to pass user-arg is also used to pass it to the routines. This is an optional argument; if the argument is omitted, zero is passed by value.

src-rslt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

String into which the source resultant file specification of the last file processed is copied. This is an optional argument. If present, it is used to store the file specification passed to the user supplied routines instead of a default class S, type T string. Any string class is supported.

If you are specifying one or more of the action routine arguments, be sure that the descriptor class used to pass the resultant name is the same as the descriptor class required by the action routine. For example, VAX Ada requires a class SB descriptor for string arguments to Ada routines, but will use a class A descriptor by default when calling external routines.

Refer to your language manual to determine the proper descriptor class to use.

trg-rslt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

String into which target OpenVMS RMS resultant file specification of the last file processed is written. The trg-rslt-file-spec argument is the address of a descriptor pointing to the target name. This is an optional argument. If present, it is used to store the file specification passed to the user supplied routines instead of a class S, type T string. Any string class is supported.

If you are specifying one or more of the action routine arguments, be sure that the descriptor class used to pass the resultant name is the same as the descriptor class required by the action routine. For example, VAX Ada requires a class SB descriptor for string arguments to Ada routines, but will use a class A descriptor by default when calling external routines. Refer to your language manual to determine the proper descriptor class to use.

context

VMS Usage: **context**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

A longword integer variable into which the routine stores a context value for use by future calls. The context argument is an unsigned longword integer containing the address of the context. This variable must be set to zero before the first call. You can use the same context argument from one call to another. This argument is used to retain the context when processing multiple input files. Portions of file specifications that the user does not specify may be inherited from the last files processed because the file contexts are retained in this argument. You must not change the value of context in subsequent calls.

extended-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating extended behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Flag	Usage
BINARY	Force binary file types into binary file format.
CONFIRM	A confirmation is required before any action is performed on a file.
COPY	.
DELETE	.
IGNORE_ DIRECTORIES	.
HEADER	.
MULTIPLE	.
NOSEARCHLIST	
NO_RLF	.

Flag	Usage
PRT	.
QUALIFIED	The extended file search qualifiers are to be used.
RETAIN_CDT	.
RETAIN_RDT	.
RECORD	Use record mode rather than block mode.
SEQUENTIAL	.
UPDATE	Open files with update allowed.

The flag values have symbolic names of the form UFSA$M_*flag* and UFSA$V_*flag*.

ufsa-type

VMS Usage: **code**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies the structure of the source file specification.

The following table lists the types of file specification argument which can be passed.

Type	Structure
INDEX	A binary index passed by reference (not implemented yet).
LNKLST	Linked list of strings passed by reference.
STRING	String passed by descriptor.

The type values have a symbolic name of the form UFSA$C_*type*.

By default, STRING is assumed.

log-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating logging behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Description

The SWRK_COPY_FILE routine ...

See also

SWRK_COPY_MSG_VEC Template

The SWRK_COPY_MSG_VEC routine ...

Format

status=*SWRK_COPY_MSG_VEC(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_COPY_MSG_VEC routine ...

See also

SWRK_COPY_STRING Template

The SWRK_COPY_STRING routine ...

Format

status=*SWRK_COPY_STRING(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_COPY_STRING routine ...

See also

SWRK_CRASH_IMAGE Template

The SWRK_CRASH_IMAGE routine ...

Format

status=*SWRK_CRASH_IMAGE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CRASH_IMAGE routine ...

See also

SWRK_CREATE_BLOCK Template

The SWRK_CREATE_BLOCK routine ...

Format

status=*SWRK_CREATE_BLOCK(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CREATE_BLOCK routine ...

See also

SWRK_CREATE_FILE_FDL Template

The SWRK_CREATE_FILE_FDL routine ...

Format

status=*SWRK_CREATE_FILE_FDL(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CREATE_FILE_FDL routine ...

See also

SWRK_CREATE_FILE Template

The SWRK_CREATE_FILE routine ...

Format

status=*SWRK_CREATE_FILE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CREATE_FILE routine ...

See also

SWRK_CVT_CONTEXT_COMMAND Template

The SWRK_CVT_CONTEXT_COMMAND routine ...

Format

status=*SWRK_CVT_CONTEXT_COMMAND(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CVT_CONTEXT_COMMAND routine ...

See also

SWRK_CVT_CONTEXT_DESCRIPTION Template

The SWRK_CVT_CONTEXT_DESCRIPTION routine ...

Format

status=*SWRK_CVT_CONTEXT_DESCRIPTION(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CVT_CONTEXT_DESCRIPTION routine ...

See also

SWRK_CVT_CONTEXT_TYPE Template

The SWRK_CVT_CONTEXT_TYPE routine ...

Format

status=*SWRK_CVT_CONTEXT_TYPE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CVT_CONTEXT_TYPE routine ...

See also

SWRK_CVT_INDEX_TO_STRING Template

The SWRK_CVT_INDEX_TO_STRING routine ...

Format

status=*SWRK_CVT_INDEX_TO_STRING(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CVT_INDEX_TO_STRING routine ...

See also

SWRK_CVT_LNKLST_TO_STRING Template

The SWRK_CVT_LNKLST_TO_STRING routine ...

Format

status=*SWRK_CVT_LNKLST_TO_STRING(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CVT_LNKLST_TO_STRING routine ...

See also

SWRK_CVT_SCOPE_TYPE_TO_CODE Template

The SWRK_CVT_SCOPE_TYPE_TO_CODE routine ...

Format

status=*SWRK_CVT_SCOPE_TYPE_TO_CODE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CVT_SCOPE_TYPE_TO_CODE routine ...

See also

SWRK_CVT_SCOPE_TYPE_TO_CTL Template

The SWRK_CVT_SCOPE_TYPE_TO_CTL routine ...

Format

status=*SWRK_CVT_SCOPE_TYPE_TO_CTL(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CVT_SCOPE_TYPE_TO_CTL routine ...

See also

SWRK_CVT_TRACE_DESCRIPTION Template

The SWRK_CVT_TRACE_DESCRIPTION routine ...

Format

status=*SWRK_CVT_TRACE_DESCRIPTION(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_CVT_TRACE_DESCRIPTION routine ...

See also

SWRK_DCL_HANDLER
DCL Interface Handler

The DCL Interface Handler routine initializes a DCL utility in either single command or multi command mode. In single command mode the command present on the command line is parsed and executed. In multi line mode, the utility is passed no command line and prompts the user for a number of commands until an EXIT command is used.

Format

status=**SWRK_DCL_HANDLER***(tables, utlnam, [prompt], [ctlflg], [cmdlin])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

tables

VMS Usage: **command_tables**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies the address of the DCL command tables. Generally this is the value associated with the name of the Command Definition Utility module.

utlnam

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the name of the utility. This name is used to lookup the first level of help (when SWRKDCL_HELP is used), and to be part of the default prompt used when the **prompt** argument is not used.

prompt

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the prompt to be used in multi line mode or when extra input is required. By default, the utility name specified in **utlnam** with a "> " appended is used. When more input is required, a leading underscore is used.

ctlflg

VMS Usage: **flags**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Flags specifying options for the SWRK_DCL_HANDLER operation. The **ctlflg** argument is a longword bit mask that is the logical OR of each bit set, where each bit corresponds to an option. The $DUIDEF macro defines a symbolic name for each flag bit. The following table describes each flag.

Flag	Description
DUI_M_SMG_INPUT	When this flag is specified, SMG$ routines are used to manage input, otherwise direct RMS calls are used. Using this feature allows use of a recall buffer.
DUI_M_SMG_OUTPUT	When this flag is specified, SMG$ routines are used to manage output, otherwise direct RMS calls are used. Using this feature allows advantage to be taken of using a video terminal with help etc.
DUI_M_TPU	When this flag is specified, TPU routines are used for input and output.
DUI_M_QUOTE_PLUS	When this flag is specified, quotation marks are placed around qualifier values containing a plus signs. This is useful for providing a front end for CMS.

cmdlin

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies a command line to be parsed and executed in place of the DCL command line.

Description

The SWRK_DCL_HANDLER routine provides a standard interface for a DCL utility.

See also

SWRKDCL_EXIT
SWRKDCL_HELP

SWRK_DEALLOCATE_BITS Template

The SWRK_DEALLOCATE_BITS routine ...

Format

status=*SWRK_DEALLOCATE_BITS(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DEALLOCATE_BITS routine ...

See also

SWRK_DEALLOCATE_MEMORY Template

The SWRK_DEALLOCATE_MEMORY routine ...

Format

status=*SWRK_DEALLOCATE_MEMORY(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DEALLOCATE_MEMORY routine ...

See also

SWRK_DECLARE_BITMAP Template

The SWRK_DECLARE_BITMAP routine ...

Format

status=*SWRK_DECLARE_BITMAP(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DECLARE_BITMAP routine ...

See also

SWRK_DECLARE_CLIENT Template

The SWRK_DECLARE_CLIENT routine ...

Format

status=*SWRK_DECLARE_CLIENT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DECLARE_CLIENT routine ...

See also

SWRK_DECLARE_COUNTER Template

The SWRK_DECLARE_COUNTER routine ...

Format

status=*SWRK_DECLARE_COUNTER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DECLARE_COUNTER routine ...

See also

SWRK_DECLARE_SERVER Template

The SWRK_DECLARE_SERVER routine ...

Format

status=*SWRK_DECLARE_SERVER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DECLARE_SERVER routine ...

See also

SWRK_DECLARE_STATISTICS Template

The SWRK_DECLARE_STATISTICS routine ...

Format

status=*SWRK_DECLARE_STATISTICS(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DECLARE_STATISTICS routine ...

See also

SWRK_DELETE_BLOCK Template

The SWRK_DELETE_BLOCK routine ...

Format

status=*SWRK_DELETE_BLOCK(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DELETE_BLOCK routine ...

See also

SWRK_DELETE_FILE Template

The SWRK_DELETE_FILE routine ...

Format

status=*SWRK_DELETE_FILE(file-spec [,dflt-file-spec] [,rltd-file-spec] [,user-success-clb] [,user-error-clb] [,user-confirm-clb] [,user-arg] [,rslt-file-spec] [,context] [,extended-flags] [,ufsa-type] [,log-flags])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

String containing the OpenVMS Record Management Services (RMS) file specification of the files to be deleted. The file-spec argument is the address of a descriptor pointing to the file specification. If the specification includes wildcards, each file that matches the specification is deleted. The string must not contain more than 255 characters. Any string class is supported.

On Alpha systems, set the LIB$M_FIL_LONG_NAMES bit in the flags argument for strings longer than 255 characters in length.

dflt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Default file specification of the files to be deleted. The dflt-file-spec argument is the address of a descriptor pointing to the default file specification. This is an optional argument; if the argument is omitted, the default is the null string. Any string class is supported.

See the OpenVMS Record Management Services Reference Manual for information about default file specifications.

rltd-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Related file specification of the files to be deleted. The rltd-file-spec argument is the address of a descriptor pointing to the related file specification. Any string class is supported. This is an optional argument; if the argument is omitted, the default is the null string.

Input file parsing is used. See the OpenVMS Record Management Services Reference Manual for information on related file specifications and input file parsing.

The related file specification is useful when you are processing lists of file specifications. Unspecified portions of the file specification are inherited from the last file processed.

user-success-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied success routine that is called after a file is successfully deleted.

The success routine can be used to display a log of the files that were deleted.

user-error-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied error routine that is called when an error is detected.

The error routine returns a success/fail value that is used to determine if more files should be deleted.

user-confirm-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied confirm routine that is called before each file is deleted. The value returned by the confirm routine determines whether or not the file will be deleted. The confirm routine can be used to select specific files for deletion based on criteria such as expiration date, size, and so on.

user-arg

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**

mechanism: **by value**

User supplied argument that is passed to the error, success, and confirm routines each time they are called. Whatever mechanism is used to pass user-arg is also used to pass it to the routines. This is an optional argument; if the argument is omitted, zero is passed by value.

rslt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

String into which the RMS resultant file specification of the last file processed is written. The rslt-file-spec argument is the address of a descriptor pointing to the resultant name.

If present, rslt-file-spec is used to store the file specification passed to the user supplied routines, instead of a default class S, type T string. Therefore, this argument should be specified when the user supplied routines are used and those routines require a descriptor type other than class S, type T. Any string class is supported.

If you specify one or more of the user supplied action routines, the descriptor used to pass rslt-file-spec must be:

- Of the same class as the descriptor required by the filespec argument of any action routines. For example, VAX Ada requires a class SB descriptor for string arguments to Ada routines but will use a class A descriptor by default when calling external routines. Refer to your language manual to determine the proper descriptor class to use.
- (Alpha only) Of the same form as the descriptor required by the filespec argument of all action routines. For example, if the filespec argument of an action routine uses a 64-bit descriptor, then the rslt-file-spec argument must also use a 64-bit descriptor.

context

VMS Usage: **context**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

A longword integer variable into which the routine stores a context value for use by future calls. The context argument is an unsigned longword integer containing the address of the context. This variable must be set to zero before the first call. You can use the same context argument from one call to another. This argument is used to retain the context when processing multiple input files. Portions of file specifications that the user does not specify may be inherited from the last files processed because the file contexts are retained in this argument. You must not change the value of context in subsequent calls.

extended-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating extended behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Flag	Usage
BINARY	Force binary file types into binary file format.
CONFIRM	A confirmation is required before any action is performed on a file.
COPY	.
DELETE	.
IGNORE_ DIRECTORIES	.
HEADER	.
MULTIPLE	.
NOSEARCHLIST	
NO_RLF	.
PRT	.
QUALIFIED	The extended file search qualifiers are to be used.
RETAIN_ CDT	.
RETAIN_ RDT	.
RECORD	Use record mode rather than block mode.
SEQUENTIAL	.
UPDATE	Open files with update allowed.

The flag values have symbolic names of the form UFSA$M_*flag* and UFSA$V_ *flag*.

ufsa-type

VMS Usage: **code**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies the structure of the source file specification.

The following table lists the types of file specification argument which can be passed.

Type	Structure
INDEX	A binary index passed by reference (not implemented yet).
LNKLST	Linked list of strings passed by reference.
STRING	String passed by descriptor.

The type values have a symbolic name of the form UFSA$C_*type*.

By default, STRING is assumed.

log-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating logging behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Description

The SWRK_DELETE_FILE routine ...

See also

SWRK_DELETE_INDEX
Delete an index

The SWRK_DELETE_INDEX routine deletes the elements in an index.

Format

status=*SWRK_DELETE_INDEX(index_root)*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

index_root

VMS Usage: **structure**
type: **INDEX_ROOT structure**
access: **modify**
mechanism: **by reference**

Specifies the address of the index root from which to delete the index elements.

Description

The SWRK_DELETE_INDEX routine deletes all the elements in an index structure leaving an empty index root. Calling SWRK_DELETE_INDEX with an empty index root causes no actions.

See also

SWRK_INSERT_INDEX
SWRK_LOOKUP_INDEX
SWRK_SCAN_INDEX

SWRK_DELETE_LNKLST Template

The SWRK_DELETE_LNKLST routine ...

Format

status=*SWRK_DELETE_LNKLST(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DELETE_LNKLST routine ...

See also

SWRK_DELETE_RECORD Template

The SWRK_DELETE_RECORD routine ...

Format

status=*SWRK_DELETE_RECORD(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DELETE_RECORD routine ...

See also

SWRK_DEQUEUE_SERVER_MESSAGE Template

The SWRK_DEQUEUE_SERVER_MESSAGE routine ...

Format

status=*SWRK_DEQUEUE_SERVER_MESSAGE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DEQUEUE_SERVER_MESSAGE routine ...

See also

SWRK_DISCONNECT_ALL Template

The SWRK_DISCONNECT_ALL routine ...

Format

status=*SWRK_DISCONNECT_ALL(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DISCONNECT_ALL routine ...

See also

SWRK_DISPLAY_HELP
Display OpenVMS help

The SWRK_DISPLAY_HELP routine displays OpenVMS help.

Format

status=*SWRK_DISPLAY_HELP(subject, library)*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

subject

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the subject on which HELP is to be displayed.

library

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the HELP library in which the subject should be found. If no HELP library is passed, the stanrad HELP search list is used.

Description

The SWRK_DISPLAY_HELP routine displays help on the requested subject by calling LBR$OUTPUT_HELP or SMG$PUT_HELP_TEXT.

It must be used in conjunction with the SWRK_DCL_HANDLER routine which establishes a command line context in which SWRK_DISPLAY_HELP operates.

SWRK_DO_COMMAND Template

The SWRK_DO_COMMAND routine ...

Format

status=*SWRK_DO_COMMAND(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DO_COMMAND routine ...

See also

SWRK_DUMP_IMAGE_INFO Template

The SWRK_DUMP_IMAGE_INFO routine ...

Format

status=*SWRK_DUMP_IMAGE_INFO(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DUMP_IMAGE_INFO routine ...

See also

SWRK_DUMP_MEMORY_BEGIN Template

The SWRK_DUMP_MEMORY_BEGIN routine ...

Format

status=*SWRK_DUMP_MEMORY_BEGIN(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DUMP_MEMORY_BEGIN routine ...

See also

SWRK_DUMP_MEMORY_CELL Template

The SWRK_DUMP_MEMORY_CELL routine ...

Format

status=*SWRK_DUMP_MEMORY_CELL(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DUMP_MEMORY_CELL routine ...

See also

SWRK_DUMP_MEMORY_CONTEXT Template

The SWRK_DUMP_MEMORY_CONTEXT routine ...

Format

status=*SWRK_DUMP_MEMORY_CONTEXT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DUMP_MEMORY_CONTEXT routine ...

See also

SWRK_DUMP_MEMORY_END Template

The SWRK_DUMP_MEMORY_END routine ...

Format

status=*SWRK_DUMP_MEMORY_END(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DUMP_MEMORY_END routine ...

See also

SWRK_DUMP_MEMORY_RANGE Template

The SWRK_DUMP_MEMORY_RANGE routine ...

Format

status=*SWRK_DUMP_MEMORY_RANGE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DUMP_MEMORY_RANGE routine ...

See also

SWRK_DUMP_PROCESS_INFO Template

The SWRK_DUMP_PROCESS_INFO routine ...

Format

status=*SWRK_DUMP_PROCESS_INFO(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_DUMP_PROCESS_INFO routine ...

See also

SWRK_EMPTY_STRING Template

The SWRK_EMPTY_STRING routine ...

Format

status=*SWRK_EMPTY_STRING(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_EMPTY_STRING routine ...

See also

SWRK_ENQUEUE_SERVER_MESSAGE Template

The SWRK_ENQUEUE_SERVER_MESSAGE routine ...

Format

status=*SWRK_ENQUEUE_SERVER_MESSAGE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_ENQUEUE_SERVER_MESSAGE routine ...

See also

SWRK_ESTABLISH
Establish Exception Handler

The Establish Exception Handler routine causes the SysWorks exception handler to be used for process exceptions.

Format

status=**SWRK_ESTABLISH***([force-frame])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

force-frame

VMS Usage: **boolean**
type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

Specifies whether the exception handler is to be placed in the current frame even when SWRK_ESTABLISH is called in user mode from within a shareable image when not debugging or gathering performance criteria. See the description below for details.

Description

The SWRK_ESTABLISH routine causes the SysWorks exception handler to be used for process exceptions. This can be achieved by placing the exception handler on the current frame or by declaring a secondary exception handler.

If SWRK_ESTABLISH is called from an elevated mode, the secondary exception handler is declared for that mode and for each outer mode eg. if declared in executive mode, it is also declared in supervisor and user mode.

If SWRK_ESTABLISH is called in user mode, a check is made to see whether the DEBUG or PCA$COLLECTOR image is present. If so, no exception handler is declared. If not, a check is made to see whether SWRK_ESTABLISH wass called from the main image or from a shareable image. If from the main image, the exception handler is established on the current frame, otherwise the secondary exception handler is declared as for elevated modes. If the **force-frame** argument is passed as true, the main or shareable image check is not made and the handler is always placed on the current frame.

SWRK_EXCEPTION_HANDLER Template

The SWRK_EXCEPTION_HANDLER routine ...

Format

status=*SWRK_EXCEPTION_HANDLER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_EXCEPTION_HANDLER routine ...

See also

SWRK_EXIT_IMAGE Template

The SWRK_EXIT_IMAGE routine ...

Format

status=*SWRK_EXIT_IMAGE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_EXIT_IMAGE routine ...

See also

SWRK_EXTENDED_FILE_SEARCH
Extended File Search

The Extended File Search routine performs a file search with extended input and output arguments compared with LIB$FILE_FILE.

Format

status=**SWRK_EXTENDED_FILE_SEARCH***(func, filespec, [default-filespec], [related-filespec], itmlst, [fab], [context])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

func

VMS Usage: **function_code**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies the function to be performed. Consists of a code in the low order word and a set of flags in the high order word.

filespec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

default-filespec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

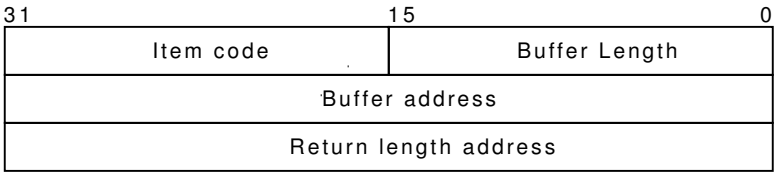
related-filespec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

itmlst

VMS Usage: **item_list_3**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Item list supplying information to be used in performing the function specified by the **func** argument. The **itmlst** argument is the address of the item list. The item list consists of one or more item descriptors, each of which contains an item code, buffer length, buffer address and retrn length address. The item list is terminated by an item code of 0 or by a longword of 0. The following diagram depicts the structure of a single item descriptor.



The following table defines the item descriptor fields.

Descriptor Field	Definition
Buffer length	A word specifying the length of the buffer; the buffer either supplies information to SWRK_EXTENDED_FILE_SEARCH or receives information from SWRK_EXTENDED_FILE_ SEARCH. The required length of the buffer varies, depending on the item code specified, and is given in the description of each item code.
Item code	A word containing an item code, which identifies the nature of the information supplied for SWRK_EXTENDED_FILE_ SEARCH or which is received from SWRK_EXTENDED_ FILE_SEARCH. Each item code has a symbolic name; the $EFSDEF macro defines these symbolic names.
Buffer address	Address of the buffer that specifies or receives the information.
Return length address	Address of a word to receive the length of information returned by SWRK_EXTENDED_FILE_SEARCH.

The item codes' symbolic names have the following format:

EFS$_code

There are two types of item code:

- **Input value item code.**

Since the **filespec**, **default-filespec** and **related-filespec** arguments are effectively input items, few function codes require that you specify at least one input value item code. The function code or codes for which each item code is valid are shown in parentheses after the item code description.

For input value item codes, the buffer length and buffer address fields of the item descriptor must be nonzero; the return length field must be zero. Specific buffer length requirements are given in the description of each item code.

- **Output value item code.**

  Output value item codes specify a buffer for information returned by SWRK_EXTENDED_FILE_SEARCH. For output value item codes, the buffer length and buffer address fields of the item descriptor must be nonzero; the return length field can be zero or nonzero. Specific buffer length requirements are given in the description of each item code.

fab

VMS Usage: **fab**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

Address of a FAB control block to be used instead of the internal FAB. If this argument is missing or zero, SWRK_EXTENDED_FILE_SEARCH uses its internal FAB and NAM control blocks by default. This argument is useful if a calling program is using its own FAB and wishes to use SWRK_EXTENDED_FILE_SEARCH to extract information not normally provided directly by RMS.

context

VMS Usage: **context**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

Address of a longword containing the number of a context stream for this call to the SWRK_EXTENDED_FILE_SEARCH routine. If the argument is unspecified or 0, the routine uses the default context stream (#0).

To generate a new context stream, the specified longword must contain -1. SWRK_EXTENDED_FILE_SEARCH then modifies the longword to hold the context number for that stream of operation. The context is marked with the caller's mode (user, supervisor, executive, or kernel). Any attempt to use that context in suc- cessive calls is checked and no call from a mode outside the recorded mode is allowed access.

To clean up a context, make a SWRK_EXTENDED_FILE_SEARCH call using the EFS$_FINISH function code and specify the address of the context number as the **context** argument.

Function Codes

This section lists each of the SWRK_EXTENDED_FILE_SEARCH function codes, describes the function, and lists the related item codes. All functions are context sensitive i.e. the action only applies to the specified or default context.

Where a function relates to a previous call to SWRK_EXTENDED_FILE_ SEARCH and no previous call has been made or the last call used the EFS$_ FINISH function, an error is returned

EFS$_CONTINUE
This request retrieves information about the next file associated with a previous call to SWRK_EXTENDED_FILE_SEARCH.

EFS$_FINISH
This request terminates a file search operation that may have been initiated by a previous call to SWRK_EXTENDED_FILE_SEARCH by closing channels and releasing the context block.

EFS$_GET_MORE
This request retrieves more information about the last file accessed by a previous call to SWRK_EXTENDED_FILE_SEARCH.

EFS$_GET_ACE
This request retrieves information about the first or next ACE associated with the file accessed by a previous call to SWRK_EXTENDED_FILE_SEARCH. If the previous call used the EFS$_GET_ACE function, the information about the next ACE is returned. If the previous call used the EFS$_START, EFS$_CONTINUE or EFS$_GET_MORE function code, information about the first ACE is returned.

EFS$_START
This request starts a new file search operation. Any request initiated by a previous call to SWRK_EXTENDED_FILE_SEARCH is terminated before the new request is started.

Function Modifiers

This section lists and describes each of the SWRK_EXTENDED_FILE_SEARCH function modifiers. Each modifier has a standard OpenVMS EFS$M_*modifier* and EFS$V_*modifier* forms which are a bit mask and a bit position.

EFS$M_QUALIFIED

Item Codes

This section lists each of the SWRK_EXTENDED_FILE_SEARCH item codes, describes the item, and lists the function codes for which it is relevant.

EFS$_ACE
This item returns the next ACE of the file as a textual string.

EFS$_ACE_BINARY
This item returns the next ACE of the file as a binary string.

EFS$_ACL
This item returns the entire ACL of the file as a textual string.

EFS$_ACL_BINARY
This item returns the entire ACL of the file as a binary string.

EFS$_BACKUP_DATE
This item returns the backup date of the file as a quadword date.

EFS$_CHANNEL
This item returns the channel used to access the file as a word integer.

EFS$_CREATION_DATE
This item returns the creation date of the file as a quadword date.

EFS$_DEVICE
This item returns the *device* part of the resultant file specification as a string.

EFS$_DEVICE_DIRECTORY
This item returns the *device and directory* part of the resultant file specification as a string.

EFS$_DIRECTORY
This item returns the *directory* part of the resultant file specification as a string.

EFS$_DISK_QUOTA
This item returns the disk quota of the owner of the file as a longword integer.

EFS$_EXPANDED_DEFAULT
This item returns the *device and directory* part of the expanded file specification as a string. This can be used to for the default for subsequent file specifications in a list.

EFS$_EXPANDED_FILE_SPEC
This item returns the expanded file specification as a string.

EFS$_EXPIRY_DATE
This item returns the expiry date of the file as a quadword date.

EFS$_FAB
This item returns the address of the FAB control block used to access the file as a longword pointer.

EFS$_FAT
This item returns the address of the FAT used to access the file as a longword pointer.

EFS$_FIB
This item returns the address of the FIB used to access the file as a longword pointer.

EFS$_FILE_CHARACTERISTICS
This item returns the file characteristics the file as a longword integer.

EFS$_FILE_ID
This item returns the file ID of the file as a three word array.

EFS$_FILE_SPEC
This item returns the resultant file specification as a string.

EFS$_NAME
This item returns the *file name* part of the resultant file specification as a string.

EFS$_NAME_TYPE
This item returns the *file name and type* part of the resultant file specification as a string.

EFS$_NODE
This item returns the *node* part of the resultant file specification as a string.

EFS$_OWNER
This item returns the owner of the file as a longword identifier.

EFS$_PROTECTION
This item returns the SOGW protection of the file as a word bitmask.

EFS$_REVISION_DATE
This item returns the revision date of the file as a quadword date.

EFS$_SIZE_ALLOCATED
This item returns the number of blocks allocated to the file as a longword integer.

EFS$_SIZE_USED
This item returns the number of blocks used by the file as a longword integer.

EFS$_TYPE
This item returns the *file type* part of the resultant file specification as a string.

EFS$_VERSION
This item returns the *version* part of the resultant file specification as a string.

Description

The SWRK_EXTENDED_FILE_SEARCH routine provides an enhanced file search ability. It uses the conventional **filespec**, **default-filespec** and **related-filespec** arguments associated with file search. It also uses an **itmlst** which provides input items for search qualification such as created after a certain date and output items which return values such as the creation date.

SWRK_FIND_ASSOCIATIONS Template

The SWRK_FIND_ASSOCIATIONS routine ...

Format

status=*SWRK_FIND_ASSOCIATIONS(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_FIND_ASSOCIATIONS routine ...

See also

SWRK_FIND_CLASS Template

The SWRK_FIND_CLASS routine ...

Format

status=*SWRK_FIND_CLASS(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_FIND_CLASS routine ...

See also

SWRK_FIND_OBJECT_ID Template

The SWRK_FIND_OBJECT_ID routine ...

Format

status=*SWRK_FIND_OBJECT_ID(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_FIND_OBJECT_ID routine ...

See also

SWRK_FIND_OBJECT Template

The SWRK_FIND_OBJECT routine ...

Format

status=*SWRK_FIND_OBJECT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_FIND_OBJECT routine ...

See also

SWRK_FIND_RESOURCE Template

The SWRK_FIND_RESOURCE routine ...

Format

status=*SWRK_FIND_RESOURCE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_FIND_RESOURCE routine ...

See also

SWRK_FIND_VARIABLE_INTEGER Template

The SWRK_FIND_VARIABLE_INTEGER routine ...

Format

status=*SWRK_FIND_VARIABLE_INTEGER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_FIND_VARIABLE_INTEGER routine ...

See also

SWRK_FIND_VARIABLE_STRING Template

The SWRK_FIND_VARIABLE_STRING routine ...

Format

status=*SWRK_FIND_VARIABLE_STRING(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_FIND_VARIABLE_STRING routine ...

See also

SWRK_FIXUP_STRING Template

The SWRK_FIXUP_STRING routine ...

Format

status=*SWRK_FIXUP_STRING(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_FIXUP_STRING routine ...

See also

SWRK_GENERIC_ST_BU Template

The SWRK_GENERIC_ST_BU routine ...

Format

status=*SWRK_GENERIC_ST_BU(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GENERIC_ST_BU routine ...

See also

SWRK_GENERIC_ST_RO Template

The SWRK_GENERIC_ST_RO routine ...

Format

status=*SWRK_GENERIC_ST_RO(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GENERIC_ST_RO routine ...

See also

SWRK_GENERIC_ST_RW Template

The SWRK_GENERIC_ST_RW routine ...

Format

status=*SWRK_GENERIC_ST_RW(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GENERIC_ST_RW routine ...

See also

SWRK_GET_ARCHITECTURE Template

The SWRK_GET_ARCHITECTURE routine ...

Format

status=*SWRK_GET_ARCHITECTURE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_ARCHITECTURE routine ...

See also

SWRK_GET_CHAN_FILE_SPEC Template

The SWRK_GET_CHAN_FILE_SPEC routine ...

Format

status=*SWRK_GET_CHAN_FILE_SPEC(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_CHAN_FILE_SPEC routine ...

See also

SWRK_GET_COMPUTER_NODE Template

The SWRK_GET_COMPUTER_NODE routine ...

Format

status=*SWRK_GET_COMPUTER_NODE(computer-name)*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

computer-arg

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

String into which SWRK_GET_COMPUTER_NODE returns the name of the computer.

Description

The SWRK_GET_COMPUTER_NODE routine ...

See also

SWRK_GET_CONTEXT Template

The SWRK_GET_CONTEXT routine ...

Format

status=*SWRK_GET_CONTEXT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_CONTEXT routine ...

See also

SWRK_GET_COUNTER Template

The SWRK_GET_COUNTER routine ...

Format

status=*SWRK_GET_COUNTER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_COUNTER routine ...

See also

SWRK_GET_CUR_ENV Template

The SWRK_GET_CUR_ENV routine ...

Format

status=*SWRK_GET_CUR_ENV(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_CUR_ENV routine ...

See also

SWRK_GET_CUR_PFX Template

The SWRK_GET_CUR_PFX routine ...

Format

status=*SWRK_GET_CUR_PFX(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_CUR_PFX routine ...

See also

SWRK_GET_DATE Template

The SWRK_GET_DATE routine ...

Format

status=*SWRK_GET_DATE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_DATE routine ...

See also

SWRK_GET_DATE_TIME Template

The SWRK_GET_DATE_TIME routine ...

Format

status=*SWRK_GET_DATE_TIME(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_DATE_TIME routine ...

See also

SWRK_GET_DEFAULT Template

The SWRK_GET_DEFAULT routine ...

Format

status=*SWRK_GET_DEFAULT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_DEFAULT routine ...

See also

SWRK_GET_FAB_FILE_SPEC Template

The SWRK_GET_FAB_FILE_SPEC routine ...

Format

status=*SWRK_GET_FAB_FILE_SPEC(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_FAB_FILE_SPEC routine ...

See also

SWRK_GET_FILE_SPEC Template

The SWRK_GET_FILE_SPEC routine ...

Format

status=*SWRK_GET_FILE_SPEC(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_FILE_SPEC routine ...

See also

SWRK_GET_IMAGE_NAME Template

The SWRK_GET_IMAGE_NAME routine ...

Format

status=*SWRK_GET_IMAGE_NAME(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_IMAGE_NAME routine ...

See also

SWRK_GET_INPUT Template

The SWRK_GET_INPUT routine ...

Format

status=*SWRK_GET_INPUT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_INPUT routine ...

See also

SWRK_GET_LAST_MSG_LOGGED Template

The SWRK_GET_LAST_MSG_LOGGED routine ...

Format

status=*SWRK_GET_LAST_MSG_LOGGED(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_LAST_MSG_LOGGED routine ...

See also

SWRK_GET_LOG_DEFAULTS Template

The SWRK_GET_LOG_DEFAULTS routine ...

Format

status=*SWRK_GET_LOG_DEFAULTS(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_LOG_DEFAULTS routine ...

See also

SWRK_GET_LOG_QUAL_FILE Template

The SWRK_GET_LOG_QUAL_FILE routine ...

Format

status=*SWRK_GET_LOG_QUAL_FILE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_LOG_QUAL_FILE routine ...

See also

SWRK_GET_POWERHOUSE_RESOURCES Template

The SWRK_GET_POWERHOUSE_RESOURCES routine ...

Format

status=*SWRK_GET_POWERHOUSE_RESOURCES(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_POWERHOUSE_RESOURCES routine ...

See also

SWRK_GET_PROCESS_IMAGECOUNT Template

The SWRK_GET_PROCESS_IMAGECOUNT routine ...

Format

status=*SWRK_GET_PROCESS_IMAGECOUNT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_PROCESS_IMAGECOUNT routine ...

See also

SWRK_GET_RECORD Template

The SWRK_GET_RECORD routine ...

Format

status=*SWRK_GET_RECORD(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_RECORD routine ...

See also

SWRK_GET_SCOPE_CONTEXT Template

The SWRK_GET_SCOPE_CONTEXT routine ...

Format

status=*SWRK_GET_SCOPE_CONTEXT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_SCOPE_CONTEXT routine ...

See also

SWRK_GET_STATISTICS_INFO Template

The SWRK_GET_STATISTICS_INFO routine ...

Format

status=*SWRK_GET_STATISTICS_INFO(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_STATISTICS_INFO routine ...

See also

SWRK_GET_SUBCONTEXT Template

The SWRK_GET_SUBCONTEXT routine ...

Format

status=*SWRK_GET_SUBCONTEXT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_SUBCONTEXT routine ...

See also

SWRK_GET_SYMBOL_INTEGER Template

The SWRK_GET_SYMBOL_INTEGER routine ...

Format

status=*SWRK_GET_SYMBOL_INTEGER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_SYMBOL_INTEGER routine ...

See also

SWRK_GET_SYMBOL_STRING Template

The SWRK_GET_SYMBOL_STRING routine ...

Format

status=*SWRK_GET_SYMBOL_STRING(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_SYMBOL_STRING routine ...

See also

SWRK_GET_USERNAME Template

The SWRK_GET_USERNAME routine ...

Format

status=*SWRK_GET_USERNAME(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_GET_USERNAME routine ...

See also

SWRK_HANDLE_IMAGE_VECTOR Template

The SWRK_HANDLE_IMAGE_VECTOR routine ...

Format

status=*SWRK_HANDLE_IMAGE_VECTOR(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_HANDLE_IMAGE_VECTOR routine ...

See also

SWRK_HANDLE_KEYWORD Template

The SWRK_HANDLE_KEYWORD routine ...

Format

status=*SWRK_HANDLE_KEYWORD(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_HANDLE_KEYWORD routine ...

See also

SWRK_HANDLE_QUALIFIERS Template

The SWRK_HANDLE_QUALIFIERS routine ...

Format

status=*SWRK_HANDLE_QUALIFIERS(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_HANDLE_QUALIFIERS routine ...

See also

SWRK_INCREMENT_COUNTER Template

The SWRK_INCREMENT_COUNTER routine ...

Format

status=*SWRK_INCREMENT_COUNTER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_INCREMENT_COUNTER routine ...

See also

SWRK_INITIALIZE_IMAGE Template

The SWRK_INITIALIZE_IMAGE routine ...

Format

status=*SWRK_INITIALIZE_IMAGE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_INITIALIZE_IMAGE routine ...

See also

SWRK_INITIALIZE_LOGGING Template

The SWRK_INITIALIZE_LOGGING routine ...

Format

status=*SWRK_INITIALIZE_LOGGING(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_INITIALIZE_LOGGING routine ...

See also

SWRK_INITIALIZE_TRACE Template

The SWRK_INITIALIZE_TRACE routine ...

Format

status=*SWRK_INITIALIZE_TRACE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_INITIALIZE_TRACE routine ...

See also

SWRK_INQUIRE_SERVER Template

The SWRK_INQUIRE_SERVER routine ...

Format

status=*SWRK_INQUIRE_SERVER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_INQUIRE_SERVER routine ...

See also

SWRK_INSERT_INDEX
Insert a key into an index

The SWRK_INSERT_INDEX routine inserts a key into an index.

Format

status=*SWRK_INSERT_INDEX(index_root, key, new_element, item_sizes)*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

index_root

VMS Usage: **structure**
type: **INDEX_ROOT structure**
access: **modify**
mechanism: **by reference**

Specifies the address of the index root into which to insert an index element with the specified key.

key

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the address of the key string descriptor.

new_element

VMS Usage: **address**
type: **address**
access: **write only**
mechanism: **by reference**

Specifies the address of a pointer which receives the address of the new index element. If duplicates are not allowed, and an attempt is made to insert a duplicate element, the address returned is that of the existing index element.

items_sizes

VMS Usage: **address**
type: **INDEX_ELEMENT structure**
access: **write only**
mechanism: **by reference**

Specifies the address of an array of words which indicate the size of each index element data item. If this argument is not specified or is zero, no data items are associated with the index element. Each item in the array indicates the size of an index element data item. A negative value terminates the list. A zero item size indicates that the data item descriptor is not setup. A positive value indicates that the corresponding data item descriptor is setup, however no data is actually stored in the area pointed to by the descriptor. It is the responsibility of the calling procedure to move data into the areas reserved by the descriptor.

Description

The SWRK_INSERT_INDEX routine inserts an index element with the specified key into an index structure. The index root indicates whether duplicate keys are allowed.

See also

SWRK_DELETE_INDEX
SWRK_LOOKUP_INDEX
SWRK_SCAN_INDEX

SWRK_INSERT_QUEUE_HEAD Template

The SWRK_INSERT_QUEUE_HEAD routine ...

Format

status=*SWRK_INSERT_QUEUE_HEAD(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_INSERT_QUEUE_HEAD routine ...

See also

SWRK_INSERT_QUEUE_TAIL Template

The SWRK_INSERT_QUEUE_TAIL routine ...

Format

status=*SWRK_INSERT_QUEUE_TAIL(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_INSERT_QUEUE_TAIL routine ...

See also

SWRK_LOCK_RESOURCE Template

The SWRK_LOCK_RESOURCE routine ...

Format

status=*SWRK_LOCK_RESOURCE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOCK_RESOURCE routine ...

See also

SWRK_LOG_FAOZ Template

The SWRK_LOG_FAOZ routine ...

Format

status=*SWRK_LOG_FAOZ(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_FAOZ routine ...

See also

SWRK_LOG_FAO Template

The SWRK_LOG_FAO routine ...

Format

status=*SWRK_LOG_FAO(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_FAO routine ...

See also

SWRK_LOG_IMAGE_FINISH Template

The SWRK_LOG_IMAGE_FINISH routine ...

Format

status=*SWRK_LOG_IMAGE_FINISH(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_IMAGE_FINISH routine ...

See also

SWRK_LOG_IMAGE_START Template

The SWRK_LOG_IMAGE_START routine ...

Format

status=*SWRK_LOG_IMAGE_START(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_IMAGE_START routine ...

See also

SWRK_LOG_MSG_2_VEC
Log a message passed as a message vector

This routine logs a message which is passed as a message vector.

Format

status=**SWRK_LOG_MSG_VEC_2***(flags, source_file, source_reference, message_vector, facility_code, return_message_buffer)*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

See SWRK_LOG_MSG_VEC for more details about the first four arguments.

Arguments

flags

VMS Usage: **flags**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies which logging actions the routine should take.

source_file

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the source from which the message is being logged.

source_reference

VMS Usage: **number**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies a reference number within the source.

message_vector

VMS Usage: **cntrlblk**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies the message vector which contains the messages to be logged.

facility_code

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the facility code to be used for the first message in the message vector. If this argument is omitted, the facility code indicated in the message vector status will be used by default.

return_message_buffer

VMS Usage: **char_string_ptr**
type: **character string pointer**
access: **write only**
mechanism: **by reference**

Specifies the return address of a pointer to the expanded message buffer. The SysWorks logging procedures use internal buffers to store the expanded message before it is written to any log file or destination. The contents of these buffers are left intact until the next call to a message logging routine. Therefore, the address returned can be used by the caller to implement extra facilities not directly provided by the SysWorks message logging procedures.

Description

This procedure logs the message specified by the message vector to appropriate log files. It expands the SWRK_LOG_MSG_VEC routine by providing access to some internal features.

SWRK_LOG_MSG_VEC
Log a message passed as a message vector

This routine logs a message which is passed as a message vector.

Format

status=**SWRK_LOG_MSG_VEC***(flags, source_file, source_reference, message_vector)*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

flags

VMS Usage: **flags**
type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

Specifies which logging actions the routine should take. The following values are permitted:

Flag	Usage
LOG_M_CONTEXT	Log the message in a context shared log file. The file name SWRK_CONTEXT_LOG_FILE is used to create the context log file. Defining this name as a logical name allows the log file to be created with another file specification.
LOG_M_CRASH	Force the image to crash. Overrides LOG_M_EXIT if that is also specified.
LOG_M_DEFAULT	Use the current default flags value.
LOG_M_ERROR	Log the message to SYS$ERROR.
LOG_M_EXIT	Force the image to exit. Ignored in LOG_M_CRASH is also specified.
LOG_M_FLUSH	Force any cached log file (such as an image log file) to be flushed.
LOG_M_IMAGE	Log the message to the image log file.
LOG_M_INDIRECT	Log a one line message in the context log file which directs the reader to the (image) log file which contains the full message. Overrides LOG_M_CONTEXT if that is also specified.

Flag	Usage
LOG_M_MAIL	Send the message as mail. The address to which the mail is sent is defined by the logical name SWRK_MAIL_ADDRESS . The equivalence of this logical may start with an @, in which case the address is a distribution list.
LOG_M_MORE	Replace any leading percent character with a hyphen, effectively creating a continuation message.
LOG_M_NOHEADING	Don't include a heading line.
LOG_M_NOW	Don't return or exit until the message is logged. This flag is assumed if LOG_M_CRASH or LOG_M_EXIT are specified.
LOG_M_OPERATOR	Log the message to the operator log file.
LOG_M_OUTPUT	Log the message to SYS$OUTPUT.
LOG_M_SIMPLE	Equivalent to LOG_M_ERROR + LOG_M_OUTPUT + LOG_M_ NOHEADING.
LOG_M_TRACE	Log the message to the image trace file.

source_file

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the source from which the message is being logged. In C, this would normally be a descriptor provided by the `SWRK_BASE.H` header file and based on the `__FILE__` builtin. In Cobol, it would be a string literal containing the procedure name passed by descriptor.

This argument is used to construct an initial line inserted before the remainder of the message indicating where the message was logged from. If this argument is ommitted, no initial line will precede the remaining message.

source_reference

VMS Usage: **number**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies a reference number within the source. In C, this would normally be the line number provided by the `__LINE__` builtin. If this argument is omitted a value of zero is assumed by default.

message_vector

VMS Usage: **cntrlblk**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies the message vector which contains the messages to be logged.

Description

This procedure logs the message specified by the message vector to appropriate log files. Other logging procedures (such as SWRK_LOG_STATUS) convert their arguments into a message vector and call this procedure.

SWRK_LOG_OUTPUT_2 Template

The SWRK_LOG_OUTPUT_2 routine ...

Format

status=*SWRK_LOG_OUTPUT_2(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_OUTPUT_2 routine ...

See also

SWRK_LOG_OUTPUT Template

The SWRK_LOG_OUTPUT routine ...

Format

status=*SWRK_LOG_OUTPUT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_OUTPUT routine ...

See also

SWRK_LOG_PRINTF
Log C printf style message

This routine logs a message which is passed as a C style format string with optional arguments.

Format

status=**SWRK_LOG_PRINTF***(flags, source_file, source_reference, format [, output_source...])***

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

See SWRK_LOG_MSG_VEC for more details about the first three arguments.

Arguments

flags

VMS Usage: **flags**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies which logging actions the routine should take.

source_file

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the source from which the message is being logged.

source_reference

VMS Usage: **number**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies a reference number within the source.

format

VMS Usage: **char_string**
type: **null terminated character string**
access: **read only**

mechanism: **by reference**

Specifies characters to be written literally to the output or converted as specified in the `outout_source` arguments.

output_source

VMS Usage: **varying_arg**
type: **longword (signed)**
access: **read only**
mechanism: **by value**

Specifies optional expressions whose resultant types correspond to conversion specifications given in the format specification.

If no conversion specifications are given, you may omit the output sources. Otherwise, the function call must have exactly as many output sources as there are conversion spec- ifications, and the conversion specifications must match the types of the output sources.

Conversion specifications are matched to output sources in left-to-right order. Excess output pointers, if any, are ignored.

Description

This routine logs the formatted message specified by the format and output source arguments to appropriate log files. It is similar to a call to `sprintf` followed by a call to SWRK_LOG_TEXT.

See also

SWRK_LOG_MSG_VEC
SWRK_LOG_TEXT

SWRK_LOG_RDB_NO_TRANS Template

The SWRK_LOG_RDB_NO_TRANS routine ...

Format

status=*SWRK_LOG_RDB_NO_TRANS(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_RDB_NO_TRANS routine ...

See also

SWRK_LOG_RDB Template

The SWRK_LOG_RDB routine ...

Format

status=*SWRK_LOG_RDB(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_RDB routine ...

See also

SWRK_LOG_RDB_SETUP_2 Template

The SWRK_LOG_RDB_SETUP_2 routine ...

Format

status=*SWRK_LOG_RDB_SETUP_2(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_RDB_SETUP_2 routine ...

See also

SWRK_LOG_RDB_SETUP Template

The SWRK_LOG_RDB_SETUP routine ...

Format

status=*SWRK_LOG_RDB_SETUP(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_RDB_SETUP routine ...

See also

SWRK_LOG_RMS_FAB_2 Template

The SWRK_LOG_RMS_FAB_2 routine ...

Format

status=*SWRK_LOG_RMS_FAB_2(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_RMS_FAB_2 routine ...

See also

SWRK_LOG_RMS_FAB Template

The SWRK_LOG_RMS_FAB routine ...

Format

status=*SWRK_LOG_RMS_FAB(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_RMS_FAB routine ...

See also

SWRK_LOG_RMS_FILE_STS Template

The SWRK_LOG_RMS_FILE_STS routine ...

Format

status=*SWRK_LOG_RMS_FILE_STS(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_RMS_FILE_STS routine ...

See also

SWRK_LOG_RMS_RAB Template

The SWRK_LOG_RMS_RAB routine ...

Format

status=*SWRK_LOG_RMS_RAB(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_LOG_RMS_RAB routine ...

See also

SWRK_LOG_STATUS
Log a message passed as a status code

This routine logs a message which is passed as an OpenVMS status with optional arguments.

Format

status=**SWRK_LOG_STATUS***(flags, source_file, source_reference, status [, value...])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

See SWRK_LOG_MSG_VEC for more details about the first three arguments.

Arguments

flags

VMS Usage: **flags**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies which logging actions the routine should take.

source_file

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the source from which the message is being logged.

source_reference

VMS Usage: **number**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies a reference number within the source.

status

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **read only**

mechanism: **by value**

Specifies the OpenVMS message id of the message to be logged. See the $GETMSG system service for more details.

values

VMS Usage: **varying_arg**
type: **longword (signed)**
access: **read only**
mechanism: **by value**

Specifies optional expressions whose resultant types correspond to conversion specifications given in the OpenVMS message text.

If no conversion specifications are given, you may omit the output sources. Otherwise, the function call must have exactly as many output sources as there are conversion spec- ifications, and the conversion specifications must match the types of the output sources.

Conversion specifications are matched to output sources in left-to-right order. Excess output pointers, if any, are ignored.

Description

This routine logs the formatted message specified by the status and value arguments to appropriate log files. It is similar to a call to SWRK_SETUP_ MSG_VEC followed by a call to SWRK_LOG_MSG_VEC.

See also

SWRK_LOG_MSG_VEC
SWRK_SETUP_MSG_VEC

SWRK_LOG_TEXT
Log text message

This routine logs a message which is passed as simple text string.

Format

status=**SWRK_LOG_TEXT***(flags, source_file, source_reference, message_text)*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

See SWRK_LOG_MSG_VEC for more details about the first three arguments.

Arguments

flags

VMS Usage: **flags**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies which logging actions the routine should take.

source_file

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the source from which the message is being logged.

source_reference

VMS Usage: **number**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies a reference number within the source.

message_text

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the message to be logged.

Description

This routine logs the message specified by the text message argument to appropriate log files.

See also

SWRK_LOG_MSG_VEC

SWRK_LOOKUP_INDEX
Lookup a key in an index

The SWRK_LOOKUP_INDEX routine searches an index for a specific key.

Format

status=*SWRK_LOOKUP_INDEX(index_root, key, element)*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

index_root

VMS Usage: **structure**
type: **INDEX_ROOT structure**
access: **modify**
mechanism: **by reference**

Specifies the address of the index root in which to search for an index element with the specified key.

key

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies the address of the key string descriptor.

new_element

VMS Usage: **address**
type: **address**
access: **write only**
mechanism: **by reference**

Specifies the address of a pointer which receives the address of the found index element.

Description

The SWRK_LOOKUP_INDEX routine searches an index for an index element with the specified key.

See also

SWRK_DELETE_INDEX
SWRK_INSERT_INDEX
SWRK_SCAN_INDEX

SWRK_MANAGE_FILE Template

The SWRK_MANAGE_FILE routine ...

Format

status=*SWRK_MANAGE_FILE(file-spec [,dflt-file-spec] [,rltd-file-spec] [,user-success-clb] [,user-error-clb] [,user-confirm-clb] [,user-arg] [,rslt-file-spec] [,context] [,extended-flags] [,ufsa-type] [,log-flags])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

String containing the OpenVMS Record Management Services (RMS) file specification of the files to be managed. The file-spec argument is the address of a descriptor pointing to the file specification. If the specification includes wildcards, each file that matches the specification is managed. The string must not contain more than 255 characters. Any string class is supported.

On Alpha systems, set the LIB$M_FIL_LONG_NAMES bit in the flags argument for strings longer than 255 characters in length.

dflt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Default file specification of the files to be managed. The dflt-file-spec argument is the address of a descriptor pointing to the default file specification. This is an optional argument; if the argument is omitted, the default is the null string. Any string class is supported.

See the OpenVMS Record Management Services Reference Manual for information about default file specifications.

rltd-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Related file specification of the files to be managed. The rltd-file-spec argument is the address of a descriptor pointing to the related file specification. Any string class is supported. This is an optional argument; if the argument is omitted, the default is the null string.

Input file parsing is used. See the OpenVMS Record Management Services Reference Manual for information on related file specifications and input file parsing.

The related file specification is useful when you are processing lists of file specifications. Unspecified portions of the file specification are inherited from the last file processed.

user-success-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied success routine that is called after a file is successfully managed.

The success routine can be used to display a log of the files that were managed.

user-error-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied error routine that is called when an error is detected.

The error routine returns a success/fail value that is used to determine if more files should be managed.

user-confirm-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied confirm routine that is called before each file is managed. The value returned by the confirm routine determines whether or not the file will be managed. The confirm routine can be used to select specific files for management based on criteria such as expiration date, size, and so on.

user-arg

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**

mechanism: **by value**

User supplied argument that is passed to the error, success, and confirm routines each time they are called. Whatever mechanism is used to pass user-arg is also used to pass it to the routines. This is an optional argument; if the argument is omitted, zero is passed by value.

rslt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

String into which the RMS resultant file specification of the last file processed is written. The rslt-file-spec argument is the address of a descriptor pointing to the resultant name.

If present, rslt-file-spec is used to store the file specification passed to the user supplied routines, instead of a default class S, type T string. Therefore, this argument should be specified when the user supplied routines are used and those routines require a descriptor type other than class S, type T. Any string class is supported.

If you specify one or more of the user supplied action routines, the descriptor used to pass rslt-file-spec must be:

- Of the same class as the descriptor required by the filespec argument of any action routines. For example, VAX Ada requires a class SB descriptor for string arguments to Ada routines but will use a class A descriptor by default when calling external routines. Refer to your language manual to determine the proper descriptor class to use.
- (Alpha only) Of the same form as the descriptor required by the filespec argument of all action routines. For example, if the filespec argument of an action routine uses a 64-bit descriptor, then the rslt-file-spec argument must also use a 64-bit descriptor.

context

VMS Usage: **context**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

A longword integer variable into which the routine stores a context value for use by future calls. The context argument is an unsigned longword integer containing the address of the context. This variable must be set to zero before the first call. You can use the same context argument from one call to another. This argument is used to retain the context when processing multiple input files. Portions of file specifications that the user does not specify may be inherited from the last files processed because the file contexts are retained in this argument. You must not change the value of context in subsequent calls.

extended-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating extended behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Flag	Usage
BINARY	Force binary file types into binary file format.
CONFIRM	A confirmation is required before any action is performed on a file.
COPY	.
DELETE	.
IGNORE_ DIRECTORIES	.
HEADER	.
MULTIPLE	.
NOSEARCHLIST	
NO_RLF	.
PRT	.
QUALIFIED	The extended file search qualifiers are to be used.
RETAIN_ CDT	.
RETAIN_ RDT	.
RECORD	Use record mode rather than block mode.
SEQUENTIAL	.
UPDATE	Open files with update allowed.

The flag values have symbolic names of the form UFSA$M_*flag* and UFSA$V_ *flag*.

ufsa-type

VMS Usage: **code**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies the structure of the source file specification.

The following table lists the types of file specification argument which can be passed.

Type	Structure
INDEX	A binary index passed by reference (not implemented yet).
LNKLST	Linked list of strings passed by reference.
STRING	String passed by descriptor.

The type values have a symbolic name of the form UFSA$C_*type*.

By default, STRING is assumed.

log-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating logging behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Description

The SWRK_MANAGE_FILE routine ...

See also

SWRK_MATCH_LNKLST Template

The SWRK_MATCH_LNKLST routine ...

Format

status=*SWRK_MATCH_LNKLST(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_MATCH_LNKLST routine ...

See also

SWRK_MATCH_STRING Template

The SWRK_MATCH_STRING routine ...

Format

status=*SWRK_MATCH_STRING(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_MATCH_STRING routine ...

See also

SWRK_MOVE_FILE Template

The SWRK_MOVE_FILE routine ...

Format

status=*SWRK_MOVE_FILE(old-file-spec, new-file-spec [,dflt-file-spec] [,rltd-file-spec] [,flags] [,user-success-clb] [,user-error-clb] [,user-confirm-clb] [,user-arg] [,old-rslt-file-spec] [,new-rslt-file-spec] [,context] [,extended-flags] [,ufsa-type] [,log-flags])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

old-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

File specification of the files to be moved. The old-file-spec argument is the address of a descriptor pointing to the old file specification. The specification may include wildcards, in which case each file that matches the specification will be moved. The string must not contain more than 255 characters. Any string class is supported.

new-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

File specification for the new file names. The new-file-spec argument is the address of a descriptor pointing to the new file specification.

This specification need not be complete; fields omitted or specified by using the wildcard character (*) will be filled in from the existing file's name using the same rules as for the equivalent DCL command. The string must not contain more than 255 characters. Any string class is supported.

dflt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**

mechanism: **by descriptor**

Default file specification of the files to be moved. The dflt-file-spec argument is the address of a descriptor pointing to the default file specification. This is an optional argument; if the argument is omitted, the default is the null string. Any string class is supported.

See the OpenVMS Record Management Services Reference Manual for information about default file specifications.

rltd-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Related file specification of the files to be moved. The rltd-file-spec argument is the address of a descriptor pointing to the related file specification. Any string class is supported. This is an optional argument; if the argument is omitted, the default is the null string.

Input file parsing is used. See the OpenVMS Record Management Services Reference Manual for information on related file specifications and input file parsing.

The related file specification is useful when you are processing lists of file specifications. Unspecified portions of the file specification are inherited from the last file processed.

flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating optional behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

user-success-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied success routine that is called after a file is successfully moved.

The success routine can be used to display a log of the files that were moved.

user-error-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied error routine that is called when an error is detected.

The error routine returns a success/fail value that is used to determine if more files should be moved.

user-confirm-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied confirm routine that is called before each file is moved. The value returned by the confirm routine determines whether or not the file will be moved. The confirm routine can be used to select specific files for moving based on criteria such as expiration date, size, and so on.

user-arg

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

User supplied argument that is passed to the error, success, and confirm routines each time they are called. Whatever mechanism is used to pass user-arg is also used to pass it to the routines. This is an optional argument; if the argument is omitted, zero is passed by value.

old-rslt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

String into which the old resultant file specification of the last file processed is copied. This is an optional argument. If present, it is used to store the file specification passed to the user supplied routines instead of a default class S, type T string. Any string class is supported.

If you are specifying one or more of the action routine arguments, be sure that the descriptor class used to pass the resultant name is the same as the descriptor class required by the action routine. For example, VAX Ada requires a class SB descriptor for string arguments to Ada routines, but will use a class A descriptor by default when calling external routines.

Refer to your language manual to determine the proper descriptor class to use.

new-rslt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

String into which new OpenVMS RMS resultant file specification of the last file processed is written. The new-rslt-file-spec argument is the address of a descriptor pointing to the new name. This is an optional argument. If present, it is used to store the file specification passed to the user supplied routines instead of a class S, type T string. Any string class is supported.

If you are specifying one or more of the action routine arguments, be sure that the descriptor class used to pass the resultant name is the same as the descriptor class required by the action routine. For example, VAX Ada requires a class SB descriptor for string arguments to Ada routines, but will use a class A descriptor by default when calling external routines. Refer to your language manual to determine the proper descriptor class to use.

context

VMS Usage: **context**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

A longword integer variable into which the routine stores a context value for use by future calls. The context argument is an unsigned longword integer containing the address of the context. This variable must be set to zero before the first call. You can use the same context argument from one call to another. This argument is used to retain the context when processing multiple input files. Portions of file specifications that the user does not specify may be inherited from the last files processed because the file contexts are retained in this argument. You must not change the value of context in subsequent calls.

extended-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating extended behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Flag	Usage
BINARY	Force binary file types into binary file format.
CONFIRM	A confirmation is required before any action is performed on a file.
COPY	.
DELETE	.
IGNORE_ DIRECTORIES	.
HEADER	.
MULTIPLE	.
NOSEARCHLIST	
NO_RLF	.

Flag	Usage
PRT	.
QUALIFIED	The extended file search qualifiers are to be used.
RETAIN_CDT	.
RETAIN_RDT	.
RECORD	Use record mode rather than block mode.
SEQUENTIAL	.
UPDATE	Open files with update allowed.

The flag values have symbolic names of the form UFSA$M_*flag* and UFSA$V_*flag*.

ufsa-type

VMS Usage: **code**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies the structure of the source file specification.

The following table lists the types of file specification argument which can be passed.

Type	Structure
INDEX	A binary index passed by reference (not implemented yet).
LNKLST	Linked list of strings passed by reference.
STRING	String passed by descriptor.

The type values have a symbolic name of the form UFSA$C_*type*.

By default, STRING is assumed.

log-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating logging behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Description

The SWRK_MOVE_FILE routine ...

See also

SWRK_OPEN_FILE Template

The SWRK_OPEN_FILE routine ...

Format

status=*SWRK_OPEN_FILE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_OPEN_FILE routine ...

See also

SWRK_OPEN_INPUT Template

The SWRK_OPEN_INPUT routine ...

Format

status=*SWRK_OPEN_INPUT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_OPEN_INPUT routine ...

See also

SWRK_OPEN_JOB Template

The SWRK_OPEN_JOB routine ...

Format

status=*SWRK_OPEN_JOB(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_OPEN_JOB routine ...

See also

SWRK_OPEN_OUTPUT Template

The SWRK_OPEN_OUTPUT routine ...

Format

status=*SWRK_OPEN_OUTPUT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_OPEN_OUTPUT routine ...

See also

SWRK_PARSE_DIRECTORY Template

The SWRK_PARSE_DIRECTORY routine ...

Format

status=*SWRK_PARSE_DIRECTORY(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_PARSE_DIRECTORY routine ...

See also

SWRK_PARSE_FILE Template

The SWRK_PARSE_FILE routine ...

Format

status=*SWRK_PARSE_FILE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_PARSE_FILE routine ...

See also

SWRK_PARSE_LOG_FILE Template

The SWRK_PARSE_LOG_FILE routine ...

Format

status=*SWRK_PARSE_LOG_FILE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_PARSE_LOG_FILE routine ...

See also

SWRK_PREFIX_MSG_VEC
Insert a new message in a message vector

This routine inserts a new message at the beginning of a message vector.

Format

status=**SWRK_PREFIX_MSG_VEC***(message_vector, status [, value...])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

status

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies the OpenVMS message id of the message to be logged. See the $GETMSG system service for more details.

values

VMS Usage: **varying_arg**
type: **longword (signed)**
access: **read only**
mechanism: **by value**

Specifies optional expressions whose resultant types correspond to conversion specifications given in the OpenVMS message text.

If no conversion specifications are given, you may omit the output sources. Otherwise, the function call must have exactly as many output sources as there are conversion spec- ifications, and the conversion specifications must match the types of the output sources.

Conversion specifications are matched to output sources in left-to-right order. Excess output pointers, if any, are ignored.

message_vector

VMS Usage: **cntrlblk**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

Specifies the message vector into which a new message will be loaded.

Description

This procedure inserts the message specified by the status and value arguments into the beginning of a message vector. The message vector must be correctly initialized before a call to SWRK_PREFIX_MSG_VEC is made. Any existing entries are moved down. The number of entries inserted (and hence the distance the existing entries are moved) is based on the facility code of the status. See the *OpenVMS System Services Reference Manual* for details about message vector layouts.

To build a message vector with more than one message, a call is made to SWRK_SETUP_MSG_VEC to setup the last message line, followed by calls to SWRK_PREFIX_MSG_VEC to setup earlier lines. The last call to SWRK_PREFIX_MSG_VEC inserts the first message line.

SWRK_PRINT_FILE Template

The SWRK_PRINT_FILE routine ...

Format

status=*SWRK_PRINT_FILE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_PRINT_FILE routine ...

See also

SWRK_PROCESS_FILE Template

The SWRK_PROCESS_FILE routine ...

Format

status=*SWRK_PROCESS_FILE(file-spec [,dflt-file-spec] [,rltd-file-spec] [,user-success-clb] [,user-error-clb] [,user-confirm-clb] [,user-arg] [,rslt-file-spec] [,user-file-clb] [,user-record-clb] [,context] [,extended-flags] [,ufsa-type] [,log-flags])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

String containing the OpenVMS Record Management Services (RMS) file specification of the files to be processed. The file-spec argument is the address of a descriptor pointing to the file specification. If the specification includes wildcards, each file that matches the specification is processed. The string must not contain more than 255 characters. Any string class is supported.

On Alpha systems, set the LIB$M_FIL_LONG_NAMES bit in the flags argument for strings longer than 255 characters in length.

dflt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Default file specification of the files to be processed. The dflt-file-spec argument is the address of a descriptor pointing to the default file specification. This is an optional argument; if the argument is omitted, the default is the null string. Any string class is supported.

See the OpenVMS Record Management Services Reference Manual for information about default file specifications.

rltd-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Related file specification of the files to be processed. The rltd-file-spec argument is the address of a descriptor pointing to the related file specification. Any string class is supported. This is an optional argument; if the argument is omitted, the default is the null string.

Input file parsing is used. See the OpenVMS Record Management Services Reference Manual for information on related file specifications and input file parsing.

The related file specification is useful when you are processing lists of file specifications. Unspecified portions of the file specification are inherited from the last file processed.

user-success-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied success routine that is called after a file is successfully processed.

The success routine can be used to display a log of the files that were processed.

user-error-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied error routine that is called when an error is detected.

The error routine returns a success/fail value that is used to determine if more files should be processed.

user-confirm-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied confirm routine that is called before each file is processed. The value returned by the confirm routine determines whether or not the file will be processed. The confirm routine can be used to select specific files for processing based on criteria such as expiration date, size, and so on.

user-arg

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**

mechanism: **by value**

User supplied argument that is passed to the error, success, and confirm routines each time they are called. Whatever mechanism is used to pass user-arg is also used to pass it to the routines. This is an optional argument; if the argument is omitted, zero is passed by value.

rslt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

String into which the RMS resultant file specification of the last file processed is written. The rslt-file-spec argument is the address of a descriptor pointing to the resultant name.

If present, rslt-file-spec is used to store the file specification passed to the user supplied routines, instead of a default class S, type T string. Therefore, this argument should be specified when the user supplied routines are used and those routines require a descriptor type other than class S, type T. Any string class is supported.

If you specify one or more of the user supplied action routines, the descriptor used to pass rslt-file-spec must be:

- Of the same class as the descriptor required by the filespec argument of any action routines. For example, VAX Ada requires a class SB descriptor for string arguments to Ada routines but will use a class A descriptor by default when calling external routines. Refer to your language manual to determine the proper descriptor class to use.
- (Alpha only) Of the same form as the descriptor required by the filespec argument of all action routines. For example, if the filespec argument of an action routine uses a 64-bit descriptor, then the rslt-file-spec argument must also use a 64-bit descriptor.

user-file-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied file routine that is called for each file that is processed.

user-record-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied record routine that is called for each record in then file that is being processed. If this argument is supplied, each file which is to be processed is opened and read. Each record found in the file is then passed to this routine to be processed is some user specified fashion. If this argument is not supplied, the file is not opened and read.

context

VMS Usage: **context**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

A longword integer variable into which the routine stores a context value for use by future calls. The context argument is an unsigned longword integer containing the address of the context. This variable must be set to zero before the first call. You can use the same context argument from one call to another. This argument is used to retain the context when processing multiple input files. Portions of file specifications that the user does not specify may be inherited from the last files processed because the file contexts are retained in this argument. You must not change the value of context in subsequent calls.

extended-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating extended behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Flag	Usage
BINARY	Force binary file types into binary file format.
CONFIRM	A confirmation is required before any action is performed on a file.
COPY	.
DELETE	.
IGNORE_ DIRECTORIES	.
HEADER	.
MULTIPLE	.
NOSEARCHLIST	
NO_RLF	.
PRT	.
QUALIFIED	The extended file search qualifiers are to be used.
RETAIN_ CDT	.
RETAIN_ RDT	.
RECORD	Use record mode rather than block mode.
SEQUENTIAL	.
UPDATE	Open files with update allowed.

The flag values have symbolic names of the form UFSA$M_*flag* and UFSA$V_ *flag*.

ufsa-type

VMS Usage: **code**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies the structure of the source file specification.

The following table lists the types of file specification argument which can be passed.

Type	Structure
INDEX	A binary index passed by reference (not implemented yet).
LNKLST	Linked list of strings passed by reference.
STRING	String passed by descriptor.

The type values have a symbolic name of the form UFSA$C_*type*.

By default, STRING is assumed.

log-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating logging behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Description

The SWRK_PROCESS_FILE routine ...

See also

SWRK_PURGE_FILE Template

The SWRK_PURGE_FILE routine ...

Format

status=*SWRK_PURGE_FILE(file-spec [,dflt-file-spec] [,rltd-file-spec] [,keep] [,user-success-clb] [,user-error-clb] [,user-confirm-clb] [,user-arg] [,rslt-file-spec] [,context] [,extended-flags] [,ufsa-type] [,log-flags])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

String containing the OpenVMS Record Management Services (RMS) file specification of the files to be purged. The file-spec argument is the address of a descriptor pointing to the file specification. If the specification includes wildcards, each file that matches the specification is purged. The string must not contain more than 255 characters. Any string class is supported.

On Alpha systems, set the LIB$M_FIL_LONG_NAMES bit in the flags argument for strings longer than 255 characters in length.

dflt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Default file specification of the files to be purged. The dflt-file-spec argument is the address of a descriptor pointing to the default file specification. This is an optional argument; if the argument is omitted, the default is the null string. Any string class is supported.

See the OpenVMS Record Management Services Reference Manual for information about default file specifications.

rltd-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Related file specification of the files to be purged. The rltd-file-spec argument is the address of a descriptor pointing to the related file specification. Any string class is supported. This is an optional argument; if the argument is omitted, the default is the null string.

Input file parsing is used. See the OpenVMS Record Management Services Reference Manual for information on related file specifications and input file parsing.

The related file specification is useful when you are processing lists of file specifications. Unspecified portions of the file specification are inherited from the last file processed.

keep

VMS Usage: **integer**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies the maximum number of versions of the specified files to be retained in the directory. If you do not include this argument of the suppied value is zero, all but the highest numbered version of the specified files are deleted from the directory.

user-success-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied success routine that is called after a file is successfully purged.

The success routine can be used to display a log of the files that were purged.

user-error-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied error routine that is called when an error is detected.

The error routine returns a success/fail value that is used to determine if more files should be purged.

user-confirm-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**

mechanism: **by value**

User supplied confirm routine that is called before each file is purged. The value returned by the confirm routine determines whether or not the file will be purged. The confirm routine can be used to select specific files for purge based on criteria such as expiration date, size, and so on.

user-arg

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

User supplied argument that is passed to the error, success, and confirm routines each time they are called. Whatever mechanism is used to pass user-arg is also used to pass it to the routines. This is an optional argument; if the argument is omitted, zero is passed by value.

rslt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

String into which the RMS resultant file specification of the last file processed is written. The rslt-file-spec argument is the address of a descriptor pointing to the resultant name.

If present, rslt-file-spec is used to store the file specification passed to the user supplied routines, instead of a default class S, type T string. Therefore, this argument should be specified when the user supplied routines are used and those routines require a descriptor type other than class S, type T. Any string class is supported.

If you specify one or more of the user supplied action routines, the descriptor used to pass rslt-file-spec must be:

- Of the same class as the descriptor required by the filespec argument of any action routines. For example, VAX Ada requires a class SB descriptor for string arguments to Ada routines but will use a class A descriptor by default when calling external routines. Refer to your language manual to determine the proper descriptor class to use.
- (Alpha only) Of the same form as the descriptor required by the filespec argument of all action routines. For example, if the filespec argument of an action routine uses a 64-bit descriptor, then the rslt-file-spec argument must also use a 64-bit descriptor.

context

VMS Usage: **context**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

A longword integer variable into which the routine stores a context value for use by future calls. The context argument is an unsigned longword integer containing the address of the context. This variable must be set to zero before the first call. You can use the same context argument from one call to another. This argument is used to retain the context when processing multiple input files. Portions of file specifications that the user does not specify may be inherited from the last files processed because the file contexts are retained in this argument. You must not change the value of context in subsequent calls.

extended-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating extended behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Flag	Usage
BINARY	Force binary file types into binary file format.
CONFIRM	A confirmation is required before any action is performed on a file.
COPY	.
DELETE	.
IGNORE_ DIRECTORIES	.
HEADER	.
MULTIPLE	.
NOSEARCHLIST	
NO_RLF	.
PRT	.
QUALIFIED	The extended file search qualifiers are to be used.
RETAIN_ CDT	.
RETAIN_ RDT	.
RECORD	Use record mode rather than block mode.
SEQUENTIAL	.
UPDATE	Open files with update allowed.

The flag values have symbolic names of the form UFSA$M_*flag* and UFSA$V_ *flag*.

ufsa-type

VMS Usage: **code**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies the structure of the source file specification.

The following table lists the types of file specification argument which can be passed.

Type	Structure
INDEX	A binary index passed by reference (not implemented yet).
LNKLST	Linked list of strings passed by reference.
STRING	String passed by descriptor.

The type values have a symbolic name of the form UFSA$C_*type*.

By default, STRING is assumed.

log-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating logging behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Description

The SWRK_PURGE_FILE routine ...

See also

SWRK_PUT_OUTPUT_FAOZ Template

The SWRK_PUT_OUTPUT_FAOZ routine ...

Format

status=*SWRK_PUT_OUTPUT_FAOZ(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_PUT_OUTPUT_FAOZ routine ...

See also

SWRK_PUT_OUTPUT_FAO Template

The SWRK_PUT_OUTPUT_FAO routine ...

Format

status=*SWRK_PUT_OUTPUT_FAO(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_PUT_OUTPUT_FAO routine ...

See also

SWRK_PUT_OUTPUT_PRINTF Template

The SWRK_PUT_OUTPUT_PRINTF routine ...

Format

status=*SWRK_PUT_OUTPUT_PRINTF(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_PUT_OUTPUT_PRINTF routine ...

See also

SWRK_PUT_OUTPUT Template

The SWRK_PUT_OUTPUT routine ...

Format

status=*SWRK_PUT_OUTPUT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_PUT_OUTPUT routine ...

See also

SWRK_PUT_RECORD Template

The SWRK_PUT_RECORD routine ...

Format

status=*SWRK_PUT_RECORD(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_PUT_RECORD routine ...

See also

SWRK_RANDOM_NUMBER Template

The SWRK_RANDOM_NUMBER routine ...

Format

status=*SWRK_RANDOM_NUMBER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_RANDOM_NUMBER routine ...

See also

SWRK_READ_SERVER_MSG_ITMLST Template

The SWRK_READ_SERVER_MSG_ITMLST routine ...

Format

status=*SWRK_READ_SERVER_MSG_ITMLST(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_READ_SERVER_MSG_ITMLST routine ...

See also

SWRK_READ_SERVER_MSG_RECORD Template

The SWRK_READ_SERVER_MSG_RECORD routine ...

Format

status=*SWRK_READ_SERVER_MSG_RECORD(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_READ_SERVER_MSG_RECORD routine ...

See also

SWRK_RECEIVE_CLIENT_A Template

The SWRK_RECEIVE_CLIENT_A routine ...

Format

status=*SWRK_RECEIVE_CLIENT_A(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_RECEIVE_CLIENT_A routine ...

See also

SWRK_RECEIVE_CLIENT Template

The SWRK_RECEIVE_CLIENT routine ...

Format

status=*SWRK_RECEIVE_CLIENT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_RECEIVE_CLIENT routine ...

See also

SWRK_RECEIVE_SERVER_A Template

The SWRK_RECEIVE_SERVER_A routine ...

Format

status=*SWRK_RECEIVE_SERVER_A(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_RECEIVE_SERVER_A routine ...

See also

SWRK_RECEIVE_SERVER Template

The SWRK_RECEIVE_SERVER routine ...

Format

status=*SWRK_RECEIVE_SERVER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_RECEIVE_SERVER routine ...

See also

SWRK_RECYCLE_SERVICE_CHANNEL Template

The SWRK_RECYCLE_SERVICE_CHANNEL routine ...

Format

status=*SWRK_RECYCLE_SERVICE_CHANNEL(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_RECYCLE_SERVICE_CHANNEL routine ...

See also

SWRK_REGISTER_EVENT Template

The SWRK_REGISTER_EVENT routine ...

Format

status=*SWRK_REGISTER_EVENT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_REGISTER_EVENT routine ...

See also

SWRK_RELEASE_SUBCONTEXT Template

The SWRK_RELEASE_SUBCONTEXT routine ...

Format

status=*SWRK_RELEASE_SUBCONTEXT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_RELEASE_SUBCONTEXT routine ...

See also

SWRK_REMOVE_OBJECT Template

The SWRK_REMOVE_OBJECT routine ...

Format

status=*SWRK_REMOVE_OBJECT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_REMOVE_OBJECT routine ...

See also

SWRK_REMOVE_QUEUE_HEAD Template

The SWRK_REMOVE_QUEUE_HEAD routine ...

Format

status=*SWRK_REMOVE_QUEUE_HEAD(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_REMOVE_QUEUE_HEAD routine ...

See also

SWRK_REMOVE_QUEUE_TAIL Template

The SWRK_REMOVE_QUEUE_TAIL routine ...

Format

status=*SWRK_REMOVE_QUEUE_TAIL(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_REMOVE_QUEUE_TAIL routine ...

See also

SWRK_RENAME_FILE Template

The SWRK_RENAME_FILE routine ...

Format

status=*SWRK_RENAME_FILE(old-file-spec, new-file-spec [,dflt-file-spec] [,rltd-file-spec] [,flags] [,user-success-clb] [,user-error-clb] [,user-confirm-clb] [,user-arg] [,old-rslt-file-spec] [,new-rslt-file-spec] [,context] [,extended-flags] [,ufsa-type] [,log-flags])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

old-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

File specification of the files to be renamed. The old-file-spec argument is the address of a descriptor pointing to the old file specification. The specification may include wildcards, in which case each file that matches the specification will be renamed. The string must not contain more than 255 characters. Any string class is supported.

new-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

File specification for the new file names. The new-file-spec argument is the address of a descriptor pointing to the new file specification.

This specification need not be complete; fields omitted or specified by using the wildcard character (*) will be filled in from the existing file's name using the same rules as for the equivalent DCL command. The string must not contain more than 255 characters. Any string class is supported.

dflt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**

mechanism: **by descriptor**

Default file specification of the files to be renamed. The dflt-file-spec argument is the address of a descriptor pointing to the default file specification. This is an optional argument; if the argument is omitted, the default is the null string. Any string class is supported.

See the OpenVMS Record Management Services Reference Manual for information about default file specifications.

rltd-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Related file specification of the files to be renamed. The rltd-file-spec argument is the address of a descriptor pointing to the related file specification. Any string class is supported. This is an optional argument; if the argument is omitted, the default is the null string.

Input file parsing is used. See the OpenVMS Record Management Services Reference Manual for information on related file specifications and input file parsing.

The related file specification is useful when you are processing lists of file specifications. Unspecified portions of the file specification are inherited from the last file processed.

flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating optional behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

user-success-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied success routine that is called after a file is successfully renamed.

The success routine can be used to display a log of the files that were renamed.

user-error-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied error routine that is called when an error is detected.

The error routine returns a success/fail value that is used to determine if more files should be renamed.

user-confirm-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied confirm routine that is called before each file is renamed. The value returned by the confirm routine determines whether or not the file will be renamed. The confirm routine can be used to select specific files for rename based on criteria such as expiration date, size, and so on.

user-arg

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

User supplied argument that is passed to the error, success, and confirm routines each time they are called. Whatever mechanism is used to pass user-arg is also used to pass it to the routines. This is an optional argument; if the argument is omitted, zero is passed by value.

old-rslt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

String into which the old resultant file specification of the last file processed is copied. This is an optional argument. If present, it is used to store the file specification passed to the user supplied routines instead of a default class S, type T string. Any string class is supported.

If you are specifying one or more of the action routine arguments, be sure that the descriptor class used to pass the resultant name is the same as the descriptor class required by the action routine. For example, VAX Ada requires a class SB descriptor for string arguments to Ada routines, but will use a class A descriptor by default when calling external routines.

Refer to your language manual to determine the proper descriptor class to use.

new-rslt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

String into which new OpenVMS RMS resultant file specification of the last file processed is written. The new-rslt-file-spec argument is the address of a descriptor pointing to the new name. This is an optional argument. If present, it is used to store the file specification passed to the user supplied routines instead of a class S, type T string. Any string class is supported.

If you are specifying one or more of the action routine arguments, be sure that the descriptor class used to pass the resultant name is the same as the descriptor class required by the action routine. For example, VAX Ada requires a class SB descriptor for string arguments to Ada routines, but will use a class A descriptor by default when calling external routines. Refer to your language manual to determine the proper descriptor class to use.

context

VMS Usage: **context**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

A longword integer variable into which the routine stores a context value for use by future calls. The context argument is an unsigned longword integer containing the address of the context. This variable must be set to zero before the first call. You can use the same context argument from one call to another. This argument is used to retain the context when processing multiple input files. Portions of file specifications that the user does not specify may be inherited from the last files processed because the file contexts are retained in this argument. You must not change the value of context in subsequent calls.

extended-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating extended behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Flag	Usage
BINARY	Force binary file types into binary file format.
CONFIRM	A confirmation is required before any action is performed on a file.
COPY	.
DELETE	.
IGNORE_ DIRECTORIES	.
HEADER	.
MULTIPLE	.
NOSEARCHLIST	
NO_RLF	.

Flag	Usage
PRT	.
QUALIFIED	The extended file search qualifiers are to be used.
RETAIN_CDT	.
RETAIN_RDT	.
RECORD	Use record mode rather than block mode.
SEQUENTIAL	.
UPDATE	Open files with update allowed.

The flag values have symbolic names of the form UFSA$M_*flag* and UFSA$V_*flag*.

ufsa-type

VMS Usage: **code**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies the structure of the source file specification.

The following table lists the types of file specification argument which can be passed.

Type	Structure
INDEX	A binary index passed by reference (not implemented yet).
LNKLST	Linked list of strings passed by reference.
STRING	String passed by descriptor.

The type values have a symbolic name of the form UFSA$C_*type*.

By default, STRING is assumed.

log-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating logging behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Description

The SWRK_RENAME_FILE routine ...

See also

SWRK_RENAME_OBJECT Template

The SWRK_RENAME_OBJECT routine ...

Format

status=*SWRK_RENAME_OBJECT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_RENAME_OBJECT routine ...

See also

SWRK_REPORT_STATISTICS Template

The SWRK_REPORT_STATISTICS routine ...

Format

status=*SWRK_REPORT_STATISTICS(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_REPORT_STATISTICS routine ...

See also

SWRK_RESUME_SERVER Template

The SWRK_RESUME_SERVER routine ...

Format

status=*SWRK_RESUME_SERVER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_RESUME_SERVER routine ...

See also

SWRK_RETURN_DATE_TIME Template

The SWRK_RETURN_DATE_TIME routine ...

Format

status=*SWRK_RETURN_DATE_TIME(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_RETURN_DATE_TIME routine ...

See also

SWRK_ROLLBACK Template

The SWRK_ROLLBACK routine ...

Format

status=*SWRK_ROLLBACK(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_ROLLBACK routine ...

See also

SWRK_RUN_DETACHED_JOB Template

The SWRK_RUN_DETACHED_JOB routine ...

Format

status=*SWRK_RUN_DETACHED_JOB(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_RUN_DETACHED_JOB routine ...

See also

SWRK_SAVE_DATE_TIME Template

The SWRK_SAVE_DATE_TIME routine ...

Format

status=*SWRK_SAVE_DATE_TIME(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SAVE_DATE_TIME routine ...

See also

SWRK_SCAN_INDEX
Scans an index

The SWRK_SCAN_INDEX routine scans an index.

Format

status=*SWRK_SCAN_INDEX(index_root, action_procedure)*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

index_root

VMS Usage: **structure**
type: **INDEX_ROOT structure**
access: **read only**
mechanism: **by reference**

Specifies the address of the index root from which to scan the index elements.

action_procedure

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User-supplied action routine called by SWRK_SCAN_INDEX for each element in the index. The **action_procedure** must return a success status for SWRK_ SCAN_INDEX to continue scanning.

For more information, see “Call Format for an Action Procedure” in the Description section.

Description

The SWRK_SCAN_INDEX routine scans in an index in key order and calls an action procedure for each element found. Calling SWRK_SCAN_INDEX with an empty index root causes no calls to the action procedure.

See also

SWRK_DELETE_INDEX
SWRK_LOOKUP_INDEX
SWRK_SCAN_INDEX

SWRK_SCAN_LNKLST Template

The SWRK_SCAN_LNKLST routine ...

Format

status=*SWRK_SCAN_LNKLST(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SCAN_LNKLST routine ...

See also

SWRK_SEND_CLIENT_A Template

The SWRK_SEND_CLIENT_A routine ...

Format

status=*SWRK_SEND_CLIENT_A(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SEND_CLIENT_A routine ...

See also

SWRK_SEND_CLIENT Template

The SWRK_SEND_CLIENT routine ...

Format

status=*SWRK_SEND_CLIENT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SEND_CLIENT routine ...

See also

SWRK_SEND_MAIL Template

The SWRK_SEND_MAIL routine ...

Format

status=*SWRK_SEND_MAIL(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SEND_MAIL routine ...

See also

SWRK_SEND_SERVER_A Template

The SWRK_SEND_SERVER_A routine ...

Format

status=*SWRK_SEND_SERVER_A(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SEND_SERVER_A routine ...

See also

SWRK_SEND_SERVER_MESSAGE_A Template

The SWRK_SEND_SERVER_MESSAGE_A routine ...

Format

status=*SWRK_SEND_SERVER_MESSAGE_A(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SEND_SERVER_MESSAGE_A routine ...

See also

SWRK_SEND_SERVER_MESSAGE Template

The SWRK_SEND_SERVER_MESSAGE routine ...

Format

status=*SWRK_SEND_SERVER_MESSAGE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SEND_SERVER_MESSAGE routine ...

See also

SWRK_SEND_SERVER Template

The SWRK_SEND_SERVER routine ...

Format

status=*SWRK_SEND_SERVER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SEND_SERVER routine ...

See also

SWRK_SETUP_BLOCK Template

The SWRK_SETUP_BLOCK routine ...

Format

status=*SWRK_SETUP_BLOCK(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SETUP_BLOCK routine ...

See also

SWRK_SETUP_FILE Template

The SWRK_SETUP_FILE routine ...

Format

status=*SWRK_SETUP_FILE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SETUP_FILE routine ...

See also

SWRK_SETUP_MSG_VEC
Load a message vector with a single message

This routine loads a message vector with a single message.

Format

status=**SWRK_SETUP_MSG_VEC***(message_vector, status [, value...])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

status

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies the OpenVMS message id of the message to be logged. See the $GETMSG system service for more details.

values

VMS Usage: **varying_arg**
type: **longword (signed)**
access: **read only**
mechanism: **by value**

Specifies optional expressions whose resultant types correspond to conversion specifications given in the OpenVMS message text.

If no conversion specifications are given, you may omit the output sources. Otherwise, the function call must have exactly as many output sources as there are conversion spec- ifications, and the conversion specifications must match the types of the output sources.

Conversion specifications are matched to output sources in left-to-right order. Excess output pointers, if any, are ignored.

message_vector

VMS Usage: **cntrlblk**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Specifies the message vector which will be loaded with a message.

Description

This procedure loads the message specified by the status and value arguments into a message vector. The number of entries filled is based on the facility code of the status. See the *OpenVMS System Services Reference Manual* for details about message vector layouts. Unused entries within the message vector are set to zero.

To build a message vector with more than one message, a call is made to SWRK_SETUP_MSG_VEC to setup the last message line, followed by calls to SWRK_PREFIX_MSG_VEC to setup earlier lines. The last call to SWRK_PREFIX_MSG_VEC inserts the first message line.

SWRK_SET_CONTEXT Template

The SWRK_SET_CONTEXT routine ...

Format

status=*SWRK_SET_CONTEXT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_CONTEXT routine ...

See also

SWRK_SET_COUNTER Template

The SWRK_SET_COUNTER routine ...

Format

status=*SWRK_SET_COUNTER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_COUNTER routine ...

See also

SWRK_SET_DEFAULT Template

The SWRK_SET_DEFAULT routine ...

Format

status=*SWRK_SET_DEFAULT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_DEFAULT routine ...

See also

SWRK_SET_EXIT_COMMAND Template

The SWRK_SET_EXIT_COMMAND routine ...

Format

status=*SWRK_SET_EXIT_COMMAND(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_EXIT_COMMAND routine ...

See also

SWRK_SET_FILE_SECURITY Template

The SWRK_SET_FILE_SECURITY routine ...

Format

status=*SWRK_SET_FILE_SECURITY(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_FILE_SECURITY routine ...

See also

SWRK_SET_LOG_DEFAULTS Template

The SWRK_SET_LOG_DEFAULTS routine ...

Format

status=*SWRK_SET_LOG_DEFAULTS(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_LOG_DEFAULTS routine ...

See also

SWRK_SET_LOG_MAIL_SUBJECT Template

The SWRK_SET_LOG_MAIL_SUBJECT routine ...

Format

status=*SWRK_SET_LOG_MAIL_SUBJECT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_LOG_MAIL_SUBJECT routine ...

See also

SWRK_SET_PRIVILEGES_OFF Template

The SWRK_SET_PRIVILEGES_OFF routine ...

Format

status=*SWRK_SET_PRIVILEGES_OFF(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_PRIVILEGES_OFF routine ...

See also

SWRK_SET_PRIVILEGES_ON Template

The SWRK_SET_PRIVILEGES_ON routine ...

Format

status=*SWRK_SET_PRIVILEGES_ON(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_PRIVILEGES_ON routine ...

See also

SWRK_SET_SUBPROCESS_STYLE Template

The SWRK_SET_SUBPROCESS_STYLE routine ...

Format

status=*SWRK_SET_SUBPROCESS_STYLE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_SUBPROCESS_STYLE routine ...

See also

SWRK_SET_SYMBOL_INTEGER Template

The SWRK_SET_SYMBOL_INTEGER routine ...

Format

status=*SWRK_SET_SYMBOL_INTEGER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_SYMBOL_INTEGER routine ...

See also

SWRK_SET_SYMBOL_STRING Template

The SWRK_SET_SYMBOL_STRING routine ...

Format

status=*SWRK_SET_SYMBOL_STRING(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_SYMBOL_STRING routine ...

See also

SWRK_SET_TIMER_LONG Template

The SWRK_SET_TIMER_LONG routine ...

Format

status=*SWRK_SET_TIMER_LONG(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_TIMER_LONG routine ...

See also

SWRK_SET_TIMER Template

The SWRK_SET_TIMER routine ...

Format

status=*SWRK_SET_TIMER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_TIMER routine ...

See also

SWRK_SET_TIMER_SHORT Template

The SWRK_SET_TIMER_SHORT routine ...

Format

status=*SWRK_SET_TIMER_SHORT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_TIMER_SHORT routine ...

See also

SWRK_SET_UIC Template

The SWRK_SET_UIC routine ...

Format

status=*SWRK_SET_UIC(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_UIC routine ...

See also

SWRK_SET_USERNAME Template

The SWRK_SET_USERNAME routine ...

Format

status=*SWRK_SET_USERNAME(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SET_USERNAME routine ...

See also

SWRK_START_SERVER Template

The SWRK_START_SERVER routine ...

Format

status=*SWRK_START_SERVER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_START_SERVER routine ...

See also

SWRK_START_STATISTIC Template

The SWRK_START_STATISTIC routine ...

Format

status=*SWRK_START_STATISTIC(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_START_STATISTIC routine ...

See also

SWRK_STOP_SERVER Template

The SWRK_STOP_SERVER routine ...

Format

status=*SWRK_STOP_SERVER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_STOP_SERVER routine ...

See also

SWRK_STOP_STATISTIC Template

The SWRK_STOP_STATISTIC routine ...

Format

status=*SWRK_STOP_STATISTIC(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_STOP_STATISTIC routine ...

See also

SWRK_STRIP_OBJECT Template

The SWRK_STRIP_OBJECT routine ...

Format

status=*SWRK_STRIP_OBJECT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_STRIP_OBJECT routine ...

See also

SWRK_SUBMIT_BATCH_JOB_CONTEXT Template

The SWRK_SUBMIT_BATCH_JOB_CONTEXT routine ...

Format

status=*SWRK_SUBMIT_BATCH_JOB_CONTEXT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SUBMIT_BATCH_JOB_CONTEXT routine ...

See also

SWRK_SUSPEND_SERVER Template

The SWRK_SUSPEND_SERVER routine ...

Format

status=*SWRK_SUSPEND_SERVER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_SUSPEND_SERVER routine ...

See also

SWRK_TOUCH_FILE Template

The SWRK_TOUCH_FILE routine ... _[element-or-template]...

Format

status=*SWRK_TOUCH_FILE(file-spec [,dflt-file-spec] [,rltd-file-spec] [,create-flag] [,user-success-clb] [,user-error-clb] [,user-confirm-clb] [,user-arg] [,rslt-file-spec] [,context] [,extended-flags] [,ufsa-type] [,log-flags])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

String containing the OpenVMS Record Management Services (RMS) file specification of the files to be touched. The file-spec argument is the address of a descriptor pointing to the file specification. If the specification includes wildcards, each file that matches the specification is touched. The string must not contain more than 255 characters. Any string class is supported.

On Alpha systems, set the LIB$M_FIL_LONG_NAMES bit in the flags argument for strings longer than 255 characters in length.

dflt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Default file specification of the files to be touched. The dflt-file-spec argument is the address of a descriptor pointing to the default file specification. This is an optional argument; if the argument is omitted, the default is the null string. Any string class is supported.

See the OpenVMS Record Management Services Reference Manual for information about default file specifications.

rltd-file-spec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Related file specification of the files to be touched. The rltd-file-spec argument is the address of a descriptor pointing to the related file specification. Any string class is supported. This is an optional argument; if the argument is omitted, the default is the null string.

Input file parsing is used. See the OpenVMS Record Management Services Reference Manual for information on related file specifications and input file parsing.

The related file specification is useful when you are processing lists of file specifications. Unspecified portions of the file specification are inherited from the last file processed.

create-flag

VMS Usage: **flag**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies whether the file is created if not already present.

user-success-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied success routine that is called after a file is successfully touched.

The success routine can be used to display a log of the files that were touched.

user-error-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied error routine that is called when an error is detected.

The error routine returns a success/fail value that is used to determine if more files should be touched.

user-confirm-clb

VMS Usage: **procedure**
type: **procedure value**
access: **function call (before return)**
mechanism: **by value**

User supplied confirm routine that is called before each file is touched. The value returned by the confirm routine determines whether or not the file will be touched. The confirm routine can be used to select specific files for touching based on criteria such as expiration date, size, and so on.

user-arg

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

User supplied argument that is passed to the error, success, and confirm routines each time they are called. Whatever mechanism is used to pass user-arg is also used to pass it to the routines. This is an optional argument; if the argument is omitted, zero is passed by value.

rslt-file-spec

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

String into which the RMS resultant file specification of the last file processed is written. The rslt-file-spec argument is the address of a descriptor pointing to the resultant name.

If present, rslt-file-spec is used to store the file specification passed to the user supplied routines, instead of a default class S, type T string. Therefore, this argument should be specified when the user supplied routines are used and those routines require a descriptor type other than class S, type T. Any string class is supported.

If you specify one or more of the user supplied action routines, the descriptor used to pass rslt-file-spec must be:

- Of the same class as the descriptor required by the filespec argument of any action routines. For example, VAX Ada requires a class SB descriptor for string arguments to Ada routines but will use a class A descriptor by default when calling external routines. Refer to your language manual to determine the proper descriptor class to use.
- (Alpha only) Of the same form as the descriptor required by the filespec argument of all action routines. For example, if the filespec argument of an action routine uses a 64-bit descriptor, then the rslt-file-spec argument must also use a 64-bit descriptor.

context

VMS Usage: **context**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

A longword integer variable into which the routine stores a context value for use by future calls. The context argument is an unsigned longword integer containing the address of the context. This variable must be set to zero before the first call. You can use the same context argument from one call to another. This argument is used to retain the context when processing multiple input files. Portions of file

specifications that the user does not specify may be inherited from the last files processed because the file contexts are retained in this argument. You must not change the value of context in subsequent calls.

extended-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating extended behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Flag	Usage
BINARY	Force binary file types into binary file format.
CONFIRM	A confirmation is required before any action is performed on a file.
COPY	.
DELETE	.
IGNORE_ DIRECTORIES	.
HEADER	.
MULTIPLE	.
NOSEARCHLIST	
NO_RLF	.
PRT	.
QUALIFIED	The extended file search qualifiers are to be used.
RETAIN_ CDT	.
RETAIN_ RDT	.
RECORD	Use record mode rather than block mode.
SEQUENTIAL	.
UPDATE	Open files with update allowed.

The flag values have symbolic names of the form UFSA$M_*flag* and UFSA$V_ *flag*.

ufsa-type

VMS Usage: **code**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Specifies the structure of the source file specification.

The following table lists the types of file specification argument which can be passed.

Type	Structure
INDEX	A binary index passed by reference (not implemented yet).
LNKLST	Linked list of strings passed by reference.
STRING	String passed by descriptor.

The type values have a symbolic name of the form UFSA$C_*type*.

By default, STRING is assumed.

log-flags

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword of flag bits designating logging behavior. The flags argument is the address of an unsigned longword containing the flag bits. This is an optional argument; if omitted, the default is that all flags are clear.

Description

The SWRK_TOUCH_FILE routine ...

See also

SWRK_TRANSLATE_LOGICAL Template

The SWRK_TRANSLATE_LOGICAL routine ...

Format

status=*SWRK_TRANSLATE_LOGICAL(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_TRANSLATE_LOGICAL routine ...

See also

SWRK_TRIGGER_AND_WAIT_FOR_EVENT Template

The SWRK_TRIGGER_AND_WAIT_FOR_EVENT routine ...

Format

status=*SWRK_TRIGGER_AND_WAIT_FOR_EVENT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_TRIGGER_AND_WAIT_FOR_EVENT routine ...

See also

SWRK_TRIGGER_EVENT Template

The SWRK_TRIGGER_EVENT routine ...

Format

status=*SWRK_TRIGGER_EVENT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_TRIGGER_EVENT routine ...

See also

SWRK_UNLOCK_RESOURCE Template

The SWRK_UNLOCK_RESOURCE routine ...

Format

status=*SWRK_UNLOCK_RESOURCE(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_UNLOCK_RESOURCE routine ...

See also

SWRK_UPDATE_OBJECT Template

The SWRK_UPDATE_OBJECT routine ...

Format

status=*SWRK_UPDATE_OBJECT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_UPDATE_OBJECT routine ...

See also

SWRK_UPDATE_RECORD Template

The SWRK_UPDATE_RECORD routine ...

Format

status=*SWRK_UPDATE_RECORD(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_UPDATE_RECORD routine ...

See also

SWRK_UPDATE_VARIABLE_INTEGER Template

The SWRK_UPDATE_VARIABLE_INTEGER routine ...

Format

status=*SWRK_UPDATE_VARIABLE_INTEGER(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_UPDATE_VARIABLE_INTEGER routine ...

See also

SWRK_UPDATE_VARIABLE_STRING Template

The SWRK_UPDATE_VARIABLE_STRING routine ...

Format

status=*SWRK_UPDATE_VARIABLE_STRING(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_UPDATE_VARIABLE_STRING routine ...

See also

SWRK_WAIT_FOR_EVENT Template

The SWRK_WAIT_FOR_EVENT routine ...

Format

status=*SWRK_WAIT_FOR_EVENT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_WAIT_FOR_EVENT routine ...

See also

SWRK_WRITE_JOB Template

The SWRK_WRITE_JOB routine ...

Format

status=*SWRK_WRITE_JOB(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRK_WRITE_JOB routine ...

See also

Part III

SWRKDCL_ Reference Section

This section contains detailed discussions about routines provided by the SysWorks SWRKSHR runtime library for DCL interfaces.

SWRKDCL_ATTACH Template

The SWRKDCL_ATTACH routine ...

Format

status=*SWRKDCL_ATTACH(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRKDCL_ATTACH routine ...

See also

SWRKDCL_ATTACH_IDENTIFICATION Template

The SWRKDCL_ATTACH_IDENTIFICATION routine ...

Format

status=*SWRKDCL_ATTACH_IDENTIFICATION(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRKDCL_ATTACH_IDENTIFICATION routine ...

See also

SWRKDCL_ATTACH_PARENT Template

The SWRKDCL_ATTACH_PARENT routine ...

Format

status=*SWRKDCL_ATTACH_PARENT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRKDCL_ATTACH_PARENT routine ...

See also

SWRKDCL_DEFINE_KEY Template

The SWRKDCL_DEFINE_KEY routine ...

Format

status=*SWRKDCL_DEFINE_KEY(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRKDCL_DEFINE_KEY routine ...

See also

SWRKDCL_EXIT
DCL Interface EXIT Command

The DCL Interface EXIT Command routine implements the EXIT command for a DCL utility.

Format (in CLD)

```
!
!  EXIT command
!

define verb exit

    routine swrkdcl_exit
```

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Description

The SWRKDCL_EXIT routine implements the EXIT command for a DCL utility. The syntax above is used in a Command Definition Utility source (file type .CLD) to cause the EXIT command to exit the utility.

See Also

SWRK_DCL_HANDLER

SWRKDCL_HELP
DCL Interface HELP Command

The DCL Interface HELP Command routine implements the HELP command for a DCL utility.

Format (in CLD)

```
!
!  HELP command
!

define verb help

    routine swrkdcl_help
```

Returns

VMS Usage:	**cond_value**
type:	**integer (unsigned)**
access:	**write only**
mechanism:	**by value in R0**

Description

The SWRKDCL_HELP routine implements the HELP command for a DCL utility. The syntax above is used in a Command Definition Utility source (file type .CLD) to cause the HELP command to invoke OpenVMS help.

See Also

SWRK_DCL_HANDLER

SWRKDCL_NOCOMMAND Template

The SWRKDCL_NOCOMMAND routine ...

Format

status=*SWRKDCL_NOCOMMAND(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRKDCL_NOCOMMAND routine ...

See also

SWRKDCL_SHOW_CONTEXT Template

The SWRKDCL_SHOW_CONTEXT routine ...

Format

status=*SWRKDCL_SHOW_CONTEXT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRKDCL_SHOW_CONTEXT routine ...

See also

SWRKDCL_SHOW_DEFAULT Template

The SWRKDCL_SHOW_DEFAULT routine ...

Format

status=*SWRKDCL_SHOW_DEFAULT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRKDCL_SHOW_DEFAULT routine ...

See also

SWRKDCL_SHOW_VERSION Template

The SWRKDCL_SHOW_VERSION routine displays the utility version.

Format

status=*SWRKDCL_SHOW_VERSION ()*

Returns

VMS Usage:	**cond_value**
type:	**integer (unsigned)**
access:	**write only**
mechanism:	**by value in R0**

Description

The SWRKDCL_SHOW_DEFAULT routine displays the utility version. At this time, the utility version is always the SysWorks major version.

See also

SWRKDCL_SHOW_SUBCONTEXT Template

The SWRKDCL_SHOW_SUBCONTEXT routine ...

Format

status=*SWRKDCL_SHOW_SUBCONTEXT(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRKDCL_SHOW_SUBCONTEXT routine ...

See also

SWRKDCL_SPAWN Template

The SWRKDCL_SPAWN routine ...

Format

status=*SWRKDCL_SPAWN(arg1, [arg2])*

Returns

VMS Usage: **cond_value**
type: **integer (unsigned)**
access: **write only**
mechanism: **by value in R0**

Arguments

arg1

VMS Usage: **arg1_type**
type: **integer (unsigned)**
access: **read only**
mechanism: **by reference**

Specifies arg1...

arg2

VMS Usage: **arg2_type**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Specifies arg2...

Description

The SWRKDCL_SPAWN routine ...

See also

Part IV

Macros Reference Section

This section contains detailed discussions about macros provided by SysWorks for Macro-32 programming.

VECTOR
Shareable Image Vector Macro

The Shareable Image Vector macro is a simple method of creating shareable images for OpenVMS VAX.

Format

VECTOR *name, mode, narg, image, seq*

Arguments

name

The name of an entry point for the shareable image. The keyword `old` may be used to indicate an entry point which is no longer used.

mode

The mode in which the routine should execute. The valid values are:

- **subroutine**

  Indicates that the routine is called by a VAX BSB, BSW or JSB instruction instead of the more formal CALLG or CALLS instruction.

- **user**

  Indicates that the routine is called at the current mode of the process. This is the default value.

- **exec**

  Indicates that the routine is to be called in executive mode. This mode of entry point should not be called from kernel mode.

- **kernel**

  Indicates that the routine should be called in kernel mode.

By default, the mode is user. If an elevated mode such as executive or kernel is used, the **narg** argument must also be supplied.

narg

Indicates the number of arguments which must be supplied for executive and kernel mode routines. This argument is not required or used for subroutine or user mode entry points.

image

Indicates that the code associated with the entry point resides in another shareable image. This argument has as its value the file specification of the shareable image in which another entry point for the routine exists and the code resides.

seq

Indicates the sequence number of the entry point within the transfer vectors and is used as a double check that vectors remain in the correct order and are not accidentally removed. This argument must be supplied for each use of the **vector** macro, and must start at 1 for the first vector, 2 for the second vector and so on.

Description

The vector macro is used in a Macro-32 source which becomes the base for building a shareable image for both OpenVMS VAX and OpenVMS Alpha.

It is recommended that entry points be added alphabetically within releases. Each release appears sequentially within the source so that the entry points never move once a release has been made.

Examples

SWRKSHR.MAR

```
        .title  swrkshr Shareable image root for OpenVMS
        .ident  "V3.4"

;++
;
;  Module:
;       SWRKSHR
;
;  Purpose:
;       SWRKSHR shareable image root for OpenVMS
;
;  Copyright:
;       Copyright © 1987 - 2005 Corpita Pty Ltd
;       15 Bedford Street, Collingwood 3066, Australia
;
;  History:
;      05-Mar-1991 by Simon L. Jackson
;           New version
;
;--

        .library "swrk_macro_lib"

module swrkshr

; Special aliases

        alias   swrk_find_object_lcl    swrk_find_object

; Vectors
```

```
        vector  swrk_extended_file_search               seq=001
        vector  swrk_call_remote                        seq=002
        vector  swrk_dequeue_server_message             seq=003
        vector  swrk_enqueue_server_message             seq=004
        vector  old                                     seq=005 ; swrk_initia
        vector  swrk_inquire_server                     seq=006
        vector  swrk_read_server_msg_itmlst             seq=007
        vector  swrk_read_server_msg_record             seq=008
        vector  swrk_resume_server                      seq=009
        vector  swrk_send_server_message                seq=010
        vector  swrk_send_server_message_a              seq=011
        vector  swrk_start_server                       seq=012
        vector  swrk_stop_server                        seq=013
        vector  swrk_suspend_server                     seq=014

        vector  swrk_dcl_handler                        seq=015
        vector  swrkdcl_exit                            seq=016
        vector  swrkdcl_nocommand                       seq=017
        vector  swrkdcl_help                            seq=018
        vector  swrk_handle_keyword                     seq=019
        vector  swrk_handle_qualifiers                  seq=020

        vector  swrk_delete_lnklst                      seq=021
        vector  old                                     seq=022 ; swrk_delete

; Added on 19-Dec-1992
        vector  swrk_copy_file                          seq=023
        vector  swrk_delete_file                        seq=024
        vector  swrk_do_command                         seq=025
        vector  swrk_purge_file                         seq=026

; Added on 31-May-1993
        vector  swrk_append_string_to_lnklst            seq=027

; Added on 03-Sep-1993
        vector  swrk_dump_image_info                    seq=028
        vector  swrk_establish                          seq=029
        vector  swrk_exception_handler                  seq=030
        vector  swrk_append_lnklst_to_lnklst            seq=031
        vector  swrk_append_item_to_lnklst              seq=032

; Added on 15-Oct-1994
        vector  swrk_close_output                       seq=033
        vector  swrk_handle_image_vector,mode=subroutine seq=034
        vector  swrk_open_output                        seq=035
        vector  swrk_put_output                         seq=036

; Added on 19-Nov-1994
        vector  swrk_get_cur_pfx                        seq=037

; Version V3.0
; Added on 23-Sep-1995 & 06-Nov-1995
        vector  swrk_allocate_memory                    seq=038
        vector  swrk_crash_image                        seq=039
        vector  swrk_deallocate_memory                  seq=040
```

```
; Added on 18-Feb-1996
        vector  swrk_copy_msg_vec                       seq=041
        vector  swrk_get_date                           seq=042
        vector  swrk_get_date_time                      seq=043
        vector  swrk_lock_resource                      seq=044
        vector  swrk_log_image_finish                   seq=045
        vector  swrk_log_image_start                    seq=046
        vector  swrk_log_msg_vec                        seq=047
        vector  swrk_log_rms_fab                        seq=048
        vector  swrk_log_rms_fab_2                      seq=049
        vector  swrk_log_rms_rab                        seq=050
        vector  swrk_log_status                         seq=051
        vector  swrk_log_text                           seq=052
        vector  swrk_prefix_msg_vec                     seq=053
        vector  swrk_register_event                     seq=054
        vector  swrk_return_date_time                   seq=055
        vector  swrk_save_date_time                     seq=056
        vector  swrk_setup_msg_vec                      seq=057
        vector  swrk_trigger_event                      seq=058
        vector  swrk_unlock_resource                    seq=059
        vector  swrk_wait_for_event                     seq=060

; Added on 13-Apr-1996, 20-Jul-1996 & 03-Aug-1996
        vector  swrk_allocate_bits                      seq=061
        vector  swrk_deallocate_bits                    seq=062
        vector  swrk_declare_bitmap                     seq=063
        vector  swrk_get_subcontext                     seq=064
        vector  swrk_get_username                       seq=065
        vector  swrk_random_number                      seq=066
        vector  swrk_release_subcontext                 seq=067
        vector  swrk_translate_logical                  seq=068
        vector  swrk_trigger_and_wait_for_event         seq=069

; Version V3.1 & V3.2
; Added on 22-Aug-1996
        vector  swrk_get_context                        seq=070
        vector  swrk_set_context                        seq=071

; Added on 01-Oct-1996
        vector  swrk_get_computer_node                  seq=072
        vector  swrkdcl_attach                          seq=073
        vector  swrkdcl_attach_identification           seq=074
        vector  swrkdcl_attach_parent                   seq=075
        vector  swrkdcl_spawn                           seq=076
        vector  swrkdcl_define_key                      seq=077

; Version V3.1
; Added on 16-Nov-1996
        vector  swrk_dump_memory_begin                  seq=078
        vector  swrk_dump_memory_cell                   seq=079
        vector  swrk_dump_memory_end                    seq=080
        vector  swrk_get_image_name                     seq=081

; Added on 28-Dec-1996
        vector  swrk_log_printf                         seq=082
```

```
; Version V3.2 - sorted on 22-Aug-1998
        vector  swrkdcl_show_context                    seq=083
        vector  old                                     seq=084 ; swrk_attach
        vector  swrk_binary_search                      seq=085
        vector  swrk_cancel_job                         seq=086
        vector  swrk_cancel_timer                       seq=087
        vector  swrk_close_and_detach_job               seq=088
        vector  swrk_close_and_spawn_job                seq=089
        vector  swrk_close_and_submit_job               seq=090
        vector  old                                     seq=091 ; swrk_close_
        vector  swrk_create_file_fdl                    seq=092
        vector  swrk_cvt_context_description            seq=093
        vector  swrk_cvt_context_type                   seq=094
        vector  swrk_declare_client                     seq=095
        vector  swrk_declare_counter                    seq=096
        vector  swrk_declare_server                     seq=097
        vector  swrk_declare_statistics                 seq=098
        vector  swrk_delete_index                       seq=099
        vector  old                                     seq=100 ; swrk_detach
        vector  swrk_dump_memory_range                  seq=101
        vector  swrk_display_help                       seq=102
        vector  swrk_find_object                        seq=103
        vector  swrk_find_object_id                     seq=104
        vector  swrk_find_variable_string               seq=105
        vector  swrk_fixup_string                       seq=106
        vector  swrk_get_fab_file_spec                  seq=107
        vector  swrk_get_counter                        seq=108
        vector  swrk_increment_counter                  seq=109
        vector  swrk_insert_index                       seq=110
        vector  swrk_insert_queue_head                  seq=111
        vector  swrk_insert_queue_tail                  seq=112
        vector  swrk_log_msg_vec_2                      seq=113
        vector  swrk_lookup_index                       seq=114
        vector  old                                     seq=115 ; swrk_open_s
        vector  swrk_open_job                           seq=116
        vector  swrk_receive_client                     seq=117
        vector  swrk_receive_client_a                   seq=118
        vector  swrk_receive_server                     seq=119
        vector  swrk_receive_server_a                   seq=120
        vector  swrk_recycle_service_channel            seq=121
        vector  swrk_remove_queue_head                  seq=122
        vector  swrk_remove_queue_tail                  seq=123
        vector  swrk_scan_index                         seq=124
        vector  swrk_send_client                        seq=125
        vector  swrk_send_client_a                      seq=126
        vector  swrk_send_server                        seq=127
        vector  swrk_send_server_a                      seq=128
        vector  swrk_set_counter                        seq=129
        vector  swrk_set_timer                          seq=130
        vector  swrk_set_timer_long                     seq=131
        vector  swrk_set_timer_short                    seq=132
        vector  swrk_start_statistic                    seq=133
        vector  swrk_stop_statistic                     seq=134
        vector  swrk_strip_properties                   seq=135
        vector  swrk_write_job                          seq=136
```

```
; Version V3.2 - added on 31-Aug-1998, 19-Sep-1998, 02-Oct-1998
        vector  old                                     seq=137 ; swrkdcl_con
        vector  swrk_get_last_msg_logged                seq=138
        vector  swrk_get_log_defaults                   seq=139
        vector  swrk_get_object_keys                    seq=140
        vector  swrk_log_fao                            seq=141
        vector  swrk_log_faoz                           seq=142
        vector  swrk_log_output                         seq=143
        vector  swrk_log_output_2                       seq=144
        vector  swrk_parse_log_file                     seq=145
        vector  swrk_put_output_fao                     seq=146
        vector  swrk_put_output_faoz                    seq=147
        vector  swrk_put_output_printf                  seq=148
        vector  swrk_report_statistics                  seq=149
        vector  swrk_set_log_defaults                   seq=150
        vector  swrk_set_uic                            seq=151
        vector  swrk_set_username                       seq=152

; Version V3.2-1 - added on 16-Oct-1998
        vector  swrkdcl_show_subcontext                 seq=153
        vector  swrk_exit_image                         seq=154
        vector  swrk_get_default                        seq=155
        vector  swrk_set_default                        seq=156
        vector  swrk_set_exit_command                   seq=157

; Version V3.2-1 - added on 19-Nov-1998
        vector  swrk_cvt_context_command                seq=158

; Version V3.2-1 - added on 28-Nov-1998
        vector  swrk_check_resource                     seq=159
        vector  swrk_parse_directory                    seq=160
        vector  swrk_parse_file                         seq=161

; Version V3.2-1 - added on 18-Dec-1998
        vector  swrk_cvt_lnklst_to_string               seq=162
        vector  swrk_scan_lnklst                        seq=163

; Version V3.2-1 - added on 26-Dec-1998
        vector  swrk_cvt_index_to_string                seq=164
        vector  swrk_initialize_trace                   seq=165

; Version V3.2-1 - added on 30-Dec-1998
        vector  swrk_convert_date                       seq=166
        vector  swrk_convert_date_time                  seq=167
        vector  swrk_convert_time                       seq=168

; Version V3.2-1 - added on 02-Jan-1999
        vector  swrkdcl_show_default                    seq=169
        vector  swrk_cvt_trace_description              seq=170
        vector  swrk_get_cur_env                        seq=171
        vector  swrk_set_subprocess_style               seq=172

; Version V3.2-1 - added on 16-Jan-1999
        vector  swrk_initialize_image                   seq=173
        vector  swrk_send_mail                          seq=174

; Version V3.2-1 - added on 22-Jan-1999
        vector  swrk_clear_context                      seq=175
        vector  swrk_set_log_mail_subject               seq=176

; Version V3.2-1 - added on 06-Feb-1999
        vector  swrk_find_associations                  seq=177

; Version V3.2-1 - added on 22-Mar-1999
        vector  swrk_dump_process_info                  seq=178
```

```
; Version V3.2-1 - added on 19-Jun-1999
        vector  swrk_get_symbol_integer                 seq=179
        vector  swrk_get_symbol_string                  seq=180
        vector  swrk_set_symbol_integer                 seq=181
        vector  swrk_set_symbol_string                  seq=182

; Version V3.2-1 - added on 03-Jul-1999
        vector  swrk_cvt_scope_type_to_code             seq=183
        vector  swrk_find_variable_integer              seq=184
        vector  swrk_log_rms_file_sts                   seq=185
        vector  swrk_update_variable_integer            seq=186
        vector  swrk_update_variable_string             seq=187

; Version V3.2-1 - added on 07-Aug-1999
        vector  swrk_cvt_scope_type_to_ctl              seq=188
        vector  swrk_get_scope_context                  seq=189

; Version V3.2-1 - added on 03-Jan-2000
        vector  swrk_add_object                         seq=190
        vector  swrk_build_objt_record                  seq=191
        vector  old                                     seq=192 ; swrk_delete
        vector  old                                     seq=193 ; swrk_find_o
        vector  old                                     seq=194 ; swrk_get_ob
        vector  old                                     seq=195 ; swrk_put_ob
        vector  swrk_remove_object                      seq=196
        vector  old                                     seq=197 ; swrk_update
        vector  swrk_update_object                      seq=198

; Version V3.2-1 - added on 06-Mar-2000
        vector  swrk_get_statistics_info                seq=199

; Version V3.2-1 - added on 15-Mar-2000
        vector  swrk_backup_file                        seq=200
        vector  swrk_close_file                         seq=201
        vector  swrk_close_input                        seq=202
        vector  swrk_copy_string                        seq=203
        vector  swrk_delete_record                      seq=204
        vector  swrk_empty_string                       seq=205
        vector  swrk_get_record                         seq=206
        vector  swrk_get_input                          seq=207
        vector  swrk_move_file                          seq=208
        vector  swrk_open_file                          seq=209
        vector  swrk_open_input                         seq=210
        vector  swrk_print_file                         seq=211
        vector  swrk_put_record                         seq=212
        vector  swrk_rename_file                        seq=213
        vector  swrk_update_record                      seq=214
        vector  swrk_set_file_security                  seq=215

; Version V3.2-1 - added on 05-Apr-2000
        vector  swrk_find_resource                      seq=216
        vector  swrk_get_log_qual_file                  seq=217
        vector  swrk_manage_file                        seq=218
        vector  swrk_touch_file                         seq=219

; Version V3.2-1 - added on 17-Jun-2000
        vector  swrk_dump_memory_context                seq=220

; Version V3.3 - added on 17-Jul-2000
        vector  swrk_get_architecture                   seq=221

; Version V3.3 - added on 28-Jul-2000
        vector  swrk_match_lnklst                       seq=222

; Version V3.3 - added on 14-Aug-2000
        vector  swrk_find_class                         seq=223

; Version V3.3 - added on 17-Aug-2000
        vector  swrk_bitmap_operation_2                 seq=224
        vector  swrk_bitmap_operation_3                 seq=225
```

```
; Version V3.3 - added on 16-Oct-2000
        vector  swrk_initialize_logging                 seq=226

; Version V3.3 - added on 13-Mar-2001
        vector  swrk_create_file                        seq=227
        vector  swrk_get_file_spec                      seq=228
        vector  swrk_setup_file                         seq=229

; Version V3.3 - added on 15-Jun-2001
        vector  swrk_match_string                       seq=230

; Version V3.3 - added on 19-Oct-2001
        vector  swrk_rename_object                      seq=231

; Version V3.4 - added on 14-Nov-2001
        vector  swrkdcl_show_version                    seq=232
        vector  swrk_get_msg_vec_item                   seq=233
        vector  swrk_process_file                       seq=234

; Version V3.4 - added on 11-Jan-2002
        vector  swrk_add_association                    seq=235
        vector  swrk_remove_association         seq=236
        vector  swrk_set_output                         seq=237

; Version V3.4 - added on 20-Mar-2002
        vector  swrk_clear_context_item                 seq=238
        vector  swrk_set_context_item                   seq=239

; Version V3.4 - added on 01-Aug-2002
        vector  swrk_checksum_file                      seq=240
        vector  swrk_checksum_file_1                    seq=241

; Version V3.4 - added on 14-Oct-2002
        vector  swrk_crc_file                           seq=242
        vector  swrk_crc_file_1                         seq=243

; Version V3.4 - added on 21-Oct-2002
        vector  swrk_check_numcd                        seq=244
        vector  swrk_cvt_int_to_numcd                   seq=245
        vector  swrk_cvt_numcd_to_int                   seq=246
        vector  swrk_cvt_num_to_cd                      seq=247
        vector  swrk_cvt_num_to_numcd                   seq=248

; Version V3.4 - added on 19-Nov-2002
        vector  swrk_execute_method                     seq=249

; Version V3.4 - added on 02-Jan-2004
        vector  gzclose         image=zlibshr           seq=250
        vector  gzerror         image=zlibshr           seq=251
        vector  gzgets          image=zlibshr           seq=252
        vector  gzopen          image=zlibshr           seq=253

; Version V3.4 - added on 14-Feb-2004
        vector  swrk_check_method                       seq=254
        vector  swrk_cvt_string_to_lnklst               seq=255

; Version V3.4 - added on 22-Feb-2004
        vector  swrk_append_string                      seq=256

; Version V3.4 - added on 02-Jun-2004
        vector  swrk_add_exit_handler                   seq=257
        vector  swrk_rundown_image                      seq=258
```

```
; Version V3.4 - added on 18-Jul-2004
        vector  swrk_close_document                     seq=259
        vector  swrk_create_document                    seq=260
        vector  swrk_put_document_field_d               seq=261
        vector  swrk_put_document_field_z               seq=262
        vector  swrk_put_document_line                  seq=263
        vector  swrk_put_document_line_fao              seq=264
        vector  swrk_put_document_line_faoz             seq=265
        vector  swrk_put_document_line_printf           seq=266
        vector  swrk_setup_document                     seq=267
        vector  swrk_put_line                           seq=268
        vector  swrk_put_line_fao                       seq=269
        vector  swrk_put_line_faoz                      seq=270
        vector  swrk_put_line_printf                    seq=271

; Version V3.4 - added on 09-Aug-2004
        vector  swrk_compare_string                     seq=272
        vector  swrk_concat_string                      seq=273
        vector  swrk_equals_string                      seq=274

; Version V3.4 - added on 02-Sep-2004
        vector  gzputs          image=zlibshr           seq=275
        vector  gzread          image=zlibshr           seq=276
        vector  gzwrite         image=zlibshr           seq=277

; Version V3.4 - added on 10-Dec-2004
        vector  swrk_cvt_string_to_owner                seq=278

; Version V3.4 - added on 11-Jan-2005
        vector  swrk_log_file                           seq=279
        vector  swrk_log_file_2                         seq=280

; Version V3.4 - added on 14-Jul-2005
        vector  swrk_get_file_type                      seq=281
        vector  swrk_set_file_type                      seq=282

; Reservations - added on 19-Nov-1998

end module
```

This example is a Macro-32 source which is used to build a shareable image. The logical name `swrk_macro_lib` is defined by the SysWorks startup command procedures.

SWRKSHR.OPT

```
!++
!
!  Linker_options:
!       SWRKSHR-VAX
!
!  Purpose:
!       Linker options file for the SWRKSHR shareable image for OpenVMS VAX.
!
!  Copyright:
!       Copyright © 1987 - 2009 Corpita Pty Ltd
!       15 Bedford Street, Collingwood 3066, Australia
!
!  History:
!      05-Dec-1989 by Simon L. Jackson
!           Initial version
!
!--

gsmatch = always,3,5
```

```
cluster = transfer
collect = transfer,swrk_vectors

cluster = swrk_section_1
collect = swrk_section_1,-
    swrk_message_vector,-
    rdb$transaction_handle,-
    swrk_trace_data_rec

psect_attr = rdb$transaction_handle, pic, ovr, gbl, noshr
psect_attr = swrk_message_vector, pic, ovr, gbl, noshr
psect_attr = swrk_trace_data_rec, pic, ovr, gbl, noshr

cluster = swrk_section_2
collect = swrk_section_2,-
    swrk_global_data

psect_attr = swrk_global_data, pic, con, gbl, noshr

swrk_lib_dir:swrkshr

swrk_object_lib/include=(-
    rdb$transaction_handle,-
    swrk_global_data,-
    swrk_messages,-
    swrk_message_vector)

swrk_sft_dir:swrkshrprv/share

swrk_object_lib/include=swrk_quadword

swrk_object_lib/library
swrk_symbol_lib/library

sys$system:imgdef.stb/selective_search
sys$system:sys.stb/selective_search

universal = rdb$transaction_handle
universal = swrk_message_vector

universal = swrk_command_line
universal = swrk_ctx_depth

universal = swrk_cur_app
universal = swrk_cur_env
universal = swrk_cur_grp
universal = swrk_cur_pfx
universal = swrk_cur_scp
universal = swrk_cur_typ
universal = swrk_cur_typ_cod
universal = swrk_cur_usr
universal = swrk_cur_var
universal = swrk_cur_vsn

universal = swrk_dir_cod
universal = swrk_dir_dir
universal = swrk_dir_set
```

```
universal = swrk_dir_aib
universal = swrk_dir_aij
universal = swrk_dir_ail
universal = swrk_dir_bck
universal = swrk_dir_cdd
universal = swrk_dir_dat
universal = swrk_dir_doc
universal = swrk_dir_gen
universal = swrk_dir_jnl
universal = swrk_dir_kit
universal = swrk_dir_lcl
universal = swrk_dir_lib
universal = swrk_dir_rda
universal = swrk_dir_rdb
universal = swrk_dir_ruj
universal = swrk_dir_run
universal = swrk_dir_scr
universal = swrk_dir_sft
universal = swrk_dir_snp
universal = swrk_dir_src
universal = swrk_dir_tps
universal = swrk_dir_tst
universal = swrk_dir_wrk

universal = swrk_explicit_link_flg
universal = swrk_fil_gen
universal = swrk_fil_msghlp
universal = swrk_fil_tst

universal = swrk_lbr_hlp
universal = swrk_lbr_img
universal = swrk_lbr_mac
universal = swrk_lbr_mms
universal = swrk_lbr_obj
universal = swrk_lbr_sym
universal = swrk_lbr_txt

universal = swrk_lnm_cdd
universal = swrk_lnm_dir
universal = swrk_lnm_fil
universal = swrk_lnt_ctx
universal = swrk_message_vector
universal = swrk_output_file

universal = swrk_sdc_aib
universal = swrk_sdc_aij
universal = swrk_sdc_ail
universal = swrk_sdc_bck
universal = swrk_sdc_cdd
universal = swrk_sdc_dat
universal = swrk_sdc_doc
universal = swrk_sdc_gen
universal = swrk_sdc_jnl
universal = swrk_sdc_kit
universal = swrk_sdc_lcl
universal = swrk_sdc_lib
universal = swrk_sdc_rda
universal = swrk_sdc_rdb
universal = swrk_sdc_ruj
universal = swrk_sdc_run
universal = swrk_sdc_scr
universal = swrk_sdc_sft
universal = swrk_sdc_snp
universal = swrk_sdc_src
universal = swrk_sdc_tps
universal = swrk_sdc_tst
universal = swrk_sdc_wrk
```

```
universal = swrk_subctx_depth
universal = swrk_sub_output_noast
universal = swrk_sub_output_rtn
universal = swrk_tpu_result_string
universal = swrk_vmsu_ctx
universal = swrk_plrn_ctx

universal = swrk_lnm_rot
universal = swrk_rot_ail
universal = swrk_rot_dat
universal = swrk_rot_lib
universal = swrk_rot_src
universal = swrk_rot_tps

universal = swrk_arc_cod
universal = swrk_arc_typ
universal = swrk_arc_usg
universal = swrk_cur_arc
universal = swrk_dir_arc

universal = swrk_cls_cod
universal = swrk_cls_set
universal = swrk_cls_mgr
universal = swrk_cls_rep
universal = swrk_cls_usr
universal = swrk_id_acc
universal = swrk_id_mgr
universal = swrk_id_own
universal = swrk_id_rep
universal = swrk_id_usr

universal = swrk_trace_data

universal = swrk_dcl_input_routine

universal = swrk_rab_mbc
universal = swrk_rab_mbf

universal = swrk_sdc_set

universal = swrk_dir_srt
universal = swrk_sdc_srt

universal = swrk_dir_dev

universal = swrk_dir_www
universal = swrk_rot_www
universal = swrk_sdc_www

universal = swrk_dir_adalib
universal = swrk_sdc_adalib
```

This example is a Linker options file which is used to build a shareable image for OpenVMS VAX. Note that only the first three (non-comment) lines are important for all shareable images - the remaining lines are specific to the linking requirements of this specific image.

```
SWRKSHR-ALPHA.OPT
```

```
!++
!
!  Linker_options:
!       SWRKSHR-ALPHA
!
!  Purpose:
!       Linker options file for the SWRKSHR shareable image for OpenVMS Alpha
!
!  Copyright:
!       Copyright © 1995 - 2009 Corpita Pty Ltd
!       15 Bedford Street, Collingwood 3066, Australia
!
!  History:
!      07-Oct-1995 by Simon L. Jackson
!           Initial Alpha version
!
!--

gsmatch = always,3,5

cluster = swrk_section_1
collect = swrk_section_1,-
    rdb$transaction_handle,-
    swrk_message_vector,-
    swrk_trace_data_rec

psect_attr = rdb$transaction_handle, pic, ovr, gbl, noshr, mod
psect_attr = swrk_message_vector, pic, ovr, gbl, noshr, mod
psect_attr = swrk_trace_data_rec, pic, ovr, gbl, noshr, mod

cluster = swrk_section_2
collect = swrk_section_2,-
    swrk_global_data

psect_attr = swrk_global_data, pic, con, gbl, noshr, mod

swrk_lib_dir:swrkshr

swrk_object_lib/include=(-
    rdb$transaction_handle,-
    swrk_global_data,-
    swrk_messages,-
    swrk_message_vector)

swrk_sft_dir:swrkshrprv/share

swrk_object_lib/library
swrk_symbol_lib/library

sys$share:decc$shr/shareable
swrk_lib_dir:swrksys/include=sys$base_image

! %INCLUDE SWRK_AIL_DIR:SWRKSHR

symbol_vector=(-
    rdb$transaction_handle=psect,-
    swrk_message_vector=psect,-
    swrk_trace_data_rec=psect)
```

```
symbol_vector=(-
    swrk_command_line=data,-
    swrk_ctx_depth=data,-
    swrk_cur_app=data,-
    swrk_cur_env=data,-
    swrk_cur_grp=data,-
    swrk_cur_pfx=data,-
    swrk_cur_scp=data,-
    swrk_cur_typ=data,-
    swrk_cur_typ_cod=data,-
    swrk_cur_usr=data,-
    swrk_cur_var=data,-
    swrk_cur_vsn=data,-
    swrk_dir_ail=data,-
    swrk_dir_cdd=data,-
    swrk_dir_cod=data,-
    swrk_dir_dat=data,-
    swrk_dir_dir=data,-
    swrk_dir_doc=data,-
    swrk_dir_gen=data,-
    swrk_dir_kit=data,-
    swrk_dir_lib=data,-
    swrk_dir_run=data,-
    swrk_dir_scr=data,-
    swrk_dir_sft=data,-
    swrk_dir_src=data,-
    swrk_dir_tst=data,-
    swrk_dir_wrk=data,-
    swrk_explicit_link_flg=data,-
    swrk_fil_gen=data,-
    swrk_fil_msghlp=data,-
    swrk_fil_tst=data,-
    swrk_objt_cluster=private_data,-
    swrk_objt_network=private_data,-
    swrk_objt_computer_node=private_data,-
    swrk_objt_security_domain=private_data,-
    swrk_objt_site=private_data,-
    swrk_objt_system=private_data,-
    swrk_objt_tuning_domain=private_data,-
    swrk_lbr_hlp=data,-
    swrk_lbr_img=data,-
    swrk_lbr_mac=data,-
    swrk_lbr_mms=data,-
    swrk_lbr_obj=data,-
    swrk_lbr_sym=data,-
    swrk_lbr_txt=data,-
    swrk_lnm_cdd=data,-
    swrk_lnm_dir=data,-
    swrk_lnm_fil=data,-
    swrk_lnt_ctx=data,-
    swrk_message_vector=data,-
    swrk_output_file=data,-
    swrk_sdc_ail=data,-
    swrk_sdc_cdd=data,-
    swrk_sdc_dat=data,-
    swrk_sdc_doc=data,-
    swrk_sdc_gen=data,-
    swrk_sdc_kit=data,-
    swrk_sdc_lib=data,-
    swrk_sdc_run=data,-
    swrk_sdc_scr=data,-
    swrk_sdc_sft=data,-
    swrk_sdc_src=data,-
    swrk_sdc_tst=data,-
    swrk_sdc_wrk=data,-
    swrk_subctx_depth=data,-
```

```
    swrk_sub_output_noast=data,-
    swrk_sub_output_rtn=data,-
    swrk_tpu_result_string=data,-
    swrk_vmsu_ctx=data)

! Added 14-Nov-1998 by SLJ
symbol_vector=(-
    swrk_dir_aib=data,-
    swrk_dir_aij=data,-
    swrk_dir_bck=data,-
    swrk_dir_jnl=data,-
    swrk_dir_lcl=data,-
    swrk_dir_set=data,-
    swrk_dir_rda=data,-
    swrk_dir_rdb=data,-
    swrk_dir_ruj=data,-
    swrk_dir_snp=data,-
    swrk_dir_tps=data,-
    swrk_sdc_aib=data,-
    swrk_sdc_aij=data,-
    swrk_sdc_bck=data,-
    swrk_sdc_jnl=data,-
    swrk_sdc_lcl=data,-
    swrk_sdc_rda=data,-
    swrk_sdc_rdb=data,-
    swrk_sdc_ruj=data,-
    swrk_sdc_snp=data,-
    swrk_sdc_tps=data)

! Added 28-Nov-1998 by SLJ
symbol_vector=(-
    swrk_plrn_ctx=data)

! Added 09-Dec-1998 by SLJ
symbol_vector=(-
    swrk_lnm_rot=data,-
    swrk_rot_ail=data,-
    swrk_rot_dat=data,-
    swrk_rot_lib=data,-
    swrk_rot_src=data,-
    swrk_rot_tps=data)

! Added 15-Dec-1998 by SLJ
symbol_vector=(-
    swrk_arc_cod=data,-
    swrk_arc_typ=data,-
    swrk_arc_usg=data,-
    swrk_cur_arc=data,-
    swrk_dir_arc=data)

! Added 18-Dec-1998 by SLJ
symbol_vector=(-
    swrk_cls_cod=data,-
    swrk_cls_set=data,-
    swrk_cls_mgr=data,-
    swrk_cls_rep=data,-
    swrk_cls_usr=data,-
    swrk_id_acc=data,-
    swrk_id_mgr=data,-
    swrk_id_own=data,-
    swrk_id_rep=data,-
    swrk_id_usr=data)

! Added 28-Dec-1998 by SLJ
symbol_vector=(-
    swrk_trace_data=data)
```

```
! Added 30-Dec-1998 by SLJ
symbol_vector=(-
    swrk_dcl_input_routine=data)

! Added 01-Feb-1999 by SLJ
symbol_vector=(-
    swrk_rab_mbc=data,-
    swrk_rab_mbf=data)

! Added 24-Nov-1999 by SLJ
symbol_vector=(-
    rdb$transaction_handle=data,-
    swrk_sdc_set=data)

! Added 21-Mar-2000 by SLJ
symbol_vector=(-
    swrk_dir_srt=data,-
    swrk_sdc_srt=data)

! Added 17-Jul-2001 by SLJ
symbol_vector=(-
    swrk_dir_dev=data)

! Added 24-Aug-2004 by SLJ
symbol_vector=(-
    swrk_dir_www=data,-
    swrk_rot_www=data,-
    swrk_sdc_www=data)

! Added 28-Apr-2005 by SLJ
symbol_vector=(-
    swrk_dir_adalib=data,-
    swrk_sdc_adalib=data)
```

This example is a Linker options file which is used to build the same shareable image for OpenVMS Alpha. Note the use of the `%APPEND` directive which causes SysWorks to generate an options file in `SWRK_AIL_DIR:SWRKSHR.OPT_INC` based on the source `SWRK_WRK_DIR:SWRKSHR.MAR` and rules which include both the original options file `SWRK_WRK_DIR:SWRKSHR-ALPHA.OPT` and generated options file (`SWRK_AIL_DIR:SWRKSHR.OPT_INC`) as part of the `LINK` command.

`SWRKSHR-ALPHA.OPT_INC`

```
!++
!
!  Linker options include:
!       SWRKSHR
!
!  Purpose:
!       Shareable image vectors.
!
!  History:
!      15-Jul-2005 by SLJ
!           Generated version
!
!--
```

```
symbol_vector=(-
    SWRK_EXTENDED_FILE_SEARCH=procedure,-
    SWRK_CALL_REMOTE=procedure,-
    SWRK_DEQUEUE_SERVER_MESSAGE=procedure,-
    SWRK_ENQUEUE_SERVER_MESSAGE=procedure,-
    OLD=private_procedure,-
    SWRK_INQUIRE_SERVER=procedure,-
    SWRK_READ_SERVER_MSG_ITMLST=procedure,-
    SWRK_READ_SERVER_MSG_RECORD=procedure,-
    SWRK_RESUME_SERVER=procedure,-
    SWRK_SEND_SERVER_MESSAGE=procedure,-
    SWRK_SEND_SERVER_MESSAGE_A=procedure,-
    SWRK_START_SERVER=procedure,-
    SWRK_STOP_SERVER=procedure,-
    SWRK_SUSPEND_SERVER=procedure,-
    SWRK_DCL_HANDLER=procedure,-
    SWRKDCL_EXIT=procedure)

symbol_vector=(-
    SWRKDCL_NOCOMMAND=procedure,-
    SWRKDCL_HELP=procedure,-
    SWRK_HANDLE_KEYWORD=procedure,-
    SWRK_HANDLE_QUALIFIERS=procedure,-
    SWRK_DELETE_LNKLST=procedure,-
    OLD=private_procedure,-
    SWRK_COPY_FILE=procedure,-
    SWRK_DELETE_FILE=procedure,-
    SWRK_DO_COMMAND=procedure,-
    SWRK_PURGE_FILE=procedure,-
    SWRK_APPEND_STRING_TO_LNKLST=procedure,-
    SWRK_DUMP_IMAGE_INFO=procedure,-
    SWRK_ESTABLISH=procedure,-
    SWRK_EXCEPTION_HANDLER=procedure,-
    SWRK_APPEND_LNKLST_TO_LNKLST=procedure,-
    SWRK_APPEND_ITEM_TO_LNKLST=procedure)

symbol_vector=(-
    SWRK_CLOSE_OUTPUT=procedure,-
    SWRK_HANDLE_IMAGE_VECTOR=procedure,-
    SWRK_OPEN_OUTPUT=procedure,-
    SWRK_PUT_OUTPUT=procedure,-
    SWRK_GET_CUR_PFX=procedure,-
    SWRK_ALLOCATE_MEMORY=procedure,-
    SWRK_CRASH_IMAGE=procedure,-
    SWRK_DEALLOCATE_MEMORY=procedure,-
    SWRK_COPY_MSG_VEC=procedure,-
    SWRK_GET_DATE=procedure,-
    SWRK_GET_DATE_TIME=procedure,-
    SWRK_LOCK_RESOURCE=procedure,-
    SWRK_LOG_IMAGE_FINISH=procedure,-
    SWRK_LOG_IMAGE_START=procedure,-
    SWRK_LOG_MSG_VEC=procedure,-
    SWRK_LOG_RMS_FAB=procedure)
```

```
symbol_vector=(-
    SWRK_LOG_RMS_FAB_2=procedure,-
    SWRK_LOG_RMS_RAB=procedure,-
    SWRK_LOG_STATUS=procedure,-
    SWRK_LOG_TEXT=procedure,-
    SWRK_PREFIX_MSG_VEC=procedure,-
    SWRK_REGISTER_EVENT=procedure,-
    SWRK_RETURN_DATE_TIME=procedure,-
    SWRK_SAVE_DATE_TIME=procedure,-
    SWRK_SETUP_MSG_VEC=procedure,-
    SWRK_TRIGGER_EVENT=procedure,-
    SWRK_UNLOCK_RESOURCE=procedure,-
    SWRK_WAIT_FOR_EVENT=procedure,-
    SWRK_ALLOCATE_BITS=procedure,-
    SWRK_DEALLOCATE_BITS=procedure,-
    SWRK_DECLARE_BITMAP=procedure,-
    SWRK_GET_SUBCONTEXT=procedure)

symbol_vector=(-
    SWRK_GET_USERNAME=procedure,-
    SWRK_RANDOM_NUMBER=procedure,-
    SWRK_RELEASE_SUBCONTEXT=procedure,-
    SWRK_TRANSLATE_LOGICAL=procedure,-
    SWRK_TRIGGER_AND_WAIT_FOR_EVENT=procedure,-
    SWRK_GET_CONTEXT=procedure,-
    SWRK_SET_CONTEXT=procedure,-
    SWRK_GET_COMPUTER_NODE=procedure,-
    SWRKDCL_ATTACH=procedure,-
    SWRKDCL_ATTACH_IDENTIFICATION=procedure,-
    SWRKDCL_ATTACH_PARENT=procedure,-
    SWRKDCL_SPAWN=procedure,-
    SWRKDCL_DEFINE_KEY=procedure,-
    SWRK_DUMP_MEMORY_BEGIN=procedure,-
    SWRK_DUMP_MEMORY_CELL=procedure,-
    SWRK_DUMP_MEMORY_END=procedure)

symbol_vector=(-
    SWRK_GET_IMAGE_NAME=procedure,-
    SWRK_LOG_PRINTF=procedure,-
    SWRKDCL_SHOW_CONTEXT=procedure,-
    OLD=private_procedure,-
    SWRK_BINARY_SEARCH=procedure,-
    SWRK_CANCEL_JOB=procedure,-
    SWRK_CANCEL_TIMER=procedure,-
    SWRK_CLOSE_AND_DETACH_JOB=procedure,-
    SWRK_CLOSE_AND_SPAWN_JOB=procedure,-
    SWRK_CLOSE_AND_SUBMIT_JOB=procedure,-
    OLD=private_procedure,-
    SWRK_CREATE_FILE_FDL=procedure,-
    SWRK_CVT_CONTEXT_DESCRIPTION=procedure,-
    SWRK_CVT_CONTEXT_TYPE=procedure,-
    SWRK_DECLARE_CLIENT=procedure,-
    SWRK_DECLARE_COUNTER=procedure)
```

```
symbol_vector=(-
    SWRK_DECLARE_SERVER=procedure,-
    SWRK_DECLARE_STATISTICS=procedure,-
    SWRK_DELETE_INDEX=procedure,-
    OLD=private_procedure,-
    SWRK_DUMP_MEMORY_RANGE=procedure,-
    SWRK_DISPLAY_HELP=procedure,-
    SWRK_FIND_OBJECT=procedure,-
    SWRK_FIND_OBJECT_ID=procedure,-
    SWRK_FIND_VARIABLE_STRING=procedure,-
    SWRK_FIXUP_STRING=procedure,-
    SWRK_GET_FAB_FILE_SPEC=procedure,-
    SWRK_GET_COUNTER=procedure,-
    SWRK_INCREMENT_COUNTER=procedure,-
    SWRK_INSERT_INDEX=procedure,-
    SWRK_INSERT_QUEUE_HEAD=procedure,-
    SWRK_INSERT_QUEUE_TAIL=procedure)

symbol_vector=(-
    SWRK_LOG_MSG_VEC_2=procedure,-
    SWRK_LOOKUP_INDEX=procedure,-
    OLD=private_procedure,-
    SWRK_OPEN_JOB=procedure,-
    SWRK_RECEIVE_CLIENT=procedure,-
    SWRK_RECEIVE_CLIENT_A=procedure,-
    SWRK_RECEIVE_SERVER=procedure,-
    SWRK_RECEIVE_SERVER_A=procedure,-
    SWRK_RECYCLE_SERVICE_CHANNEL=procedure,-
    SWRK_REMOVE_QUEUE_HEAD=procedure,-
    SWRK_REMOVE_QUEUE_TAIL=procedure,-
    SWRK_SCAN_INDEX=procedure,-
    SWRK_SEND_CLIENT=procedure,-
    SWRK_SEND_CLIENT_A=procedure,-
    SWRK_SEND_SERVER=procedure,-
    SWRK_SEND_SERVER_A=procedure)

symbol_vector=(-
    SWRK_SET_COUNTER=procedure,-
    SWRK_SET_TIMER=procedure,-
    SWRK_SET_TIMER_LONG=procedure,-
    SWRK_SET_TIMER_SHORT=procedure,-
    SWRK_START_STATISTIC=procedure,-
    SWRK_STOP_STATISTIC=procedure,-
    SWRK_STRIP_PROPERTIES=procedure,-
    SWRK_WRITE_JOB=procedure,-
    OLD=private_procedure,-
    SWRK_GET_LAST_MSG_LOGGED=procedure,-
    SWRK_GET_LOG_DEFAULTS=procedure,-
    SWRK_GET_OBJECT_KEYS=procedure,-
    SWRK_LOG_FAO=procedure,-
    SWRK_LOG_FAOZ=procedure,-
    SWRK_LOG_OUTPUT=procedure,-
    SWRK_LOG_OUTPUT_2=procedure)
```

```
symbol_vector=(-
    SWRK_PARSE_LOG_FILE=procedure,-
    SWRK_PUT_OUTPUT_FAO=procedure,-
    SWRK_PUT_OUTPUT_FAOZ=procedure,-
    SWRK_PUT_OUTPUT_PRINTF=procedure,-
    SWRK_REPORT_STATISTICS=procedure,-
    SWRK_SET_LOG_DEFAULTS=procedure,-
    SWRK_SET_UIC=procedure,-
    SWRK_SET_USERNAME=procedure,-
    SWRKDCL_SHOW_SUBCONTEXT=procedure,-
    SWRK_EXIT_IMAGE=procedure,-
    SWRK_GET_DEFAULT=procedure,-
    SWRK_SET_DEFAULT=procedure,-
    SWRK_SET_EXIT_COMMAND=procedure,-
    SWRK_CVT_CONTEXT_COMMAND=procedure,-
    SWRK_CHECK_RESOURCE=procedure,-
    SWRK_PARSE_DIRECTORY=procedure)

symbol_vector=(-
    SWRK_PARSE_FILE=procedure,-
    SWRK_CVT_LNKLST_TO_STRING=procedure,-
    SWRK_SCAN_LNKLST=procedure,-
    SWRK_CVT_INDEX_TO_STRING=procedure,-
    SWRK_INITIALIZE_TRACE=procedure,-
    SWRK_CONVERT_DATE=procedure,-
    SWRK_CONVERT_DATE_TIME=procedure,-
    SWRK_CONVERT_TIME=procedure,-
    SWRKDCL_SHOW_DEFAULT=procedure,-
    SWRK_CVT_TRACE_DESCRIPTION=procedure,-
    SWRK_GET_CUR_ENV=procedure,-
    SWRK_SET_SUBPROCESS_STYLE=procedure,-
    SWRK_INITIALIZE_IMAGE=procedure,-
    SWRK_SEND_MAIL=procedure,-
    SWRK_CLEAR_CONTEXT=procedure,-
    SWRK_SET_LOG_MAIL_SUBJECT=procedure)

symbol_vector=(-
    SWRK_FIND_ASSOCIATIONS=procedure,-
    SWRK_DUMP_PROCESS_INFO=procedure,-
    SWRK_GET_SYMBOL_INTEGER=procedure,-
    SWRK_GET_SYMBOL_STRING=procedure,-
    SWRK_SET_SYMBOL_INTEGER=procedure,-
    SWRK_SET_SYMBOL_STRING=procedure,-
    SWRK_CVT_SCOPE_TYPE_TO_CODE=procedure,-
    SWRK_FIND_VARIABLE_INTEGER=procedure,-
    SWRK_LOG_RMS_FILE_STS=procedure,-
    SWRK_UPDATE_VARIABLE_INTEGER=procedure,-
    SWRK_UPDATE_VARIABLE_STRING=procedure,-
    SWRK_CVT_SCOPE_TYPE_TO_CTL=procedure,-
    SWRK_GET_SCOPE_CONTEXT=procedure,-
    SWRK_ADD_OBJECT=procedure,-
    SWRK_BUILD_OBJT_RECORD=procedure,-
    OLD=private_procedure)
```

```
symbol_vector=(-
    OLD=private_procedure,-
    OLD=private_procedure,-
    OLD=private_procedure,-
    SWRK_REMOVE_OBJECT=procedure,-
    OLD=private_procedure,-
    SWRK_UPDATE_OBJECT=procedure,-
    SWRK_GET_STATISTICS_INFO=procedure,-
    SWRK_BACKUP_FILE=procedure,-
    SWRK_CLOSE_FILE=procedure,-
    SWRK_CLOSE_INPUT=procedure,-
    SWRK_COPY_STRING=procedure,-
    SWRK_DELETE_RECORD=procedure,-
    SWRK_EMPTY_STRING=procedure,-
    SWRK_GET_RECORD=procedure,-
    SWRK_GET_INPUT=procedure,-
    SWRK_MOVE_FILE=procedure)

symbol_vector=(-
    SWRK_OPEN_FILE=procedure,-
    SWRK_OPEN_INPUT=procedure,-
    SWRK_PRINT_FILE=procedure,-
    SWRK_PUT_RECORD=procedure,-
    SWRK_RENAME_FILE=procedure,-
    SWRK_UPDATE_RECORD=procedure,-
    SWRK_SET_FILE_SECURITY=procedure,-
    SWRK_FIND_RESOURCE=procedure,-
    SWRK_GET_LOG_QUAL_FILE=procedure,-
    SWRK_MANAGE_FILE=procedure,-
    SWRK_TOUCH_FILE=procedure,-
    SWRK_DUMP_MEMORY_CONTEXT=procedure,-
    SWRK_GET_ARCHITECTURE=procedure,-
    SWRK_MATCH_LNKLST=procedure,-
    SWRK_FIND_CLASS=procedure,-
    SWRK_BITMAP_OPERATION_2=procedure)

symbol_vector=(-
    SWRK_BITMAP_OPERATION_3=procedure,-
    SWRK_INITIALIZE_LOGGING=procedure,-
    SWRK_CREATE_FILE=procedure,-
    SWRK_GET_FILE_SPEC=procedure,-
    SWRK_SETUP_FILE=procedure,-
    SWRK_MATCH_STRING=procedure,-
    SWRK_RENAME_OBJECT=procedure,-
    SWRKDCL_SHOW_VERSION=procedure,-
    SWRK_GET_MSG_VEC_ITEM=procedure,-
    SWRK_PROCESS_FILE=procedure,-
    SWRK_ADD_ASSOCIATION=procedure,-
    SWRK_REMOVE_ASSOCIATION=procedure,-
    SWRK_SET_OUTPUT=procedure,-
    SWRK_CLEAR_CONTEXT_ITEM=procedure,-
    SWRK_SET_CONTEXT_ITEM=procedure,-
    SWRK_CHECKSUM_FILE=procedure)
```

```
symbol_vector=(-
    SWRK_CHECKSUM_FILE_1=procedure,-
    SWRK_CRC_FILE=procedure,-
    SWRK_CRC_FILE_1=procedure,-
    SWRK_CHECK_NUMCD=procedure,-
    SWRK_CVT_INT_TO_NUMCD=procedure,-
    SWRK_CVT_NUMCD_TO_INT=procedure,-
    SWRK_CVT_NUM_TO_CD=procedure,-
    SWRK_CVT_NUM_TO_NUMCD=procedure,-
    SWRK_EXECUTE_METHOD=procedure,-
    GZCLOSE=procedure,-
    GZERROR=procedure,-
    GZGETS=procedure,-
    GZOPEN=procedure,-
    SWRK_CHECK_METHOD=procedure,-
    SWRK_CVT_STRING_TO_LNKLST=procedure,-
    SWRK_APPEND_STRING=procedure)

symbol_vector=(-
    SWRK_ADD_EXIT_HANDLER=procedure,-
    SWRK_RUNDOWN_IMAGE=procedure,-
    SWRK_CLOSE_DOCUMENT=procedure,-
    SWRK_CREATE_DOCUMENT=procedure,-
    SWRK_PUT_DOCUMENT_FIELD_D=procedure,-
    SWRK_PUT_DOCUMENT_FIELD_Z=procedure,-
    SWRK_PUT_DOCUMENT_LINE=procedure,-
    SWRK_PUT_DOCUMENT_LINE_FAO=procedure,-
    SWRK_PUT_DOCUMENT_LINE_FAOZ=procedure,-
    SWRK_PUT_DOCUMENT_LINE_PRINTF=procedure,-
    SWRK_SETUP_DOCUMENT=procedure,-
    SWRK_PUT_LINE=procedure,-
    SWRK_PUT_LINE_FAO=procedure,-
    SWRK_PUT_LINE_FAOZ=procedure,-
    SWRK_PUT_LINE_PRINTF=procedure,-
    SWRK_COMPARE_STRING=procedure)

symbol_vector=(-
    SWRK_CONCAT_STRING=procedure,-
    SWRK_EQUALS_STRING=procedure,-
    GZPUTS=procedure,-
    GZREAD=procedure,-
    GZWRITE=procedure,-
    SWRK_CVT_STRING_TO_OWNER=procedure,-
    SWRK_LOG_FILE=procedure,-
    SWRK_LOG_FILE_2=procedure,-
    SWRK_GET_FILE_TYPE=procedure,-
    SWRK_SET_FILE_TYPE=procedure)
```

This example is the linker options generated from the Macro-32 source for use with the OpenVMS Alpha shareable image.

SWRKSHRPRV.MAR

```
        .title  swrkshrprv      Protected shareable image root for OpenVMS
        .ident  "V3.4"
```

```
;++
;
;  Module:
;       SWRKSHRPRV
;
;  Purpose:
;       Protected shareable image root for OpenVMS
;
;  Copyright:
;       Copyright © 1995 - 2004 Corpita Pty Ltd
;       15 Bedford Street, Collingwood 3066, Australia
;
;  History:
;      26-Jun-1995 by Simon L. Jackson
;           Initial version
;
;--

.library "swrk_macro_lib"

module swrkshrprv

; Special aliases

        alias   swrk_clear_context      swrk_clear_context_base
        alias   swrk_clear_context_item swrk_clear_context_item_base
        alias   swrk_find_object_lcl    swrk_find_object_prx
        alias   swrk_find_resource      swrk_find_resource_prx
        alias   swrk_set_context_item   swrk_set_context_item_base

        alias   swrk_close_zlib         swrk_unsupported
        alias   swrk_get_zlib           swrk_unsupported
        alias   swrk_open_zlib          swrk_unsupported


; Vectors

        vector  old                                                     seq=0
        vector  old                                                     seq=0
        vector  old                                                     seq=0

; Version V3.2 - Sorted on 02-Jun-1998
        vector  old                                                     seq=0
        vector  old                                                     seq=0
        vector  old                                                     seq=0
        vector  old                                                     seq=0
        vector  swrk_submit_batch_job_context   mode=exec       narg=1  seq=0
        vector  old                                                     seq=0

; Version V3.2-1 - added on 29-Jul-1998
        vector  swrk_run_detached_job           mode=exec       narg=5  seq=0

; Version V3.2-1 - added on 26-Dec-1998
        vector  swrk_get_powerhouse_resources   mode=exec       narg=0  seq=0
        vector  swrk_set_privileges_off         mode=exec       narg=0  seq=0
        vector  swrk_set_privileges_on          mode=exec       narg=0  seq=0

; Version V3.2-1 - added on 14-Aug-1999
        vector  swrk_get_process_imagecount     mode=exec       narg=2  seq=0

; Version V3.2-1 - added on 03-Jan-2000
        vector  old                                                     seq=0
        vector  old                                                     seq=0
        vector  old                                                     seq=0
        vector  old                                                     seq=0

; Version V3.3 - added on 11-Jan-2001
        vector  swrk_create_block               mode=exec       narg=3  seq=0
        vector  swrk_delete_block               mode=exec       narg=1  seq=0
        vector  swrk_setup_block                mode=exec       narg=3  seq=0
```

```
; Version V3.3 - added on 13-Mar-2001
        vector  swrk_get_chan_file_spec         mode=exec       narg=3  seq=0

; Version V3.4 - added on 09-Dec-2001
        vector  swrk_add_object_prv             mode=exec       narg=7  seq=0
        vector  swrk_find_class_prv             mode=exec       narg=3  seq=0
        vector  swrk_find_object_prv            mode=exec       narg=7  seq=0
        vector  swrk_find_object_id_prv         mode=exec       narg=7  seq=0
        vector  swrk_find_associations_prv      mode=exec       narg=10 seq=0
        vector  swrk_remove_object_prv          mode=exec       narg=6  seq=0
        vector  swrk_rename_object_prv          mode=exec       narg=7  seq=0
        vector  swrk_update_object_prv          mode=exec       narg=7  seq=0

; Version V3.4 - added on 24-Dec-2001
        vector  swrk_get_msg_vec_item_prv       mode=exec       narg=3  seq=0

; Version V3.4 - added on 19-Mar-2002
        vector  swrk_clear_context_prv          mode=exec       narg=0  seq=0
        vector  swrk_clear_context_item_prv     mode=exec       narg=1  seq=0
        vector  swrk_set_context_item_prv       mode=exec       narg=2  seq=0

; Version V3.4 - added on 27-Jul-2002
        vector  swrk_add_association_prv        mode=exec       narg=7  seq=0
        vector  swrk_remove_association_prv     mode=exec       narg=7  seq=0

; Version V3.4 - added on 09-Aug-2002
        vector  swrk_cvt_scope_type_to_code_prv mode=exec       narg=2  seq=0
        vector  swrk_cvt_scope_type_to_ctl_prv  mode=exec       narg=2  seq=0
        vector  swrk_get_scope_context_prv      mode=exec       narg=4  seq=0
        vector  swrk_invalidate_scope_prv       mode=exec       narg=1  seq=0

; Version V3.4 - added on 01-May-2003
        vector  swrk_initialize_trace_2         mode=exec       narg=0  seq=0

; Version V3.4 - added on 22-Feb-2004
        vector  swrk_check_method_prv           mode=exec       narg=6  seq=0
        vector  swrk_execute_method_a_prv       mode=exec       narg=14 seq=0

; Version V3.4 - added on 01-May-2004
        vector  swrk_detach_swdatabase_all_prv  mode=exec       narg=0  seq=0
        vector  swrk_find_resource_prv          mode=exec       narg=5  seq=0

; Version V3.4 - added on 02-Jun-2004
        vector  swrk_rundown_image_prv          mode=exec               seq=0

; Reservations - added on 28-Nov-1998

end module
```

This example is a Macro-32 source which is used to build a protected shareable image.

```
SWRKSHRPRV.OPT
```

```
!++
!
!  Linker_options:
!       SWRKSHRPRV-VAX
!
!  Purpose:
!       Linker options file for the SWRKSHRPRV shareable image for OpenVMS VA
!
!  Copyright:
!       Copyright © 1995 - 1998 Corpita Pty Ltd
!       15 Bedford Street, Collingwood 3066, Australia
!
!  History:
!      26-Jun-1995 by Simon L. Jackson
!           Initial version
!
!--

gsmatch = always,3,2

cluster = transfer
collect = transfer,swrk_vectors

swrk_lib_dir:swrkshrprv

swrk_object_lib/include=(-
    rdb$transaction_handle,-
    swrk_global_data,-
    swrk_global_data_priv,-
    swrk_messages,-
    swrk_message_vector)

psect_attr = rdb$transaction_handle, noshr, lcl
psect_attr = swrk_message_vector, noshr, lcl

swrk_object_lib/include=(-
    swrk_chgmod_dispatch,-
    swrk_quadword)

swrk_object_lib/library
swrk_symbol_lib/library

sys$system:imgdef.stb/selective_search
sys$system:sys.stb/selective_search
```

This example is a Linker options file which is used to build a protected shareable image for OpenVMS VAX.

Index

S

V