

SysWorks

Application Development Guide

Order Number SWRK-ADG-35

August 2009

This manual describes and provides reference information on how to use SysWorks™ to develop and maintain applications.

Revision/Update Information: This manual supercedes the SysWorks™ V3.4-1 Application Development Guide

Operation System and Version: 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 1987 - 2009.

All Rights Reserved.

Printed in Australia

The following are trademarks of Compaq Computer Corporation: ACMS, ALL-IN-1, AXP, BASIC, Bookreader, CDA, CI, DATATRIEVE, DBMS, DDIF, DEC, DEC ACCESSWORKS, DEC Ada, DEC C, DEC Fortran, DEC Pascal, DECdecision, DECdesign, DECdirect, DECdns, DECdocument, DECdtm, DECforms, DECimage, DECintact, DECmigrate, DECnet, DECnet/OSI, DECset, DECsupport, DECTp, DECwindows, Digital, DTIF, EDT, HSC, MASSBUS, MicroVAX, MicroVAX II, MSCP, OpenVMS, OpenVMS Cluster, RA, StorageWorks, TA, TMSCP, TURBOchannel, ULTRIX, VAX, VAX C, VAX MACRO, VAX-11/780, VAXcluster, VAXELN, VAXft, VAXstation, VIDA, VMS, VMScluster, VT100, and the DIGITAL logo.

PostScript is a registered trademark of Adobe Systems Inc.

Motif is a registered trademark of Open Software Foundation, Inc.

Oracle is a registered trademark, and Oracle CDD/Repository, Oracle CODASYL DBMS, Oracle Expert, Oracle Rally, Oracle Rdb, Oracle Trace and Rdb7 are trademarks of Oracle Corporation.

OSI is a registered trademark of CA Management, Inc

All other trademarks and registered trademarks are the property of their respective holders.

This document was prepared using DECdocument V3.3.

Contents

Send Us Your Comments	vii
Preface	ix
1 Overview	
1.1 Environments	1-1
1.1.1 Common	1-1
1.1.2 Pre-Development	1-1
1.1.3 Development	1-1
1.1.4 Development Testing	1-1
1.1.5 Maintenance	1-1
1.1.6 Maintenance Testing	1-1
1.1.7 Production	1-2
1.1.8 Training	1-2
1.2 Moving between Applications and Environments	1-2
1.2.1 LOGICALS.COM	1-2
1.2.2 ENTER.COM	1-2
1.2.3 EXIT.COM	1-2
2 Logical Names and Logical Name Tables	
2.1 Common and Specific Contexts	2-1
2.1.1 Common	2-1
2.1.2 Specific	2-1
3 Source Control	
3.1 CMS Libraries	3-1
3.2 Application Common CMS Library	3-1
3.3 Classes and Generations	3-1
3.4 Common and Specific CMS Classes	3-4
3.5 Multi-variant CMS Classes	3-4
3.6 Multi-version CMS Classes	3-5
3.7 Multiple CMS Libraries	3-5
3.8 Commands	3-5
3.8.1 BUILD	3-9
3.8.2 CREATE	3-9
3.8.3 DELETE	3-9
3.8.4 DIRECTORY	3-9
3.8.5 EDIT	3-9
3.8.6 FETCH	3-9

3.8.7	RENAME	3-10
3.8.8	REPLACE	3-10
3.8.9	REPORT	3-10
3.8.10	RESERVE	3-10
3.8.11	SHOW	3-10
3.8.12	UNRESERVE	3-10

4 Change Control

5 Compiling and Building

5.1	Compiling Sources	5-1
5.2	Building Targets	5-1
5.2.1	Phases	5-2
5.2.1.1	PRECLEAN	5-2
5.2.1.2	UPDATE	5-3
5.2.1.3	TIDY	5-3
5.2.1.4	FETCH	5-3
5.2.1.5	SETUP	5-3
5.2.1.6	SCAN	5-3
5.2.1.7	RULES	5-5
5.2.1.8	DESCRIP	5-5
5.2.1.9	KITBLD	5-5
5.2.2	Add Extra Phases	5-5

6 Version Control

6.1	BUILD	6-1
6.2	COMPARE	6-1
6.3	CLEANUP	6-2
6.4	DEVELOP	6-3
6.5	MAINTAIN	6-3
6.6	SPLIT	6-4
6.7	MERGE	6-6
6.8	TRIAL	6-7
6.9	RELEASE	6-8
6.10	INSTALL	6-8
6.11	DBCOMPARE	6-9

7 Images

7.1	Creating Images	7-1
7.2	Executable Images	7-2
7.3	Shareable Images	7-2
7.3.1	Transfer Vectors	7-25
7.4	Other Features	7-26

8 Rdb Database and CDD/Repository

8.1	Database	8-1
8.1.1	Physical Database	8-1
8.1.2	Source	8-2
8.1.3	Central Control	8-2
8.1.4	Conventions and Tools	8-2
8.2	Repository	8-3
8.2.1	Fields	8-4
8.2.2	Records	8-4

9 Language Specifics

9.1	ACA Services	9-5
9.1.1	Contexts	9-5
9.1.2	Repositories	9-5
9.2	ACMS	9-6
9.2.1	Applications	9-6
9.2.2	Menus	9-6
9.2.3	Tasks	9-7
9.2.4	Task Groups	9-7
9.3	ADA	9-9
9.4	Basic	9-10
9.5	C	9-11
9.6	C++	9-14
9.7	CDD	9-15
9.8	CDD/Repository	9-16
9.9	Cobol	9-17
9.10	DCL	9-18
9.11	DECdocument	9-19
9.11.1	Graphics	9-20
9.12	DECforms	9-21
9.13	FMS	9-22
9.14	Fortran	9-23
9.15	Macro	9-24
9.16	ObjectBroker	9-25
9.16.1	Contexts	9-25
9.16.2	Repositories	9-25
9.17	Oracle Rdb	9-26
9.18	Pascal	9-27
9.19	Runoff	9-28
9.20	TDMS	9-29
9.20.1	Forms	9-29
9.20.2	Requests	9-29
9.20.3	Request Libraries	9-29

10 Naming Conventions

11 Application Environment Setup

11.1	Directories	11-1
11.2	CMS Library Creation	11-2
11.3	Granting Access for Developers	11-4

Index

Examples

7-1	SWRKSHR.MAR	7-2
7-2	SWRKSHR.OPT	7-9
7-3	SWRKSHR-ALPHA.OPT	7-13
7-4	SWRKSHR-ALPHA.OPT_INC	7-17
7-5	SWRKSHRPRV.MAR	7-23
7-6	SWRKSHRPRV.OPT	7-26

Figures

3-1	SRCCTL menu	3-8
4-1	CHGCTL menu	4-1
6-1	VSNCTL menu	6-2
6-2	VSNCTL Diagrams Legend	6-3
6-3	SPLIT flow	6-5
6-4	MERGE flow	6-6
6-5	TRIAL flow	6-7
6-6	TRIAL flow from FDEV to DEV	6-8
6-7	RELEASE flow	6-9
6-8	INSTALL flow	6-9

Tables

2-1	Specific Context Search Lists	2-1
3-1	Environments, Generations and Classes	3-2
3-2	Use of <i>appl_CMS_PATH</i> AND <i>appl_CMS_VARIANT</i>	3-2
3-3	CMS Command Availability	3-5
5-1	Build Phases	5-2
5-2	SCAN Phase Generated Files	5-4
5-3	RULES Phase Generated Files	5-5
9-1	Language Support Summary	9-1
9-2	ACA Services Context Requirements	9-5
9-3	ACA Services Repository Requirements	9-5
9-4	ACMS Application Requirements	9-6
9-5	ACMS Menu Requirements	9-6
9-6	ACMS Task Requirements	9-7
9-7	ACMS Task Group Requirements	9-8
9-8	ADA Requirements	9-9
9-9	Basic Requirements	9-10
9-10	C Requirements	9-11
9-11	C++ Requirements	9-14
9-12	CDD Requirements	9-15
9-13	CDD/Repository Requirements	9-16
9-14	Cobol Requirements	9-17
9-15	DCL Command Procedure Requirements	9-18

9-16	DECdocument Requirements	9-19
9-17	DECdocument File Name Suffices	9-19
9-18	DECforms Requirements	9-21
9-19	FMS Requirements	9-22
9-20	Fortran Requirements	9-23
9-21	Macro-32 Requirements	9-24
9-22	ObjectBroker Context Requirements	9-25
9-23	ObjectBroker Repository Requirements	9-25
9-24	Oracle Rdb Requirements	9-26
9-25	Pascal Requirements	9-27
9-26	Runoff Requirements	9-28
9-27	TDMS Forms Requirements	9-29
9-28	TDMS Request Requirements	9-29
9-29	TDMS Request Library Requirements	9-29
10-1	Form and Task Types	10-4
10-2	SQL Module Procedure Suffices for Inserting a List of Byte Varying	10-5
10-3	Procedure Types	10-7
10-4	Program Type	10-7
10-5	Structure Types	10-8
10-6	Page Record Suffices	10-9
10-7	Table Record Suffices	10-9
10-8	SQL Module Procedure Suffices for a Singleton Select of List of Byte Varying	10-10
10-9	SQL Module Procedure Suffices for Updating a List of Byte Varying	10-12

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 describes and provides reference information on how to use SysWorks™ to develop and maintain applications.

Intended Audience

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

Conventions

The following conventions are used in this document:

Conventions	Description
<code>Ctrl/X</code>	A sequence such as <code>Ctrl/X</code> indicates that you must hold down the key labeled <code>Ctrl</code> while you press another key or a pointing device button.
<code>[]</code>	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.
<code>{}</code>	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.
<code> </code>	In format descriptions, vertical bars separate the options. If the options are enclosed in brackets (i.e. <code>[]</code>) you can select none, one or all of the choices. If the options are enclosed in braces (i.e. <code>{}</code>) you must choose one of the options listed.
<code>()</code>	In system prompts, parenthesis indicate the list of values one of which may be entered. The values are separated by a forward slash "/"
<code>...</code>	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 <code>Disk quota (0..unlimited)</code> indicates that the keyword <code>unlimited</code> may be used to represent the highest possible disk quota.
<i>italic text</i>	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.
<code>monospace text</code>	Normal monospace text indicates system prompts and output.
<code>bold monospace text</code>	Bold monospace text indicates user responses to system prompts.
<i><code>bold monospace italic text</code></i>	Bold monospace italic text indicates user responses to system prompts which need appropriate value substitution.

Conventions	Description
mouse	The term <i>mouse</i> 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.

This chapter provides an overview of developing applications using SysWorks™.

When SysWorks™ is installed at a system of turnkey level and is used to create an application environment, various OpenVMS authorisation and disk structures are created. If SysWorks™ is not used to create these structures, they have to be created manually or by other site specific means. See Chapter 11 for details about what structures need to be created and how this may be done.

1.1 Environments

Each application is developed and maintained in a set of environments.

When CMS is used, a common CMS library contains a class for each environment. See Section 3.3 for more details. The following sections describe each of the standard application development environments.

1.1.1 Common

The fixed code for the common environment is APPL.

1.1.2 Pre-Development

The default code for the pre or future development environment is FDEV. Other codes used include DDEV.

1.1.3 Development

The default code for the development environment is DEV. Other codes used include DEVL.

1.1.4 Development Testing

The default code for the development testing environment is DTST.

1.1.5 Maintenance

The default code for the maintenance environment is MNT. Other codes used include MAINT.

1.1.6 Maintenance Testing

The default code for the maintenance testing environment is MTST.

1.1.7 Production

The default code for the production environment is PROD. Other codes used include PRD.

1.1.8 Training

The default code for the training environment is TRNG.

1.2 Moving between Applications and Environments

After logging in to a cluster, the first command to use to access the application development sub-system is the CONTEXT APPLICATION command. In full SysWorks™ environments, the symbol that implements this command is defined at login time. In private SysWorks™ environments, the user needs to ensure that this symbol (and any others referred to in this guide) are defined. The most common way to define these symbols is by executing SWRK_SFT_DIR:SYLOGIN.COM. If in doubt, consult your site manager for the definition of the SWRK_SFT_DIR logical name. The CONTEXT APPLICATION command performs various actions required to ‘move’ the user to the application environment. See the command dictionary for full details on this command. One of the actions performed is changing the users default device and directory.

1.2.1 LOGICALS.COM

An application environment’s LOGICALS.COM command procedure is used to define application logical names. In a full SysWorks™ installation, this procedure is normally executed under the application environment’s username at system startup or disk mount time, and populates a system wide logical name table. It is also executed by the CONTEXT APPLICATION , CONTEXT ENVIRONMENT or CONTEXT VERSION commands if the application environment about to be entered does not have populated logical name table. If the logical name tables does not exist, these commands create a process logical name table - see section Chapter 2 for more details.

1.2.2 ENTER.COM

This command procedure is executed after other actions of the CONTEXT APPLICATION , CONTEXT ENVIRONMENT or CONTEXT VERSION commands have been completed. It is associated with the context in effect after the move takes place.

It is used to create DCL symbols and define process and job logical names. These logical name definitions are required because some products (eg. ACMS task group debugging) need logicals to be in the job table rather than the application environment table.

1.2.3 EXIT.COM

This command procedure is executed before other actions of the CONTEXT APPLICATION , CONTEXT ENVIRONMENT or CONTEXT VERSION commands. It is associated with the context in effect before the move takes place.

All actions taken in an applications ENTER.COM should be reversed in its EXIT.COM. It should delete DCL symbols created in, and deassign logicals defined in the corresponding ENTER.COM. There is no need to deassign logical

names in the application environment logical name table because it is 'hidden' when the application environment is not in use.

Logical Names and Logical Name Tables

Each application environment uses a logical name table. The name of this table has the form `LNM_appl_envr`.

If SysWorks™ is installed at the system level or above, this table would normally be a system wide table. In private installations, it would normally be a process logical name table. In either case, if there is no process or system logical name table when a `CONTEXT APPLICATION`, `CONTEXT ENVIRONMENT` or `CONTEXT VERSION` command is used, a process or system wide logical name table will be created depending on the developers privileges.

2.1 Common and Specific Contexts

Some application environments support the both a common and specific context. These environments are typically the development and maintenance style environments. Testing and production style environments normally support only the common context.

2.1.1 Common

In the common context, all users share the application environments common directory structure such as software, data, library and work directories. Compiling and building under the common context causes the new versions to be seen by all other users and developers who are also using the common context.

2.1.2 Specific

In the specific context, users have private versions of software and/or data on which they are working. This is achieved by making many of the standard application logicals search list in the users job logical name table. These search lists are designed to use the users private version first, and if that is not present, to use the common version.

Table 2–1 describes the objects for which search lists may be defined.

Table 2–1 Specific Context Search Lists

Object	Job Logical Name Usage
CDD/Repository default	Includes a <code>USER.user</code> directory before the root directory.
CMS Class Path	If the common CMS path does not specify the latest generation, includes a user specific path before the common path. See Section 3.3 for more details.

Table 2-1 (Cont.) Specific Context Search Lists

Object	Job Logical Name Usage
The following directories:	For each of these directories a user sub-directory of the associated common directory or, if that is not present, the users work sub-directory, is included.
<ul style="list-style-type: none">• Data• Library• Runtime• Software• Work	

This chapter describes the source control facilities provided by SysWorks. At this time SysWorks only uses CMS for source control.

Each application has at least one application common CMS library.

3.1 CMS Libraries

SysWorks supports a common CMS library for all environments of an application. This CMS library must be located in the APPL environment (eg. `DISK_APPL:[appl.SRC.CMSLIB]`). Each environment is represented by a CMS class within the library.

By default, the class names are of the form *appl_envr*. For example, the `XYZ_DEV` class would contain the development generations of the XYZ application's source.

The system also supports optional separate CMS libraries for development and requires them pre or future development environments (eg. `DISK_DEV:[appl.SRC.CMSLIB]`). These CMS libraries should still contain the appropriate classes.

One of the primary reasons for supporting these split CMS libraries is performance. Having multiple CMS libraries reduces locking, both at a CMS level and a disk level.

3.2 Application Common CMS Library

This CMS library resides in the common area of the application. For example the CMS library for the FIN application would reside in:

```
DISK_APPL: [FIN.SRC.CMSLIB]
```

No reference directory is used as the application environment work directories effectively become the copy of the appropriate generations.

3.3 Classes and Generations

In the common CMS library there is a class for each application environment and application release. These classes have names of the format *appl_envr* and *appl_vrsn*. For example `FIN_DEV` might represent the financial application development environment class, and `FIN_V010` might represent the production version V1.0 of the application.

Table 3–1 lists the various environments and their use of CMS. The following should be noted:

- The mainline or variant generations only apply in the common CMS library.

- Future development environments do not have an associated class in the common CMS library.
- When a development or maintenance environment has its own local CMS library:
 - Mainline generations are used in the local CMS library;
 - Classes are still used in the common CMS library and should still use the mainline or variant conventions;
 - A class may be used in the local CMS library;

Table 3–1 Environments, Generations and Classes

Environment Type	Default Environment	CMS Library	Mainline or Variant	Notes
COMMON	APPL	Yes		
DEVELOPMENT	FDEV	Yes	Mainline	Standalone CMS library with no connections to the common CMS library.
	DEV	Optional	Mainline	Newer than previous release from which development is occurring. Generally the latest generation.
DEV_TESTING	DTST	No	Mainline	Similar to above, except not necessarily the latest generation.
MAINTENANCE	MNT	Optional	Variant	Variant from the release from which maintenance is occurring. Generally the latest generation in the variant branch.
MNT_TESTING	MTST	No	Variant	Similar to above, except not necessarily the latest variant generation.

Access to an application environment always grants access to the class associated with that environment. If SysWorks Administrator is used, access to other classes is read only.

There are two logical names which are used to assist in managing the CMS libraries classes. These are:

- *appl_CMS_PATH*
- *appl_CMS_VARIANT*

Normally these logicals are defined by the application's LOGICALS.COM in the application logical name table.

By supplying combinations of these logical names different effects are produced. Table 3–2 summarises the effects of the combination of these logical names.

Table 3–2 Use of *appl_CMS_PATH* AND *appl_CMS_VARIANT*

<i>appl_CMS_PATH</i>	<i>appl_CMS_VARIANT</i>	Effect
Undefined	Undefined	No management of classes is performed. BUILDS always use the latest generation of all elements.

Table 3–2 (Cont.) Use of *appl_CMS_PATH* AND *appl_CMS_VARIANT*

<i>appl_CMS_PATH</i>	<i>appl_CMS_VARIANT</i>	Effect
Undefined	<i>letter</i>	Invalid
<i>appl_envr</i>	Undefined	<p>Elements are managed within the class. The following behaviour is implemented:</p> <ul style="list-style-type: none"> • BUILDS use the generations from the class. • CREATE ELEMENT inserts generation 1 of the element into the class. • RESERVE requires the current generation of the element in the class be the latest. • REPLACE inserts the new latest generation of the element into the class.
<i>appl_envr</i>	<i>letter</i>	<p>Elements are managed within the class. The following behaviour is implemented:</p> <ul style="list-style-type: none"> • BUILDS use the generations from the class. • CREATE ELEMENT creates two generations, 1 and 1A1 and inserts generation 1A1 into the class. • RESERVE requires the current generation of the element in the class to be the latest. • REPLACE inserts the new latest generation of the element into the class.
<i>appl_envr+</i>	Undefined	<p>Elements are managed within the class. The following behaviour is implemented:</p> <ul style="list-style-type: none"> • BUILDS use the latest generations from the class. • CREATE ELEMENT inserts generation 1 of the element into the class. • RESERVE reserves the latest generation of the element within the class.
<i>appl_envr+</i>	<i>letter</i>	<p>Elements are managed within the class. The following behaviour is implemented:</p> <ul style="list-style-type: none"> • BUILDS use the latest generations from the class. • CREATE ELEMENT creates two generations, 1 and 1A1 and inserts generation 1A1 into the class. • RESERVE reserves the latest generation of the element within the class.

During the various version control tasks, generations are duplicated in classes. For example, a TRIAL from development to development testing would cause all the generations in the development class to be inserted into the development testing class.

3.4 Common and Specific CMS Classes

If a specific context is used and the first item in the application *appl_CMS_PATH* logical name does not specify the latest generation of a class (i.e. it does not finish in a +), a job *appl_CMS_PATH* logical name is defined with a user specific class as the new first item. This user specific class is of the form *appl_envr_user*.

For example an environment which does support specific generations:

```
$ context application fin dev/common
  DISK_DEV5:[FIN.WRK.JONES_AB]
  Executing application FIN in DEV's ENTER.COM
$ show logical fin_cms_path
  "FIN_CMS_PATH" = "FIN_DEV" (LNM_FIN_DEV)
$ show logical fin_cms_variant
%SHOW-S-NOTRAN, no translation for logical name FIN_CMS_VARIANT
$ context specific
  DISK_DEV5:[FIN.WRK.JONES_AB]
  Executing application FIN in DEV's ENTER.COM
$ show logical fin_cms_path
  "FIN_CMS_PATH" = "FIN_DEV_JONES_AB+" (LNM$JOB_80E05930)
  = "FIN_DEV"
  "FIN_CMS_PATH" = "FIN_DEV" (LNM_FIN_DEV)
```

In the above example, a DEVTOOLS CMS PROMOTE or SRCCTL PROMOTE command would be used to promote the developers specific generation into the common class.

For example an environment which does not support specific generations:

```
$ context environment mnt/common
  DISK_MNT4:[FIN.WRK.JONES_AB]
  Executing application FIN in MNT's ENTER.COM
$ show logical fin_cms_path
  "FIN_CMS_PATH" = "FIN_MNT+" (LNM_FIN_MNT)
$ show logical fin_cms_variant
  "FIN_CMS_VARIANT" = "A" (LNM_FIN_MNT)
$ context specific
  DISK_MNT4:[FIN.WRK.JONES_AB]
  Executing application FIN in MNT's ENTER.COM
$ show logical fin_cms_path
  "FIN_CMS_PATH" = "FIN_MNT+" (LNM_FIN_MNT)
```

3.5 Multi-variant CMS Classes

If a multi-variant development environment is in use the previously described formats for the *appl_CMS_PATH* logical name items become *appl_envr_VARvrnt* and *appl_envr_VARvrnt_user* as appropriate.

For example:

```
$ context application fin dev b/common
  DISK_DEV5:[FIN.VARB.WRK.JONES_AB]
  Executing application FIN in DEV variant B's ENTER.COM
$ show logical fin_cms_path
  "FIN_CMS_PATH" = "FIN_DEV_VARB" (LNM_FIN_DEV_VARB)
$ show logical fin_cms_variant
  "FIN_CMS_VARIANT" = "B" (LNM_FIN_DEV_VARB)
$ context specific
  DISK_DEV5:[FIN.VARB.WRK.JONES_AB]
  Executing application FIN in DEV variant B's ENTER.COM
$ show logical fin_cms_path
  "FIN_CMS_PATH" = "FIN_DEV_VARB_JONES_AB+" (LNM$JOB_80E05930)
  = "FIN_DEV_VARB"
  "FIN_CMS_PATH" = "FIN_DEV_VARB" (LNM_FIN_DEV_VARB)
```

3.6 Multi-version CMS Classes

If a multi-version maintenance environment is in use the previously described formats for the *appl_CMS_PATH* logical name items become *appl_envr_vrsn* and *appl_envr_vrsn_user* as appropriate.

For example:

```
$ context application fin mnt v2.1/common
  DISK_MNT5:[FIN.V021.WRK.JONES_AB]
  Executing application FIN in MNT version V021's ENTER.COM
$ show logical fin_cms_path
  "FIN_CMS_PATH" = "FIN_MNT_V021" (LNM_FIN_MNT_V021)
$ show logical fin_cms_variant
  "FIN_CMS_VARIANT" = "A" (LNM_FIN_MNT_V021)
$ context specific
  DISK_MNT5:[FIN.V021.WRK.JONES_AB]
  Executing application FIN in MNT version V021's ENTER.COM
$ show logical fin_cms_path
  "FIN_CMS_PATH" = "FIN_MNT_V021_JONES_AB+" (LNM$JOB_80E05930)
  = "FIN_MNT_V021"
  "FIN_CMS_PATH" = "FIN_MNT_V021" (LNM_FIN_MNT_V021)
```

3.7 Multiple CMS Libraries

SysWorks also allows development style environments to have their own CMS libraries. This is primarily to allow higher performance during the highly volatile development stage, and to reduce the number of 'junk' versions in the primary CMS library. This arrangement is made on an application environment basis. For example the FIN application in development may have its own CMS library which would reside in:

```
DISK_DEV:[FIN.SRC.CMSLIB]
```

No reference directory is used for the same reason as with the common CMS library. If a separate CMS library is used for a development style environment, the *appl_envr* class and the *appl_CMS_PATH* logical name should still be defined.

During the various version control procedures the generations are copied across to the application common CMS library.

3.8 Commands

Three command interfaces are used to control sources. These are:

- CMS
- SRCCTL (minimum abbreviation SRC)
- DEVTOOLS CMS (minimum abbreviation DEV CMS)

The last two are used to act as a front end for CMS. See Table 3–3 for a list of CMS commands and the front ends which will accept them.

Table 3–3 CMS Command Availability

Command	CMS	DEVTOOLS	SRCCTL
ACCEPT GENERATION	X		

Table 3-3 (Cont.) CMS Command Availability

Command	CMS	DEVTOOLS	SRCCTL
ANNOTATE	X		
CANCEL REVIEW	X		
COPY ELEMENT	X		
CREATE CLASS	X		X ¹
CREATE GROUP	X		X ¹
CREATE ELEMENT	X	X	X
CREATE LIBRARY	X		1
DELETE CLASS	X		X ¹
DELETE GENERATION	X		X ¹
DELETE GROUP	X		X ¹
DELETE ELEMENT	X	X	X
DELETE HISTORY	X		X ¹
DIFFERENCES	X		
FETCH	X	X	X
INSERT ELEMENT	X		
INSERT GENERATION	X	X	
INSERT GROUP	X		
MARK GENERATION	X		
MODIFY CLASS	X		
MODIFY ELEMENT	X		
MODIFY GENERATION	X		
MODIFY GROUP	X		
MODIFY LIBRARY	X		
PROMOTE		X	X
REJECT GENERATION	X	X	
REMARK	X		
REMOVE ELEMENT	X		
REMOVE GENERATION	X	X	
REMOVE GROUP	X		
REPLACE	X	X	X
REPORT		X	X
RESERVE	X	X	X
RETRIEVE ARCHIVE	X		
REVIEW GENERATION	X		
SET ACL	X		
SET LIBRARY	X	X	
SHOW ACL	X		X ¹

¹The whole compound (i.e. multiple word) command must be entered at the Selection: prompt. If only the first word of the compound command is entered at the prompt, the second word is assumed to be ELEMENT. For example Selection: **CREATE ELEMENT**

Table 3-3 (Cont.) CMS Command Availability

Command	CMS	DEVTOOLS	SRCCTL
SHOW ARCHIVE	X		X ¹
SHOW CLASS	X		X ¹
SHOW ELEMENT	X		X
SHOW GENERATION	X		X ¹
SHOW GROUP	X		X ¹
SHOW HISTORY	X		X ¹
SHOW LIBRARY	X		X ¹
SHOW RESERVATIONS	X		X ¹
SHOW REVIEWS_PENDING	X		X ¹
SHOW VERSION	X		X ¹
UNRESERVE	X	X	X
VERIFY	X		

¹The whole compound (i.e. multiple word) command must be entered at the Selection: prompt. If only the first word of the compound command is entered at the prompt, the second word is assumed to be ELEMENT. For example Selection: **CREATE ELEMENT**

If the SysWorks Developer CMS front end (SRCCTL or DEVTOOLS CMS) is used, the *appl_CMS_PATH* and *appl_CMS_VARIANT* logical names provide defaults for use with the /PATH qualifier. The /PATH qualifier is not a CMS qualifier, rather it is an extension provided by SysWorks to element based CMS commands such as CREATE ELEMENT to indicate that the element is to be placed in the indicated class after being created in the CMS library.

By default, a command which supports the /PATH qualifier will, if the qualifier is not specified, default to the equivalence of the *appl_CMS_PATH* logical name.

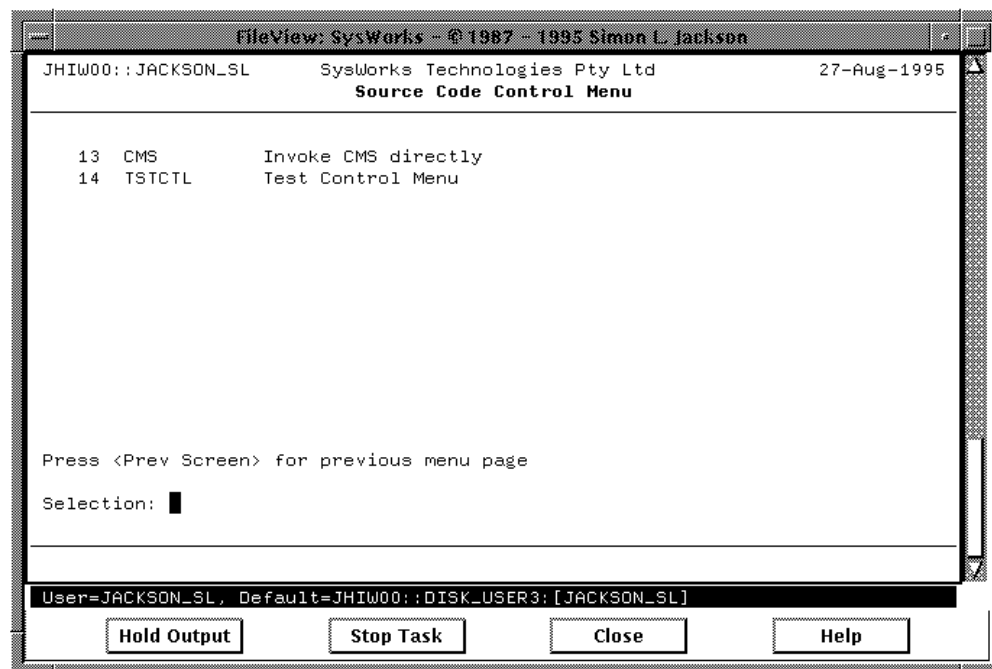
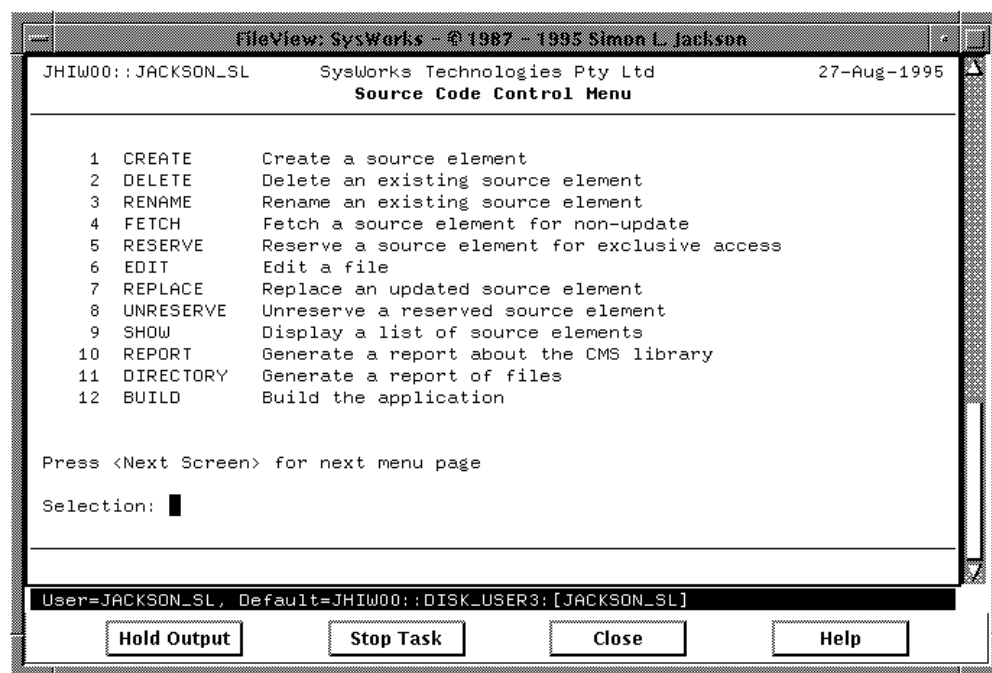
By default, a command which supports the /VARIANT qualifier will, if the qualifier is not specified, default to the equivalence of the *appl_CMS_VARIANT*.

The SRCCTL method supplies a menu based front end which ultimately uses the DEVTOOLS CMS command to perform the actual operation. If the SRCCTL command is entered from DCL without any sub-command following or is selected from the session manager **Manage** ⇒ **Sources** pulldown menu, it displays the menu illustrated in Figure 3-1 and prompts for a sub-command. If the SRCCTL command is followed by one of the sub-commands indicated in Figure 3-1, the menu is not displayed and the sub-command dialog is entered directly.

The DEVTOOLS CMS method is a foreign utility with a full command line interface (i.e. entering DEVTOOLS without any sub-command causes a DEVTOOLS> prompt to appear). At this time not all SRCCTL commands are fully supported by DEVTOOLS CMS - some are still implemented by using CMS directly - so although DEVTOOLS CMS is faster, SRCCTL is more complete. In a future version of SysWorks Developer, the DEVTOOLS CMS command will support all CMS commands and qualifiers.

Both of these front ends extend CMS by allowing lists of elements and an indirect construction. For example, the following command:

Figure 3-1 SRCCTL menu



`$ devtools cms reserve fin_main_prog.c,fin_other.c "Fix"`
will reserve both elements. Also, the following command:

```
$ devtools cms reserve @t1.txt "Fix them all"
```

will reserve each of the files specified in T1.TXT. Another enhancement allows full file specifications to be given in place of CMS element names. This becomes useful in the following example:

```
$ search [-]*.com* old_sym/window=0/out=t1.txt
$ devtools cms reserve @t1 "Change OLD_SYM to NEW_SYM"
$ change *.com* old_sym new_sym/verify
```

In this example the search command with /WINDOW=0 produces only the file specifications of the files containing the requested text. The output of this can then be fed into the DEVTOOLS CMS RESERVE command, which will reserve each of the elements which contain the text. The CHANGE command will then change the text to a new value and display each of the lines thus changed in each file. Note the use of the [-] construction in the SEARCH command. Given a developers default directory will be of the form DISK_envr:[*appl.WRK.user*], this construct results in searching the common work directory, which would normally contain copies of the appropriate generations (i.e. current or latest from class) of all the elements associated with the application environment's CMS class.

The following sub-sections describe the extra actions that the front end command will perform above the base CMS operation, and extra tasks provided by the SRCCTL front end. They are presented in alphabetic order.

3.8.1 BUILD

One of the items on both the SRCCTL and VSNCTL menus is BUILD. When selected, it asks a set of questions before building the application's software. See Section 5.2 for a description of how BUILD works. It is also available from DCL as a command. When used directly from DCL, the BUILD command takes qualifiers much like any normal DCL command. See the Command Dictionary or use HELP BUILD for more details on using the BUILD command from DCL.Intermediate Directories

3.8.2 CREATE

3.8.3 DELETE

3.8.4 DIRECTORY

3.8.5 EDIT

3.8.6 FETCH

3.8.7 RENAME

3.8.8 REPLACE

3.8.9 REPORT

Examples

```
$ srcctl report
Version [V2.0]: V1.0
Development environment [DEV]:
Development testing environment [DTST]:
Maintenance environment [MNT]:
Maintenance testing environment [MTST]:
Output [SYS$OUTPUT]:
Display (All/New/Unused) [All]: NEW,UNUSED
Execution (Batch/Detached/Online/Subprocess) [Batch]: ONLINE
```

This example generates a report listing elements which are not in the release class or are not in any class. The report is generated online.

3.8.10 RESERVE

3.8.11 SHOW

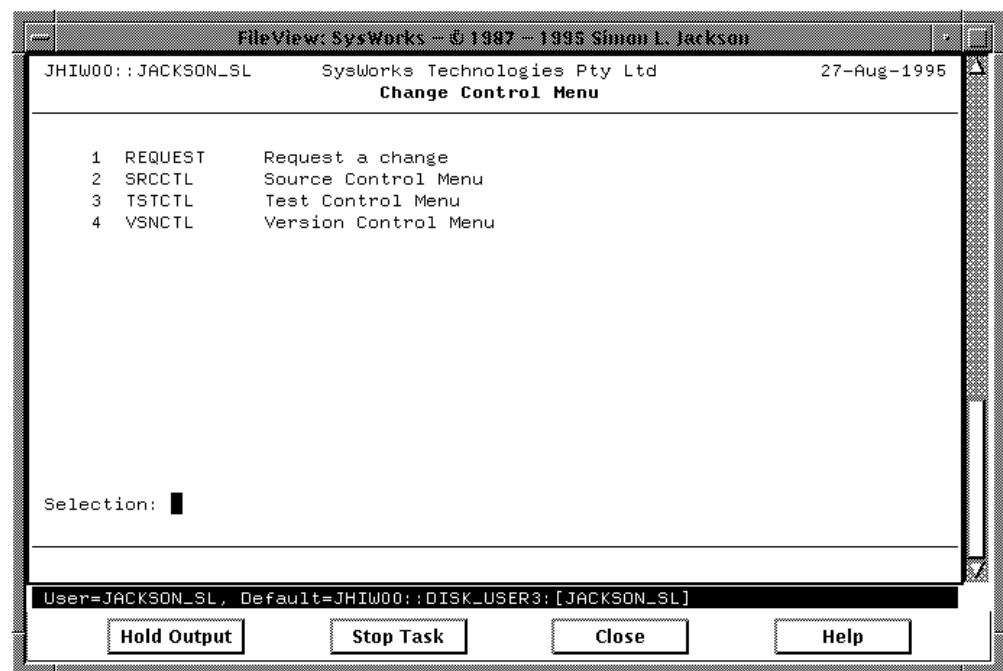
3.8.12 UNRESERVE

Change Control

This chapter discusses the change control facilities provided by SysWorks.

Figure 4-1 illustrates the version control menu presented by using the CHGCTL command without any parameters or the session manager **Manage** ⇒ **Versions** pulldown menu.

Figure 4-1 CHGCTL menu



Compiling and Building

This chapter describes compiling sources and building targets.

5.1 Compiling Sources

The `COMPILE` command is used to compile or translate a source into its next form, which for most sources is an intermediate form such as an object file. For example, compiling a Cobol source would produce an object file. Some sources can be directly translated into a target form. For example, compiling a Linker options file would produce an image file. The one parameter to the `COMPILE` command is a list of source files. Generally the `COMPILE` command does not take into account dependences. For example, no attempt is made to check that a dictionary record used in a Cobol source is present or up-to-date in the dictionary. See the Command Dictionary for more details.

If a developer has just edited a single source, the `COMPILE` command is a much simpler way of building the target than using the `BUILD` command. For example, the command:

```
$ compile fixed_module.c,[-]main_prog.opt
```

might be useful if the developer has just fixed `fixed_module.c` which needs to be linked into `main_prog.exe`. This is much faster than the equivalent `BUILD` command which would be:

```
$ build fin_sft_dir:main_prog.exe
```

5.2 Building Targets

The `BUILD` command is used to build targets from their constituent sources. The one parameter to the `BUILD` command is a list of target files. The `BUILD` command uses MMS to ensure that all dependencies between sources are handled in a logical order. For example, building an image file will ensure the dictionary records are up-to-date, compile a Cobol source, and finally link it into an image.

In order to remove most the the need to maintain MMS description files, the `BUILD` command is implemented as a multi-phase tool. The standard phases are listed along with their qualifiers in Table 5-1.

Table 5-1 Build Phases

Phase	BUILD Qualifier
PRECLEAN	/FROM[=location]
UPDATE	/UPDATE
TIDY	/TIDY
FETCH	/FETCH
SETUP	/PHASE=SETUP
SCAN	/PHASE=SCAN
RULES	/PHASE=RULES
DESCRIP	/PHASE=DESCRIP
RELATED	/RELATED
KITBLD	/KITBLD

A BUILD operation consists of the execution of one or more of these phases. Each phase produces outputs which are used as input to the next phase. For example, the RULES phase might generate an MMS script of all source dependencies which is then used in the subsequent DESCRIP phase. An application may use any number of phases in a BUILD.

These phases may have any name, however, the DESCRIP phase has some special defaults applied to it. The UPDATE, FETCH and PRECLEAN phases are controlled by qualifiers on the BUILD command. All other phases are driven by a DCL command procedure and/or an MMS script. These have the same file name as the BUILD phase, and have a file type of .COM or .MMS as appropriate. They should be maintained in the CMS library as per any other source file. If both are found, the DCL command procedure is executed first.

Typically the SETUP, RULES and DESCRIP phases are MMS scripts, and the SCAN and KIT phases are DCL command procedures. The BUILD operation will fetch DCL command procedures associated with phases from the CMS in much the same fashion as MMS does.

When BUILD invokes MMS to perform actions during a phase, the target requested has the same name as the phase. The exception to this is the DESCRIP phase, in which MMS is passed the target from the original BUILD command.

5.2.1 Phases

The following sub-sections describe each phase in greater detail. They are presented in the order in which they would normally be executed.

5.2.1.1 PRECLEAN

Clean out appropriate intermediate and target areas. This is achieved by deleting all files from the work, library and/or software directory trees (including the CDD repository), thus forcing the application to be built from sources in the CMS library.

Note that the three files ENTER.COM, EXIT.COM and LOGICALS.COM are preserved in the software directory so that commands such as CONTEXT will continue to work.

This phase is not generally explicitly included in a build, rather the `/FROM=location` qualifier forces this phase to be executed before the SETUP phase.

The set of directories which will be emptied during the PRECLEAN phase is controlled by the location code specified with the `/FROM` qualifier. See the `/FROM` qualifier for details about the location codes, the directories which are emptied and the effect this has on building the application.

5.2.1.2 UPDATE

Update the application work directory with the latest sources from the developers work directory. This phase is not used with the SPECIFIC scope.

5.2.1.3 TIDY

Deletes files in the developers work directory which exist in the common work directory and are not reserved by the developer. Also deletes the associated files in the developers library directory. Also deletes all `.TAG_CDD` files in the developers library directory and all the objects in the developers CDD/Repository area. In a future release of SysWorks, only `.TAG_CDD` files and CDD/Repository objects that need to be deleted will be deleted.

5.2.1.4 FETCH

Fetches all newer sources from the CMS library. This phase is an alternative to using CMS during the RULES and DESCRIP phases.

5.2.1.5 SETUP

The SETUP phase performs actions required before other ALL type phases. It is typically used to generate build time structures and software. This allows build time executables (typically DCL command procedures and other scripts needed during the RULES sub-phase) to be built prior to the main build sub-phases.

5.2.1.6 SCAN

The SCAN phase is typically used to generate dependency lists based on current sources - i.e. convert a list of sources (usually taken from the CMS library) into an MMS script which will generate the source based MMS scripts in the RULES sub-phase. To standardise this phase, the DCL command procedure `SWDEV_SFT_DIR:SWDEV_BUILD_SCAN` may be executed to generate a set of standard MMS scripts. These MMS scripts are generated with a file type of `.MMS`, and are created in the library directory. Where targets in them are generated MMS scripts, the file type will be `.MMS_INC`. This is so that at the end of the RULES phase (when all these `.MMS_INC` scripts should be up to date), they can be copied into a single `appl_APPENDS.MMS` script for inclusion in the DESCRIP phase. This reduces the overhead of having to `.INCLUDE` a large number of individual source based MMS scripts.

The SCAN phase generated files are described in Table 5-2.

Table 5–2 SCAN Phase Generated Files

Generated File	Usage
<i>appl_CONSTRAINTS</i> .SQL	An SQL script containing a list of all the sources with a file name and type of the form <i>appl_table_CKn</i> .SQL, <i>appl_table_FKn</i> .SQL and <i>appl_table_PK</i> .SQL. The @ character precedes each source file specification so that this procedure can be used from within the SQL utility.
<i>appl_DOCUMENTATION</i> .MMS	A dependency list of all the documentation targets.
<i>appl_DOMAINS</i> .SQL	An SQL script containing a list of all the sources with a file name and type of the form <i>appl_domain_DOM</i> .SQL. The @ character precedes each source file specification so that this procedure can be used from within the SQL utility.
<i>appl_DOMAIN_SDML_LIST</i> .MMS	A dependency list of all the generated SDML domain documentation sources.
<i>appl_FORM_LIST</i> .MMS	A dependency list of all the generated DECforms form MMS scripts, so that a generated linker options file may be regenerated when any DECforms form is modified. This regeneration is typically performed near the end of the RULES phase.
<i>appl_HELP</i> .MMS	A dependency list of all the help targets.
<i>appl_INDICES</i> .SQL	An SQL script containing a list of all the sources with a file name and type of the form <i>appl_index_IDX</i> .SQL or <i>appl_index_PIDX</i> .SQL or <i>appl_index_SIDXn</i> .SQL. The @ character precedes each source file specification so that this procedure can be used from within the SQL utility.
<i>appl_LINK_OPT_LIST</i> .MMS	A dependency list of all the Linker options sources.
<i>appl_MSGHLP</i> .MMS	A dependency list of all the message help targets.
<i>appl_PROTECTIONS</i> .SQL	An SQL source which will apply a standard protection to each appropriate object in the database.
<i>appl_ROUTINES</i> .SQL	An SQL script containing a list of all the sources with a file name and type of the form <i>appl_routine_RTN</i> .SQL. The @ character precedes each source file specification so that this procedure can be used from within the SQL utility.
<i>appl_SCHEMA</i> .MMS	A dependency list of all the Rdb database schema sources. This script is usually included in the application DESCRIP.MMS script.
<i>appl_STORAGE_MAPS</i> .SQL	An SQL script containing a list of all the sources with a file name and type of the form <i>appl_storage_map_SM</i> .SQL. The @ character precedes each source file specification so that this procedure can be used from within the SQL utility.
<i>appl_TABLES</i> .SQL	An SQL script containing a list of all the sources with a file name and type of the form <i>appl_table_TBL</i> .SQL. Note that SQL constraints definitions are embedded within the table definition sources.
<i>appl_TABLE_SDML_LIST</i> .MMS	A dependency list of all the generated SDML table documentation sources.
<i>appl_TARGETS</i> .MMS	A dependency list of all the targets.
<i>appl_TASK_LIST</i> .MMS	A dependency list of all the generated ACMS task MMS scripts, so that a generated ACMS task group may be regenerated when any ACMS task is modified. This regeneration is typically performed near the end of the RULES phase.
<i>appl_TRIGGERS</i> .SQL	An SQL script containing a list of all the sources with a file name and type of the form <i>appl_trigger_TRG</i> .SQL or <i>appl_trigger_TRGR</i> .SQL. The @ character precedes each source file specification so that this procedure can be used from within the SQL utility.

Table 5–2 (Cont.) SCAN Phase Generated Files

Generated File	Usage
<i>appl_VIEWS</i> .SQL	An SQL script containing a list of all the sources with a file name and type of the form <i>appl_view_VIEW</i> .SQL or <i>appl_view_VW</i> .SQL. The @ character precedes each source file specification so that this procedure can be used from within the SQL utility.
<i>appl_VIEW_SDML_LIST</i> .MMS	A dependency list of all the generated SDML view documentation sources.

5.2.1.7 RULES

The RULES phase is typically used to generate dependencies based on current sources - i.e. convert sources to MMS scripts. This can be achieved by including the *appl_SCRIPT_LIST* and *appl_RULES* scripts from the SCAN phase. The actions normally generated in the *appl_RULES* script are a DEVTOOLS CONVERT/GENERATE command to convert a source to an MMS include script (file type .MMS_INC).

At the end of the RULES phase, all the generated MMS include script files are copied into one large *appl_APPENDS*.MMS. This is because it is faster to copy all these files together into one script for MMS than have MMS include each of the scripts. This copy is achieved using the DEVTOOLS COPY/NODUPLICATES command, so that the application library directory logical name may be a search list and only the earliest copy of each MMS include script file is copied into the *appl_APPENDS*.MMS script file.

The RULES phase generated files (other than MMS scripts generated from sources) are described in Table 5–3.

Table 5–3 RULES Phase Generated Files

Generated File	Usage
<i>appl_DEPENDENCIES</i> .MMS	An MMS script containing all the generated dependencies used to build the application.

5.2.1.8 DESCRIP

The DESCRIP phase is used to create targets based on dependencies - i.e. the traditional MMS build. This is achieved by including the generated MMS scripts from the RULES sub-phase into the DESCRIP.MMS script.

5.2.1.9 KITBLD

The KITBLD phase is typically used to create installation kit which is a set of OpenVMS BACKUP save-sets ready for OpenVMS INSTAL to install on target clusters. This phase is only executed when the BUILD/KITBLD qualifier is used.

5.2.2 Add Extra Phases

Some applications may need extra site or application specific phases. This is typically the case when a database is designed externally to the application and/or the application has generated sources.

This chapter discusses the version control facilities provided by SysWorks.

Figure 6-1 illustrates the version control menu presented by using the VSNCTL command without any parameters or the session manager **Manage** ⇒ **Versions** pulldown menu.

Figure 6-2 illustrates the various symbols used in the VSNCTL flow diagrams.

6.1 BUILD

This option invokes a set of questions which lead to a BUILD command being executed. See section 4.6.2 for more details on the BUILD command.

Example:

```
$ vsnctl build
Use application or user username (Application/User) [Application]:
From sources or work area (Source/Work) [Work]:
Phases [All]:
Use CMS library (Yes/No) [Yes]:
Prefetch CMS (Yes/No) [Yes]:
Debug (Yes/No) [Yes]:
Target(s) [All]:
Execution (Batch/Detached/Online/Subprocess) [Batch]:
Batch queue [SYS$BATCH]:
Execution after [NOW]:
```

Note that this build option does not support all of the BUILD commands qualifiers.

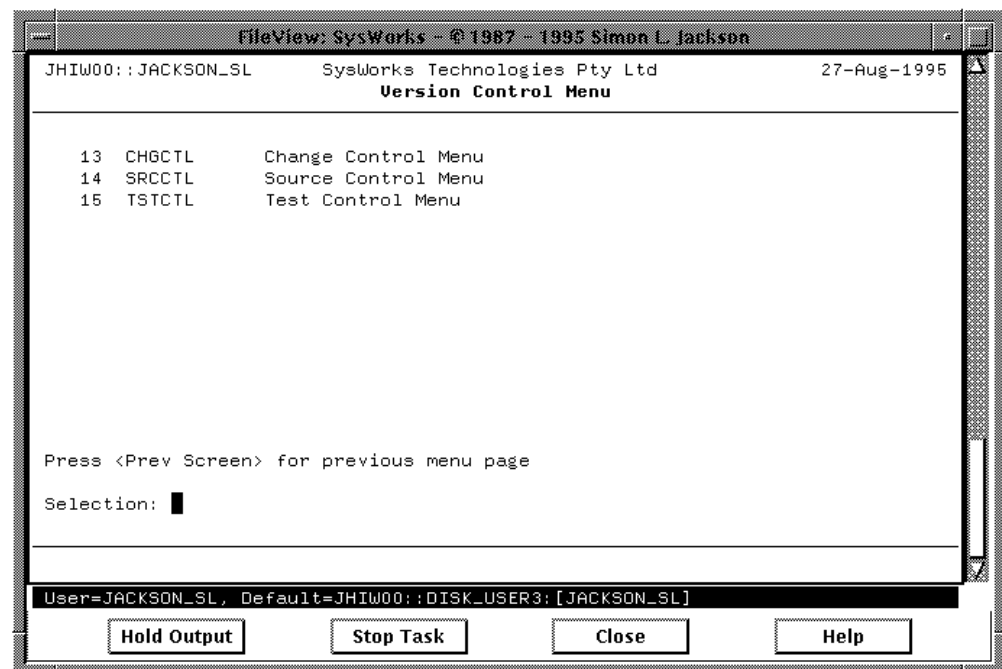
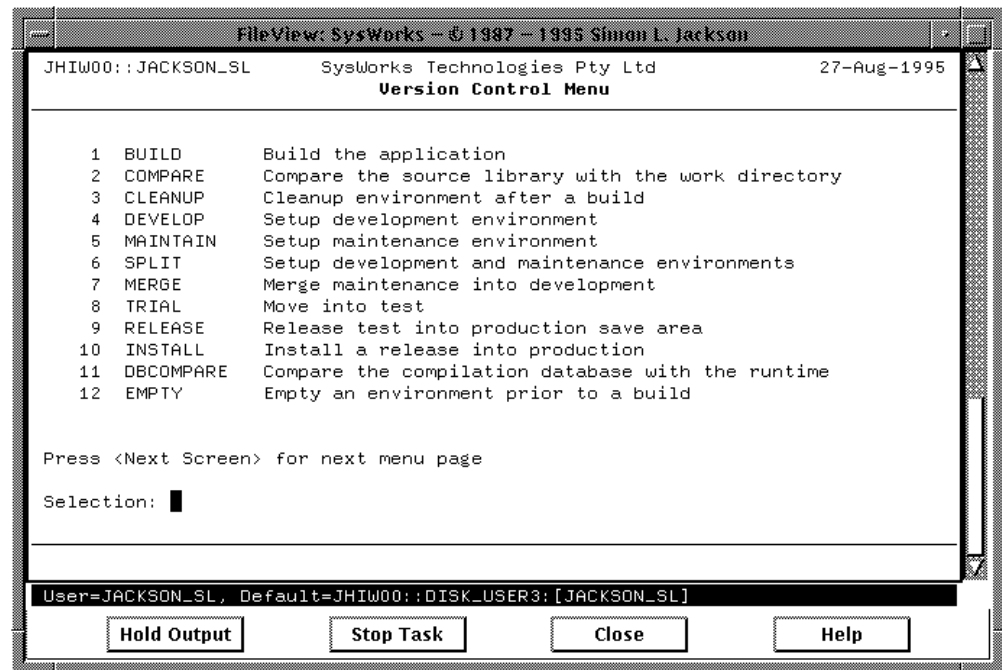
6.2 COMPARE

This option is used as a front end to the DEVTOOLS DIFFERENCES/DATES command (see also known as the COMPARE command)

Example:

```
$ vsnctl compare
```

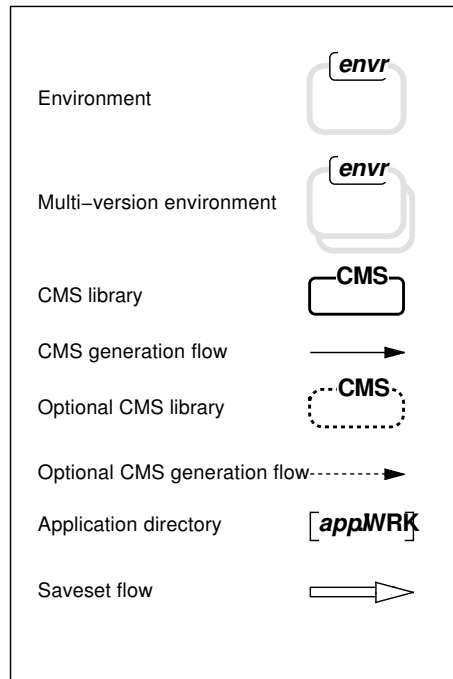
Figure 6-1 VSNCTL menu



6.3 CLEANUP

Example:

Figure 6–2 VSNCTL Diagrams Legend



```
$ vsnctl cleanup
```

6.4 DEVELOP

This option is used to take a released version into a development environment.

Example:

```
$ vsnctl develop
Version: V1.0
Development environment [DEV]:
Build (Yes/No) [Yes]:
Development testing environment [DTST]:
Execution (Batch/Detached/Online/Subprocess) [Batch]:
Batch queue [SYS$BATCH]:
Execute after [NOW]:
```

See the description of the SPLIT task for complete details of these questions and the actions associated with DEVELOP task.

6.5 MAINTAIN

This option is used to take a released version into a maintenance environment.

Example:

```

$ vsnctl maintain
Version: V1.0
Maintenance environment [MNT]:
Build (Yes/No) [Yes]:
Maintenance testing environment [MTST]:
Keep existing maintenance version (Yes/No) [Yes]:
Execution (Batch/Detached/Online/Subprocess) [Batch]:
Batch queue [SYS$BATCH]:
Execute after [NOW]:)

```

See the description of the SPLIT task for complete details of these questions and the actions associated with MAINTAIN task.

6.6 SPLIT

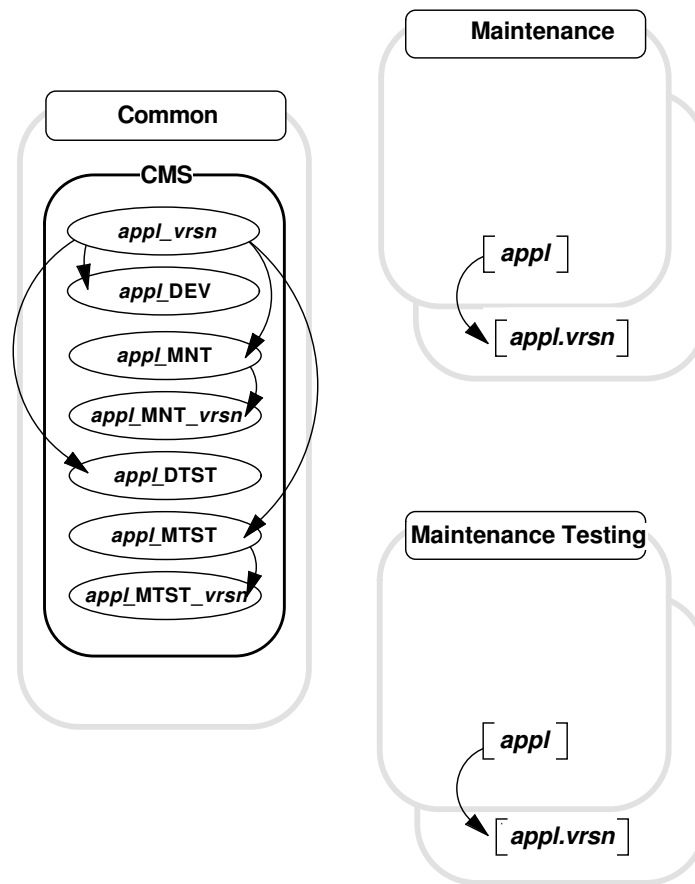
This option is used to split a released version into a development and maintenance stream. It must be used in an environment which uses the common CMS library. The release version must contain the latest generations in the CMS library. This will be the case if a RELEASE task has been executed without any subsequent CMS REPLACE or TRIAL tasks being used. This restriction will be removed in a future release of SysWorks.

Figure 6-3 illustrates the flow of sources during a SPLIT.

The following steps are taken:

- 1 If the existing maintenance version is being kept, the existing maintenance and maintenance testing CMS classes are renamed. These renames will be from *appl_envr* to *appl_envr_vrsn* where *envr* are the maintenance and maintenance testing environment codes, and *vrsn* is the existing maintenance environment version (i.e. the released version from which the previous MAINTAIN or SPLIT was performed)
- 2 Any of the four target CMS classes which do not exist are created. These classes have names of the form *appl_envr* where *envr* are the development, development testing, maintenance and maintenance testing environment codes. Note that if step 1 was taken, the new maintenance and maintenance testing classes will always be created since they no longer exist.
- 3 Each generation from the released version class is reserved and replaced twice. The first time in the main line, and the second with a variant. Each of the new main line generations is placed in the development and development testing CMS classes. Each of the new variant generations is placed in the maintenance and maintenance testing CMS class. These actions are designed to force development generations and maintenance generations to not be in the same line of descent so that future merge operations will be able to work properly. Note that in this release of SysWorks, no check is made when creating the new mainline and variant generations as to whether such a generation already exists. This creates the restriction that no work (i.e reserve and replace) can be done between the start and finish of a trial replese split sequence.
- 4 If any of the four environments have their own CMS library, this library is populated with the new generations.
- 5 If requested, a BUILD/FROM_SOURCES is executed in the development environment.

Figure 6–3 SPLIT flow



- 6 If the existing maintenance version is being kept, the existing maintenance directory structure is moved down one level into the `DISK_envr:[appl.vrsn]` directory where `envr` is the maintenance environment code and `vrsn` is the previous version code. If a CDD repository is included, a `CDO MOVE REPOSITORY` command is used and the user needs to have the appropriate access to execute this command.
- 7 If requested, a `BUILD/FROM_SOURCES` is executed in the maintenance environment.
- 8 Actions similar to step 6 are taken for the maintenance testing environment.

Example:

```
$ vsnctl split
Version: V1.0
Development environment [DEV]:
Build (Yes/No) [Yes]:
Development testing environment [DTST]:
Maintenance environment [MNT]:
Build (Yes/No) [Yes]:
Maintenance testing environment [MTST]:
Keep existing maintenance version (Yes/No) [Yes]:
Execution (Batch/Detached/Online/Subprocess) [Batch]:
Batch queue [SYS$BATCH]:
Execute after [NOW]:
```

The version of the application specified by the version question is represented in the CMS library by a class with a name of the form *appl_vrsn* where *appl* is the application code and *vrsn* is an OpenVMS kit style version number. For example, V1.0 becomes V010.

The environments requested in the four environment questions are represented in the CMS library by classes of the form *appl_envr* where *appl* is the application code and *envr* is the environment code. The defaults for these four questions are derived by translating the logical names *appl_DFLT_ENVR_envr* for each *envr* from the set of standard environment codes DEV, DTST, MNT and MTST. These logical names would normally be defined in the application logical name table by the application LOGICALS.COM command procedure. If an *appl_DFLT_ENVR_envr* logical doesn't translate, a translation of the logical name of the form *SWRK_DFLT_ENVR_envr* is attempted.

The build questions apply to the development or maintenance environment preceding them. There are no builds performed in the development testing or maintenance testing environments. The builds performed are the equivalent of `build/from_sources`

6.7 MERGE

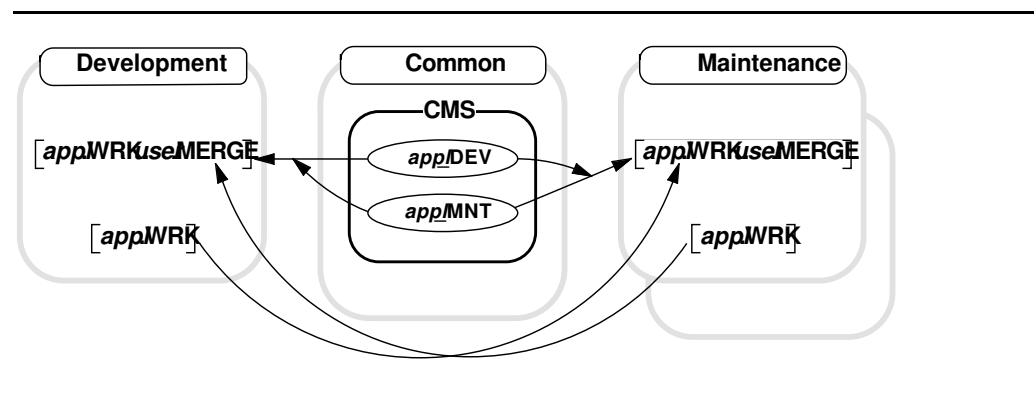
This option is used to merge in changes from a maintenance or future development environment into a development environment. It acts as a front end to the DEVTOOLS DIFFERENCES/DATES/MERGE command.

Example:

```
$ vsnctl merge
Target environment [MNT]:
Override CMS reservation check (Yes/No) [No]:
Since [BEGINNING]:
Execution (Batch/Detached/Online/Subprocess) [Batch]:
Batch queue [SYS$BATCH]:
Execute after [NOW]:
```

Figure 6-4 illustrates the flow of sources during a MERGE.

Figure 6-4 MERGE flow



6.8 TRIAL

This option is used to migrate source from a development environment to a development testing environment or a maintenance environment to a maintenance testing environment. It invokes a set of questions which lead to CMS generations being migrated between libraries and/or classes and on to perform an optional BUILD in the target testing environment.

Example:

```
$ vsnctl trial
Target environment [DTST]:
Override CMS reservations (Yes/No) [No]:
Build (Yes/No) [Yes]:
From sources or work area (Source/Work) [Work]:
Execution (Batch/Detached/Online/Subprocess) [Batch]:
Batch queue [SYS$BATCH]:
Execute after [NOW]:
```

The target environment default is determined by translating *appl_DFLT_ENVR_envr* or *SWRK_DFLT_ENVR_envr* where *envr* is the current environment code.

Figure 6–5 illustrates the flow of sources during a TRIAL.

Figure 6–5 TRIAL flow

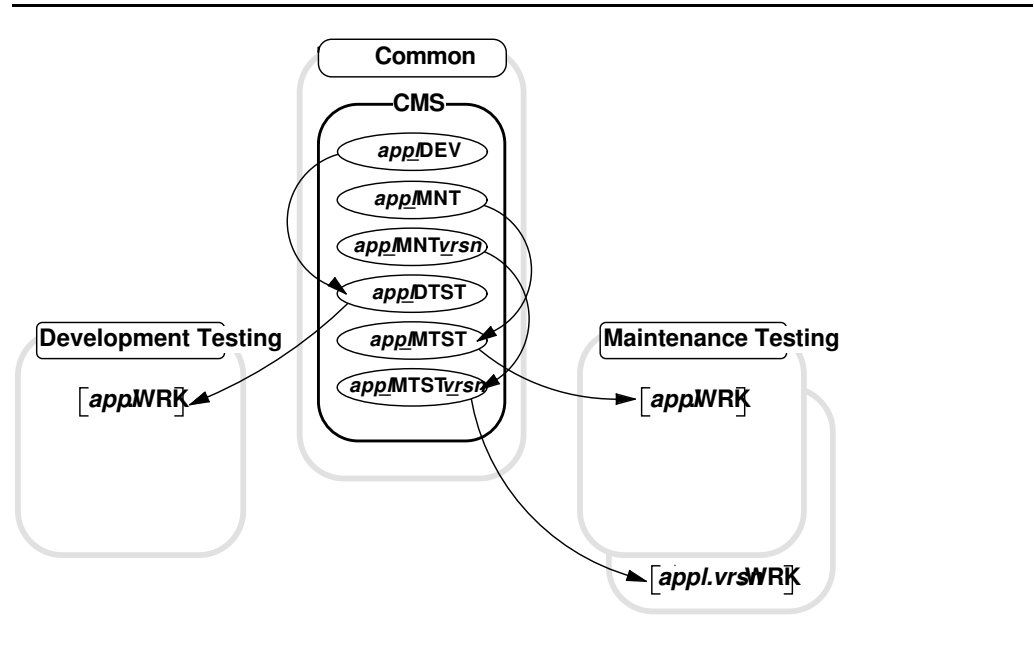
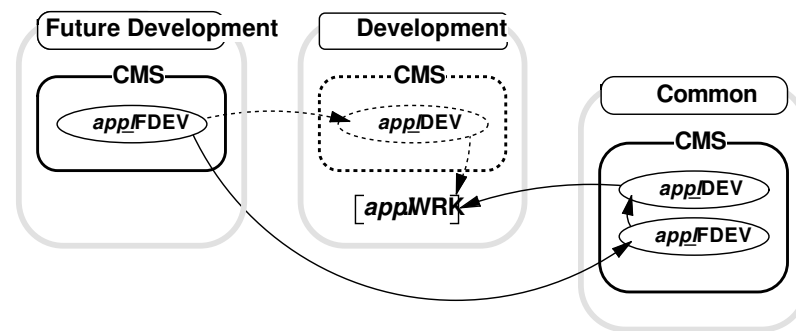


Figure 6–6 illustrates the flow of sources during a TRIAL from a future development to development environment.

Figure 6–6 TRIAL flow from FDEV to DEV



6.9 RELEASE

This option is used to release a tested version into the production save area, and prepare it for future development and/or maintenance.

The following steps are taken:

- 1 The generations in the specified development testing or maintenance testing environment are migrated into a version class.
- 2 A version kit sub-directory is created in the application common environment. This directory is of the form `DISK_APPL:[appl.KIT.vrsn]`.
- 3 The save sets in the specified development testing or maintenance testing environment kit sub-directory are copied into the created application common environment version kit directory.
- 4 The save sets are deleted from the development testing or maintenance testing kit sub-directory.

Example:

```
$ vsnctl release
Test environment [DTST]:
Version [V1.0]:
Execution (Batch/Detached/Online/Subprocess) [Batch]:
Batch queue [SYS$BATCH]:
Execute after [NOW]:
```

Figure 6–7 illustrates the flow of sources during a RELEASE.

6.10 INSTALL

This option is used to install a released version into production or training.

Example:

```
$ vsnctl install
Target environment [PROD]:
Execution (Batch/Detached/Online/Subprocess) [Batch]:
Batch queue [SYS$BATCH]:
Execute after [NOW]:
```

Figure 6–8 illustrates the flow of savesets during an INSTALL.

Figure 6–7 RELEASE flow

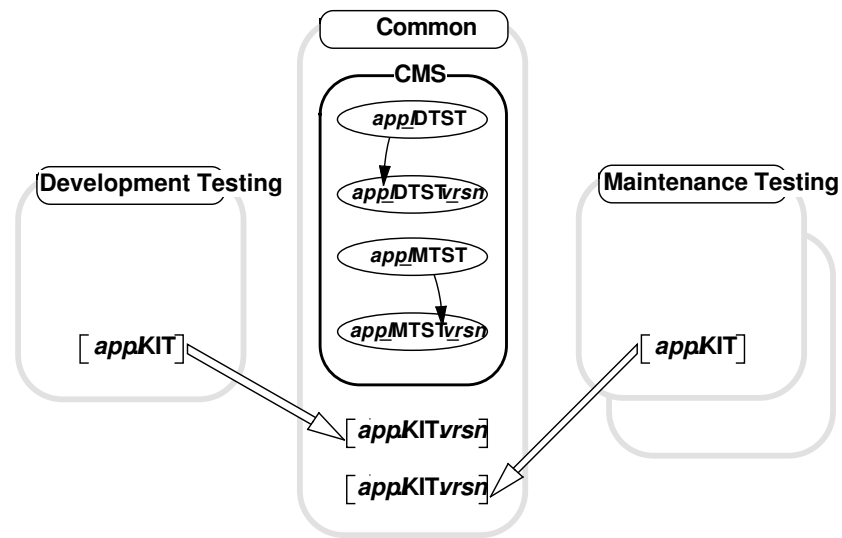
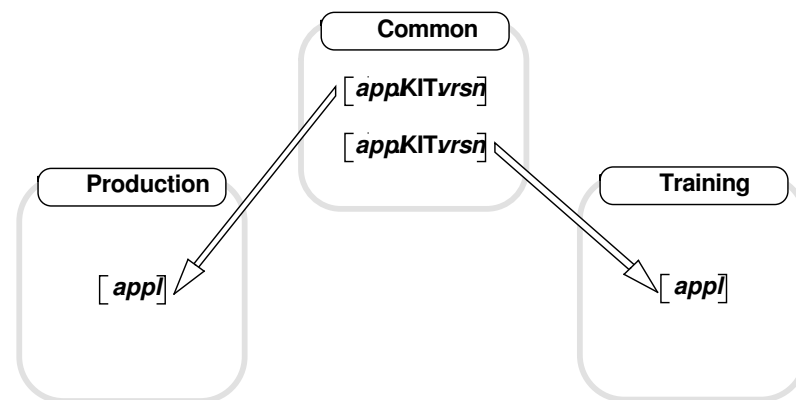


Figure 6–8 INSTALL flow



6.11 DBCOMPARE

Acts as a front end for the DEVTOOLS DIFFERENCES/RDB_DATABASE command. By default, it assumes that the first (or newer or compiletime) database exists in the software directory, and the second (or older or runtime) in the data directory.

Example:

```
$ vsnctl dbcompare
```


This chapter describes the creation of images within an application.

7.1 Creating Images

SysWorks Developer creates images based on Linker options files. These option sources must be present in order for SysWorks to create images. SysWorks does not attempt to determine which sources are image root modules by analysing the sources, even though some languages such as C have a specific syntax (eg. `main ()`) which suggest that a source forms an image root module.

SysWorks can create both executable and shareable images. Syntax within the linker options file indicates whether the resultant image should be executable or shareable.

The linker options files are parsed like most other sources in order to determine if any dependencies required. A linker options file must contain at least one module inclusion. This module can exist in the application library directory or an object library module. For example the following options file

```
!++
!
!  Linker options:
!      ACME_PROG_1
!
!  Purpose:
!      Linker options for the ACME_PROG_1 executable image.
!
!  History:
!      01-Apr-1995 by Simon L. Jackson
!          Initial version
!
!--
ACME_LIB_DIR:ACME_PROG_1
```

would generate the same image as

```
!++
!
!  Linker options:
!      ACME_PROG_1
!
!  Purpose:
!      Linker options for the ACME_PROG_1 executable image.
!
!  History:
!      01-Apr-1995 by Simon L. Jackson
!          Initial version
!
!--
```

```
ACME_OBJECT_LIB/INCLUDE=ACME_PROG_1
```

Where SysWorks is being used to maintain an existing application which does not yet have options files for its images, a procedure is provided to generate an initial set of options files. This procedure can be invoked with the following command:

```
@swdev_sft_dir:swdev_cvt_gen_exe_to_opt
```

7.2 Executable Images

7.3 Shareable Images

Shareable images should have transfer vectors in order to remove the need to relink executable images which use a shareable image when the shareable image is rebuilt.

The presence of a `GSMATCH` statement in the linker options file indicates to SysWorks that a shareable image is to be created rather than an executable image.

With OpenVMS VAX these transfer vectors are normally maintained in a Macro-32 source which uses the `.TRANSFER` directive or the SysWorks `VECTOR` macro. With OpenVMS Alpha, the transfer vectors are built using a syntax in the linker options file.

Example 7-1 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.

Example 7-1 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
```

Example 7-1 Cont'd on next page

Example 7-1 (Cont.) SWRKSHR.MAR

```
; 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_initialize
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_tree

; 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
```

Example 7-1 Cont'd on next page

Example 7-1 (Cont.) SWRKSHR.MAR

```
; 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
```

Example 7-1 Cont'd on next page

Example 7-1 (Cont.) SWRKSHR.MAR

```
; Version V3.2 - sorted on 22-Aug-1998
vector swrkdcl_show_context          seq=083
vector old                           seq=084 ; swrk_attach_swdat
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_swdat
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_swdat
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_swdata
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
```

Example 7-1 Cont'd on next page

Example 7-1 (Cont.) SWRKSHR.MAR

```
; Version V3.2 - added on 31-Aug-1998, 19-Sep-1998, 02-Oct-1998
  vector  old                               seq=137 ; swrkdcl_context
  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_lnkfst_to_string        seq=162
  vector  swrk_scan_lnkfst                 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
```

Example 7-1 Cont'd on next page

Example 7-1 (Cont.) SWRKSHR.MAR

```
; 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_objt_
  vector  old                                 seq=193 ; swrk_find_objt_
  vector  old                                 seq=194 ; swrk_get_objt_re
  vector  old                                 seq=195 ; swrk_put_objt_re
  vector  swrk_remove_object                  seq=196
  vector  old                                 seq=197 ; swrk_update_objt_
  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
```

Example 7-1 Cont'd on next page

Example 7-1 (Cont.) SWRKSHR.MAR

```
; Version V3.3 - added on 28-Jul-2000
    vector    swrk_match_lnkfst                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_lnkfst        seq=255
```

Example 7-1 Cont'd on next page

Example 7-1 (Cont.) SWRKSHR.MAR

```
; 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_run_down_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
```

Example 7-2 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.

Example 7-2 SWRKSHR.OPT

Example 7-2 Cont'd on next page

Example 7-2 (Cont.) 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
```

Example 7-2 Cont'd on next page

Example 7-2 (Cont.) SWRKSHR.OPT

```
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
```

Example 7-2 Cont'd on next page

Example 7-2 (Cont.) SWRKSHR.OPT

```
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
```

Example 7-2 Cont'd on next page

Example 7-2 (Cont.) SWRKSHR.OPT

```
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
```

Example 7-3 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.

Example 7-3 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
```

Example 7-3 Cont'd on next page

Example 7-3 (Cont.) SWRKSHR-ALPHA.OPT

```
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,-
```

Example 7-3 Cont'd on next page

Example 7-3 (Cont.) SWRKSHR-ALPHA.OPT

```
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)
```

Example 7-3 Cont'd on next page

Example 7-3 (Cont.) SWRKSHR-ALPHA.OPT

```
! 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)
```

Example 7-3 Cont'd on next page

Example 7-3 (Cont.) SWRKSHR-ALPHA.OPT

```
! Added 28-Apr-2005 by SLJ
symbol_vector=(-
    swrk_dir_adalib=data, -
    swrk_sdc_adalib=data)
```

Example 7-4 is the linker options generated from the Macro-32 source for use with the OpenVMS Alpha shareable image.

Example 7-4 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)
```

Example 7-4 Cont'd on next page

Example 7-4 (Cont.) SWRKSHR-ALPHA.OPT_INC

```
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)
```

Example 7-4 Cont'd on next page

Example 7-4 (Cont.) SWRKSHR-ALPHA.OPT_INC

```
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)
```

Example 7-4 Cont'd on next page

Example 7-4 (Cont.) SWRKSHR-ALPHA.OPT_INC

```
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)
```

Example 7-4 Cont'd on next page

Example 7-4 (Cont.) SWRKSHR-ALPHA.OPT_INC

```
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)
```

Example 7-4 Cont'd on next page

Example 7-4 (Cont.) SWRKSHR-ALPHA.OPT_INC

```
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)
```

Example 7-4 Cont'd on next page

Example 7-4 (Cont.) SWRKSHR-ALPHA.OPT_INC

```
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)
```

Example 7-5 is a Macro-32 source which is used to build a protected shareable image.

Example 7-5 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"
```

Example 7-5 Cont'd on next page

Example 7-5 (Cont.) SWRKSHRPRV.MAR

```
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=001 ;
    vector   old                      seq=002 ;
    vector   old                      seq=003 ;

; Version V3.2 - Sorted on 02-Jun-1998
    vector   old                      seq=004 ;
    vector   old                      seq=005 ;
    vector   old                      seq=006 ;
    vector   old                      seq=007 ;
    vector   swrk_submit_batch_job_context mode=exec      narg=1  seq=008
    vector   old                      seq=009 ;

; Version V3.2-1 - added on 29-Jul-1998
    vector   swrk_run_detached_job      mode=exec      narg=5  seq=010

; Version V3.2-1 - added on 26-Dec-1998
    vector   swrk_get_powerhouse_resources mode=exec      narg=0  seq=011
    vector   swrk_set_privileges_off     mode=exec      narg=0  seq=012
    vector   swrk_set_privileges_on      mode=exec      narg=0  seq=013

; Version V3.2-1 - added on 14-Aug-1999
    vector   swrk_get_process_imagecount mode=exec      narg=2  seq=014

; Version V3.2-1 - added on 03-Jan-2000
    vector   old                      seq=015 ;
    vector   old                      seq=016 ;
    vector   old                      seq=017 ;
    vector   old                      seq=018 ;

; Version V3.3 - added on 11-Jan-2001
    vector   swrk_create_block          mode=exec      narg=3  seq=019
    vector   swrk_delete_block          mode=exec      narg=1  seq=020
    vector   swrk_setup_block           mode=exec      narg=3  seq=021

; Version V3.3 - added on 13-Mar-2001
    vector   swrk_get_chan_file_spec    mode=exec      narg=3  seq=022

; Version V3.4 - added on 09-Dec-2001
    vector   swrk_add_object_prv        mode=exec      narg=7  seq=023
    vector   swrk_find_class_prv        mode=exec      narg=3  seq=024
    vector   swrk_find_object_prv       mode=exec      narg=7  seq=025
    vector   swrk_find_object_id_prv    mode=exec      narg=7  seq=026
    vector   swrk_find_associations_prv mode=exec      narg=10 seq=027
    vector   swrk_remove_object_prv     mode=exec      narg=6  seq=028
    vector   swrk_rename_object_prv     mode=exec      narg=7  seq=029
    vector   swrk_update_object_prv     mode=exec      narg=7  seq=030

; Version V3.4 - added on 24-Dec-2001
    vector   swrk_get_msg_vec_item_prv  mode=exec      narg=3  seq=031
```

Example 7-5 Cont'd on next page

Example 7-5 (Cont.) SWRKSHRPRV.MAR

```
; Version V3.4 - added on 19-Mar-2002
    vector  swrk_clear_context_prv          mode=exec      narg=0  seq=032
    vector  swrk_clear_context_item_prv     mode=exec      narg=1  seq=033
    vector  swrk_set_context_item_prv       mode=exec      narg=2  seq=034
; Version V3.4 - added on 27-Jul-2002
    vector  swrk_add_association_prv        mode=exec      narg=7  seq=035
    vector  swrk_remove_association_prv     mode=exec      narg=7  seq=036
; Version V3.4 - added on 09-Aug-2002
    vector  swrk_cvt_scope_type_to_code_prv mode=exec      narg=2  seq=037
    vector  swrk_cvt_scope_type_to_ctl_prv  mode=exec      narg=2  seq=038
    vector  swrk_get_scope_context_prv      mode=exec      narg=4  seq=039
    vector  swrk_invalidate_scope_prv       mode=exec      narg=1  seq=040
; Version V3.4 - added on 01-May-2003
    vector  swrk_initialize_trace_2         mode=exec      narg=0  seq=041
; Version V3.4 - added on 22-Feb-2004
    vector  swrk_check_method_prv          mode=exec      narg=6  seq=042
    vector  swrk_execute_method_a_prv      mode=exec      narg=14 seq=043
; Version V3.4 - added on 01-May-2004
    vector  swrk_detach_swdatabase_all_prv mode=exec      narg=0  seq=044
    vector  swrk_find_resource_prv         mode=exec      narg=5  seq=045
; Version V3.4 - added on 02-Jun-2004
    vector  swrk_run_down_image_prv        mode=exec      narg=0  seq=046
; Reservations - added on 28-Nov-1998
end module
```

Example 7-6 is a Linker options file which is used to build a protected shareable image for OpenVMS VAX.

7.3.1 Transfer Vectors

For convenience, SysWorks provides a VECTOR macro which simplifies the task of defining transfer vectors and assists in enforcing the ordered nature of transfer vectors. This macro also works under OpenVMS Alpha, although it simply generates global references in an image root module rather than defining full transfer vectors.

SysWorks also provides a Macro-32 to linker options converter which uses the VECTOR statements in the Macro-32 source to generate a linker options file with the appropriate symbol_vector statements. This generated linker options file can be included with the shareable image linker options file by using the %INCLUDE directive in the header comments. This causes SysWorks to add the included (i.e the generated) options file to the link command at build time.

Example 7-6 SWRKSHRPRV.OPT

```
!++
!  
! Linker_options:  
!     SWRKSHRPRV-VAX  
!  
! Purpose:  
!     Linker options file for the SWRKSHRPRV shareable image for OpenVMS VAX.  
!  
! 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
```

7.4 Other Features

Other features of linking with SysWorks include:

- Automatic generation and inclusion of a linker options file which adds the current application version as the image identification using a linker options statements such as:

```
identification = "SWRK-VAX V3.4"
```

- Force the image to be linked without debug even when the /DEBUG qualifier is used with the BUILD command. This is achieved using the %NODEBUG directive in the header comments.

Rdb Database and CDD/Repository

This chapter describes the use of Rdb and CDD/Repository within an application.

8.1 Database

SysWorks Developer supports the use of an Rdb database in a number of ways. Firstly, a number of database instances may be considered as follows:

- *Compiletime*

The compiletime database of an application includes the domain and table schema definitions. It may also include other schema definitions but should not include storage areas and maps. The convention with SysWorks is to locate this database in the application library directory. If the schema needs to be shipped as part of the application installation kit, a backup of the database should be made to the software directory and hence the kit rather than including it directly in the kit.

- *Runtime*

The runtime database of an application includes the compile time domain and table schema plus any physical schema. The convention with SysWorks is to locate this database in the application data directory and its journals in the journal directory. Where multiple storage area files exist and these are to be placed on different disk volumes, multiple data directories should be created on different logical sub-disks.

8.1.1 Physical Database

Note that the physical schema of a runtime database may differ between environments. An example of this is:

- *Development*

The development runtime database includes physical schema such as constraints and indices but not storage areas and maps. It can be used for functional testing.

- *Testing*

The testing runtime database includes the development schema plus the storage areas and maps. It can then be used for functional and volume testing.

- *Production*

The production database may have some schema such as constraints removed and other extra schema such as extra indices for performance reasons.

8.1.2 Source

The database source may come from one of three locations as follows:

- *Script*

With this technique database domains and tables and optionally indices and triggers are placed in SQL scripts. The convention with SysWorks is to place each definition in a separate script. This results in minimum rebuilds because only modules which use a particular changed domain or table need to be recompiled. Note however that the compiletime database is rebuilt when any of its scripts change.

- *Database*

With this technique the database schema is presented to the application in the form of a database. This is typically used where a database modelling or administration group manages the schema and wishes to provide the application with a ready made database.

SysWorks provides tools for use with this technique which create a set of scripts as described above. The resultant scripts may be placed in the CMS library as though the script technique was being used, or they may be placed into the library directory.

- *Repository*

As a variant on the database technique, some sites may provide the database schema in the repository. This technique is supported by SysWorks by creating a database from the repository then using the database technique described above. Note however that this technique is not recommended since the repository does not directly support many of the database concepts such as keys and constraints.

Note that the database and repository technique reduce down to the script technique.

8.1.3 Central Control

As indicated earlier some sites may wish to have database schema managed centrally. An alternative to providing a database or repository is to provide the scripts directly to the application. One method for doing this whilst retaining control is to use a second application CMS library which is owned by the central group but can be read by the application. This CMS library can be placed in the CMS\$LIB search list.

8.1.4 Conventions and Tools

The convention with SysWorks is to name each of these scripts according to the following format:

```
appl_domain_DOM.SQL  
appl_index_IDX.SQL  
appl_table_TBL.SQL  
appl_trigger_TRG.SQL
```

If the database to script tool is used, this naming convention will be adhered to.

During the BUILD SCAN phase, SysWorks generates SQL scripts which are lists of the source scripts. These list scripts may be useful in creating the compiletime database schema. The generated scripts are:

```
appl_DOMAINS.SQL
appl_INDICES.SQL
appl_PROTECTIONS.SQL
appl_ROUTINES.SQL
appl_TABLES.SQL
appl_TRIGGERS.SQL
```

Except for the *appl_PROTECTIONS.SQL* script, each of the above is SQL script which executes all of the appropriate SQL source scripts in alphabetic order. The *appl_PROTECTIONS.SQL* script contains a series of SQL statements which apply the standard application security model to the database and the tables within the database.

The following root database script which by convention has a file name and type of *appl_DATABASE.SQL* provides an example of how these generated scripts may be used:

```
create database filename fin_lib_dir:fin_database;
set transaction read write;

@fin_lib_dir:fin_domains
@fin_lib_dir:fin_tables
@fin_lib_dir:fin_protections

commit;
```

The following MMS script fragment from the FIN application DESCRIP.MMS script provides an example of how to use SysWorks and MMS to create the compiletime database and provide an installation database backup file.

```
ALL depends_on -
    :
    :
    DATABASE, -
    :
    :

DATABASE depends_on FIN_SFT_DIR:FIN_DATABASE.RBF
    @ ! No action required

FIN_SFT_DIR:FIN_DATABASE.RBF depends_on FIN_LIB_DIR:FIN_DATABASE.RDB
    rmu/backup $(mms$source) $(mms$target)

FIN_LIB_DIR:FIN_DATABASE.RDB depends_on -
    FIN_WRK_DIR:FIN_DATABASE.SQL, -
    FIN_LIB_DIR:FIN_SCHEMA.TAG
```

8.2 Repository

SysWorks supports the use of CDD/Repository by generating CDO scripts from the database domain and table scripts. Only database scripts which change will have new CDO scripts generated. By keeping each database domain and table in a separate scripts, a minimum number of actions will be required to incrementally build the application.

The generated CDO script files have a file type of .CDO_GEN and are normally placed in the application library directory.

8.2.1 Fields

Each SQL domain script of the form *appl_domain_DOM*.SQL will have a CDO script of the form *appl_field_FLD*.CDO_GEN generated. Note that the actual field name is left the same as the domain name i.e. no application prefix or object type suffix is used. This is one of the few exceptions to the general rule whereby every object starts with the application code and finishes with an object type suffix.

For example *FIN_ACCOUNT_NO_DOM*.SQL may define the *ACCOUNT_NO* domain in the database schema and the generated *FIN_ACCOUNT_NO_FLD*.CDO_GEN would define the *ACCOUNT_NO* field in the repository.

An application may define fields in addition to the domain equivalences. Such fields would be defined in CDO sources of the form *appl_field_FLD*.CDO. Note that more than one field can be defined in a single source. This is useful for field definitions where conditional fields (eg. Cobol 88 levels) are defined with a datatype field or when a series of related fields are more conveniently defined together.

8.2.2 Records

Each SQL table script of the form *appl_table_TBL*.SQL will have a CDO script of the form *appl_record_REC*.CDO_GEN generated. Note that the actual record name is of the form *appl_table_REC* and that this limits the length of a table name to 26 minus the size of the application code characters in length.

For example *FIN_ACCOUNT_TBL*.SQL may define the *ACCOUNT* table in the database schema and the generated *FIN_ACCOUNT_REC*.CDO_GEN would define the *FIN_ACCOUNT_REC* record in the repository. Where a column is defined for a table using an explicit datatype or a domain with a different name, the generator will define fields for the columns before the record definition.

For the following example *FIN_ACCOUNT_TBL*.SQL:

```
create table account (
    account_no      account_no,
    account_desc    general_desc,
    account_total   bigint(2) );
```

The generated *FIN_ACCOUNT_REC*.CDO_GEN script would be:

```
define field ACCOUNT_DESC based on GENERAL_DESC.
define field ACCOUNT_TOTAL datatype quadword scale -2.
define record FIN_ACCOUNT_REC.
    ACCOUNT_NO.
    ACCOUNT_DESC.
    ACCOUNT_TOTAL.
end FIN_ACCOUNT_REC record.
```

An application may define records in addition to the table equivalences. Such fields would be defined in CDO sources of the form *appl_record_REC*.CDO. Note that more than one record can be defined in a single source, although this is not recommended. Records source may also contain field definitions. This is recommended for fields which are private to the record. Where a field is used by two or more records, a separate field source should be used.

Language Specifics

This chapter describes the various language and product specific actions, features and requirements when using SysWorks.

Table 9–1 provides a summary of the language support provided with SysWorks.

Table 9–1 Language Support Summary

Language or Product	Source File Types	Compile	Target Directory	Target File Types
ACA Services	.COL, .CRL	Yes	Library Software	.C, .MMS_INC .CO, .CR
ACMS	.CRL_IND	No	Library	.CRL, .MMS_INC
	.ADF, .GDF, .MDF, .TDF	Yes	Architecture Independent Library	.MMS_INC, .OPT, .OPT_INC, .TAG_CDD
	.ADF_INC, .GDF_INC, .MDF_INC, .TDF_INC	No	Software Architecture Independent Library	.ADB, .MDB, .TDB .MMS_INC
Ada ¹	.ADA	Yes	Library	.MMS_INC, .OBJ, .OLB, .TAG_EP
Basic ¹	.BAS	Yes	Library	.MMS_INC, .OBJ, .OLB, .TAG_EP
C	.C	Yes	Library	.MMS_INC, .OBJ, .OLB, .TAG_EP
	.H	No	Library	.MMS_INC, .TAG_INC_1, .TAG_INC_2
C++	.CXX	Yes	Library	.MMS_INC, .OBJ, .OLB, .TAG_EP
	.HXX	No	Library	.MMS_INC, .TAG_INC_1, .TAG_INC_2
	.IXX	No	Library	.MMS_INC, .TAG_INC_1, .TAG_INC_2
CDD	.CDDL, .DDL	Yes	Architecture Independent Library	.MMS_INC, .TAG_CDD
CDD/Plus & CDD/Repository	.CDO	Yes	Architecture Independent Library	.MMS_INC, .TAG_CDD
Cobol	.COB, .TXT	Yes	Library	.MMS_INC, .OBJ, .OLB, .TAG_EP, .TLB

¹The MMS generators for this language and its associated associated SQL precompiler generators will be released in a future version of SysWorks Developer.

Table 9–1 (Cont.) Language Support Summary

Language or Product	Source File Types	Compile	Target Directory	Target File Types
Command Language Definition	.CLD	Yes	Library	.MMS_INC, .OBJ, .TAG_EP
	.CLD_INC	No	Library	.MMS_INC, .TAG_EP
	.CLD_INC_SRC	Yes	Software	.CLD_INC
Datatrieve	.DTR	Yes	Architecture Independent Library	.MMS_INC, .TAG_CDD
DECdocument	.GRA, .SDML	Yes	Library	.HLP, .MMS_INC, .TAG_SDML, .TXT
	.SDML_INC	No	Library Documentation	.MMS_INC, .TAG_SDML .DECW\$BOOK, .DECW\$BOOKSHELF, .GIF, .HLB, .HTML, .PS, .RELEASE_NOTES
DECwindows	.UIL	Yes	Architecture Independent Library Software	.MMS_INC .UID
FMS	.FRM	Yes	Library	.MMS_INC, .OBJ
Fortran	.FOR	Yes	Library	.MMS_INC, .OBJ, .OLB, .TAG_EP
HTTP	.SHTML	Yes	Documentation Documentation	.MMS_INC .HTML
	.GIF_COPY, .HTIMAGE_COPY, .HTM_COPY, .HTML_COPY, .HTMLS_COPY, .HTMLX_COPY, .IMAGEMAP_COPY, .SHTML_COPY	Yes	Documentation	.GIF, .HTIMAGE, .HTM, .HTML, .HTMLS, .HTMLX, .IMAGEMAP, .SHTML
Java	.JAVA	Yes	Architecture Independent Library Software	.MMS_INC .CLASS
Linker	.OPT, .OPT_APP, .OPT_INC	Yes	Library	.MMS_INC, .OPT_EXP
Macro-32	.MAR, .MLB	Yes	Software Library	.EXE .MMS_INC, .OBJ, .OLB, .TAG_EP
ObjectBroker	.IDL	Yes	Software Library	.MLB .C, .MMS_INC
Pascal	.PAS	Yes	Software Library	.IR .MMS_INC, .OBJ, .OLB, .PEN, .TAG_EP

Table 9–1 (Cont.) Language Support Summary

Language or Product	Source File Types	Compile	Target Directory	Target File Types
PL/1 ¹	.PLI	Yes	Library	.MMS_INC, .OBJ, .OLB
PowerHouse	.QKS, .QTS, .QZS	Yes	Architecture Independent Library	.MMS_INC, .TAG_INC
	.QKS_INC, .QTS_INC, .QZS_INC	No	Architecture Independent Library	.MMS_INC, .TAG_INC
Rdb	.RDO	No	Software	.QKC, .QTC, .QZC
			Architecture Independent Library	.CDO_GEN, .MMS_INC, .TAG
	.RBA, .RC, .RCO, .RFO, .RPA	Yes	Architecture Independent Library	.MMS_INC, .OBJ, .OLB, .TAG_EP
	.SAD, .SBA, .SC, .SCO, .SFO, .SPA, .SPL	Yes	Architecture Independent Library	.MMS_INC, .OBJ, .OLB, .TAG_EP
	.SQL	No	Architecture Independent Library	.CDO_GEN, .MMS_INC, .TAG
	.SQLMOD	Yes	Architecture Independent Library	.MMS_INC, .OBJ, .OLB, .TAG_EP
Runoff	.RND	Yes	Architecture Independent Library	.MMS_INC
			Documentation	.DOC
	.RNH	Yes	Architecture Independent Library	.HLP, .MMS_INC
			Documentation	.HLB
	.RNM, .RNR	Yes	Architecture Independent Library	.MMS_INC
TDMS			Documentation	.MAN, .RELEASE_NOTES
	.FRM_TDMS, .RDF	Yes	Architecture Independent Library	.MMS_INC, .TAG_CDD
	.LDF	Yes	Architecture Independent Library	.MMS_INC, .TAG_CDD
			Software	.RLB

¹The MMS generators for this language and its associated associated SQL precompiler generators will be released in a future version of SysWorks Developer.

Table 9–1 (Cont.) Language Support Summary

Language or Product	Source File Types	Compile	Target Directory	Target File Types
DECdocument	.GRA, .SDML	Yes	Architecture Independent Library	.BRF, .MMS_INC
	.SDML_INC	No	Documentation	.DECW\$BOOK, .PS, .RELEASE_NOTES, .TXT
			Architecture Independent Library	.MMS_INC

- Source files.
Usually found in the work directory, but if they are generated they will be found in the library directory . Typical file types include .C, .COB, .COM_SRC, .COM_APP, .H, .FOR, .SDML, .SDML_INC and .UIL Should start with application code except for special cases. If not generated, they should reside in the CMS library.
- Intermediate files.
Usually found in the library directory. Typical file types include .LIBS, .MAP, OBJ and .OLB.
- Tag files.
Usually found in the library directory. Typical file types include .TAG_CDD, .TAG_EP, .TAG_INC_1 and .TAG_INC_2. These files are used to manage dependencies which are either inefficient or impractical to manage with direct MMS dependencies.
- Target files.
Usually found in the software or documentation directories. Typical file types include .COM, .EXE, .HLB, .HLP, .PS and .UID.

9.1 ACA Services

9.1.1 Contexts

Requirements for ACA Services contexts are summarized in Table 9–2.

Table 9–2 ACA Services Context Requirements

File Type	Feature	Semantics
.COL		ACA Services context source.
.CO		ACA Services context database.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.1.2 Repositories

Requirements for ACA Services repository are summarized in Table 9–3.

Table 9–3 ACA Services Repository Requirements

File Type	Feature	Semantics
.CRL, .CRL_ IND		ACA Services repository source.
.CR		ACA Services repository database.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.2 ACMS

9.2.1 Applications

Requirements for ACMA applications are summarized in Table 9–4.

Table 9–4 ACMS Application Requirements

File Type	Feature	Semantics
.ADF		ACMS application source.
	<code>_srv: ... server</code>	For each label ending in <code>_srv</code> : a search is made for the <i>server name</i> in <code>...</code> clause. If found, is indicated a dependency upon the associated <code>.TAG_IMG</code> tag.
	<code>%include ...</code>	Included source. Uses a dependency on a <code>.TAG_TRS</code> tag.
.ADB		ACMS application database generated by the <code>COMPILE</code> command or the <code>BUILD DESCRIP</code> phase.
.MMS_INC		Generated in the library directory by the <code>BUILD RULES</code> phase.
.TAG_TRS		Used to manage dependencies required by <code>%include ...</code> statements.
.TAG_IMG		Used to handle transitive dependencies to the servers used by the application.

The logical name `appl_ACMS_REPLACE` indicates whether to execute the `.ADF` source or use an appropriate ADU command. By default, this is assumed to have a value of false.

9.2.2 Menus

Requirements for ACMS menus are summarized in Table 9–5.

Table 9–5 ACMS Menu Requirements

File Type	Feature	Semantics
.MDF		ACMS menu source.
	<code>menu is ...</code>	Names must also exist as <code>.TDF</code> sources and must either start with <code>appl_</code> <code>MENUS</code> . or start with <code>appl</code> and finish with <code>_MNU</code> .
	<code>task is ...</code>	Names must also exist as <code>.TDF</code> sources and must either start with <code>appl_</code> <code>TASKS</code> . or start with <code>appl</code> and finish with <code>_TSK</code> .
.TAG_CDD		Used to manage dependencies required by the above statements.
.MDB		ACMS menu database generated by the <code>COMPILE</code> command or the <code>DESCRIP BUILD</code> phase.
.MMS_INC		Generated in the library directory by the <code>BUILD RULES</code> phase.

The logical name `appl_ACMS_REPLACE` indicates whether to execute the `.MDF` source or use an appropriate ADU command. By default, this is assumed to have a value of false.

9.2.3 Tasks

Requirements for ACMS tasks are summarized in Table 9–6.

Table 9–6 ACMS Task Requirements

File Type	Feature	Semantics
.TDF		ACMS task source.
	use workspaces ..., use workspace ..., workspaces are ..., worksspace is ... task is ..., tasks are ...	Names must also exist as .CDDL or .DDL sources and must either start with <i>appl</i> _DICTIONARY., <i>appl</i> _FIELDS., <i>appl</i> _RECORDS. or <i>appl</i> _REPOSITORY. start with <i>appl</i> and finish with _TSK. Names must also exist as .TDF sources and must either start with <i>appl</i> _TASKS. or start with <i>appl</i> and finish with _TSK.
.TAG_CDD		Used to manage dependencies required by the above statements.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

The logical name *appl*_ACMS_REPLACE indicates whether to execute the .TDF source or use an appropriate ADU command. By default, this is assumed to have a value of false.

9.2.4 Task Groups

Requirements for ACMS task group are summarized in Table 9–7.

Table 9–7 ACMS Task Group Requirements

File Type	Feature	Semantics
.GDF		ACMS task group source.
	form[s] [{is are}]	A target of the form <i>appl</i> FRM.EXE with a source of <i>appl</i> FRM.OPT and .TAG_EP dependencies for each form will be generated. The <i>appl</i> FRM.OPT file should also be generated - see .OPT file type below for details.
	server[s] [{is are}]	
	cancel procedure	
	dcl process	A target of the form <i>server-name</i> .TAG_IMG with a source of <i>appl</i> _GRP.TDB is generated.
	default object file	
	{initial initialization} procedure [is]	Generates a .TAG_EP dependency as described for procedures are below.
	procedure [server] image [is]	
	procedure[s] [{is are}]	Generates a .TAG_EP dependency for each procedure listed.
	{terminal termination} procedure [is]	Generates a .TAG_EP dependency as described for procedures are above.
	task[s] [{is are}]	Names must also exist as .TDF sources and must either start with <i>appl</i> _TASKS. or start with <i>appl</i> and finish with _TSK.
	workspace[s] [{is are}]	Names must also exist as .CDO sources and must either start with <i>appl</i> _RECORDS. or start with <i>appl</i> and finish with _REC. The with name syntax is supported by generating a target for the alternative name based on the original name.
.TAG_CDD		Used to manage dependencies required by the above statements.
.TDB		ACMS task database generated by the COMPILE command or the BUILD DESCRIP phase.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.
.OPT		May be generated in the library directory to build a forms shareable image with a name of <i>appl</i> FRM. To use this feature, a DEVTOOLS CONVERT/GENERATE <i>appl</i> _GRP.GDF <i>appl</i> FRM.OPT command is required.

The logical name *appl*_ACMS_REPLACE indicates whether to execute the .GDF source or use an appropriate ADU command. By default, this is assumed to have a value of false.

9.3 ADA

Requirements for ADA are summarized in Table 9–8.

Table 9–8 ADA Requirements

File Type	Feature	Semantics
.ADA		ADA source.
.LIS, .OBJ		Generated in the library directory by the COMPILE command or the BUILD DESCRIP phase.
<i>app</i> OBJ.OLB		Created in the library directory to store the resultant object module.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.4 Basic

Requirements for Basic are summarized in Table 9–9.

Table 9–9 Basic Requirements

File Type	Feature	Semantics
.BAS		Basic source.
.LIS, .OBJ		Generated in the library directory by the COMPILE command or the BUILD DESCRIP phase.
<i>app</i> OBJ.OLB		Created in the library directory to store the resultant object module.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.5 C

Requirements for C are summarized in Table 9–10.

Table 9–10 C Requirements

File Type	Feature	Semantics
.C		C source.
	#dictionary	Used to indicate a dependency on a CDD/Repository element. If the object is a record, its name must either start with <i>appl</i> _RECORDS. or start with <i>appl</i> and finish with _REC. If the object is a field, its name must either start with <i>appl</i> _FIELDS. or not finish with _REC.
	#include	Used to indicate a dependency on an include (i.e. .H) source.
	extern	Used with a function prototype to indicate a transitive (i.e. link time) dependency on an entry point in another module.
	main	The presence of a <i>main</i> function indicates that this source forms the root module for an image. As a result, no dependency is generated to insert the resulting object module into the object library.
.H		C include source.
	#dictionary	See C source.
	#include	See C source.
	extern	See C source
.TAG_CDD		Used to manage dependencies required by #dictionary statements.
.TAG_INC_1		Used to manage dependencies caused by nested #include statements.
.TAG_INC_2		Used to manage transitive dependencies caused by extern statements in #include sources.
.LIS, .OBJ		Generated in the library directory by the COMPILE command or the BUILD DESCRIP phase.
<i>appl</i> OBJ.OLB		Created in the library directory to store the resultant object module. Note that this module is not inserted if the .C source contained a <i>main</i> function.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

A common practise with C programs is to have a function definition contained in a .H file which is included in both the .C source which defines the function and any .C source which uses the function. This is possible because the C language allows the presence on the *extern* clause even when the function is defined within s source. The following three examples illustrate such a scheme.

In *fin_our_func.h* would be the following code fragments:

```
extern int fin_our_func ( int arg_1, *char arg_2 );
```

In *fin_our_func.c* would be the following code fragments:

```
#include "fin_wrk_dir:fin_our_func.h"
.
.
.
int fin_our_func ();
{
.
.
.
};
```

In `fin_main_prog.c` would be the following code fragments:

```
#include "fin_wrk_dir:fin_our_func.h"
.
.
.
main ()
{
.
.
.
    status = fin_our_func ( 25, some_text );
.
.
.
};
```

On the other hand, SysWorks analyses dependencies on a per source basis i.e. it does not analyse all the sources in an application simultaneously in order to generate the final MMS script, rather it analyses each source independently (usually only when they have been updated) and generates a corresponding MMS script, with the set of these MMS scripts being merged to form the final script. As a result of using this technique, the three examples above lead to a circular dependency between `fin_our_func.c` and `fin_our_func.h` because:

- There is a compile time dependency within `fin_our_func.c` on the include source `fin_our_func.h`.
- There is a link time dependency for any source which includes `fin_our_func.h` on the associated entry point in the module created from the `fin_our_func.c` source.

Since MMS cannot distinguish between compile time and link time dependencies or cope with co-dependencies, it treats these cases as circular dependencies and generates errors.

To solve this problem, one solution is to split the include file into two resulting in the following four examples illustrate this technique.

In `fin_our_func.h` would be the following code fragments:

```
#include "fin_wrk_dir:fin_our_func_2.h"
.
.
.
extern int fin_our_func ();
```

In `fin_our_func_2.h` would be the following code fragments:

```
int fin_our_func ( int arg_1, *char arg_2 );
```

In `fin_our_func.c` would be the following code fragments:

```
#include "fin_wrk_dir:fin_our_func_2.h"
.
.
.
int fin_our_func ();
{
.
.
.
};
```

In `fin_main_prog.c` would be the following code fragments:

```
#include "fin_wrk_dir:fin_our_func.h"
.
.
main ();
{
.
.
.
    status = fin_our_func ( 25, some_text );
.
.
.
};
```

9.6 C++

Requirements for C++ are summarized in Table 9–11.

Table 9–11 C++ Requirements

File Type	Feature	Semantics
.CXX		C++ source.
	#dictionary	Used to indicate a dependency on a CDD/Repository element. If the object is a record, its name must either start with <i>appl</i> _RECORDS. or start with <i>appl</i> and finish with _REC. If the object is a field, its name must either start with <i>appl</i> _FIELDS. or not finish with _REC.
	#include extern	Used to indicate a dependency on an include (i.e. .HXX and .IXX) source. Used with a function prototype to indicate a transitive (i.e. link time) dependency on an entry point in another module.
	main	The presence of a <i>main</i> function indicates that this source forms the root module for an image. As a result, no dependency is generated to insert the resulting object module into the object library.
.HXX, .IXX		C++ include source.
	#dictionary	See C++ source.
	#include extern	See C++ source. See C++ source
.TAG_CDD		Used to manage dependencies required by #dictionary statements.
.TAG_INC_1		Used to manage dependencies caused by nested #include statements.
.TAG_INC_2		Used to manage transitive dependencies caused by extern statements in #include sources.
.LIS, .OBJ		Generated in the library directory by the COMPILE command or the BUILD DESCRIP phase.
<i>appl</i> OBJ.OLB		Created in the library directory to store the resultant object module. Note that this module is not inserted if the .C source contained a <i>main</i> function.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.7 CDD

Requirements for CDD are summarized in Table 9–12.

Table 9–12 CDD Requirements

File Name Suffix and File Type	Feature	Semantics
_REC.CDDL, _REC.DDL		CDD record source.
	<code>define record</code>	Record definition. The name must either start with <i>appl</i> _RECORDS. or start with <i>appl</i> and finish with _REC.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.8 CDD/Repository

Requirements for CDD/Repository are summarized in Table 9–13.

Table 9–13 CDD/Repository Requirements

File Name Suffix and File Type	Feature	Semantics
_FLD.CDO		CDD/Repository field source.
	<code>define field</code>	Global field definition. Field names should either start with <i>appl_FIELDS</i> . or not finish with <i>_REC</i> .
_REC.CDO		CDD/Repository record source.
	<code>define field</code>	Local field definition. Same rules apply as for global fields.
	<code>define record</code>	Record definition. The name must either start with <i>appl_RECORDS</i> . or start with <i>appl</i> and finish with <i>_REC</i> .
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.9 Cobol

Requirements for Cobol are summarized in Table 9–14.

Table 9–14 Cobol Requirements

File Type	Feature	Semantics
.COB		Cobol source.
	filename	Must start with application code. Should reside in the CMS library.
	program-id	Must be the same as the filename.
	copy ...	Copy in a source file. The file should reside in the work directory.
	copy ... from dictionary	Used to indicate a dependency on a CDD/Repository element. If the object is a record, its name must either start with <i>app/RECORDS.</i> or start with <i>app/</i> and finish with <i>_REC.</i> If the object is a field, its name must either start with <i>app/FIELDS.</i> or not finish with <i>_REC.</i>
	copy ... in ...	Copy a source from a library. The library should reside in the library directory. It should have a file name of <i>app/TXT.TLB.</i>
	call	Used to create a transitive dependency via <i>.TAG_EP.</i> Called program name must be in quotes.
.TAG_CDD		Used to manage dependencies required by <i>copy ... from dictionary</i> statements.
.TAG_CPY_1		Used to manage dependencies caused by nested <i>copy ...</i> and <i>copy ... in ...</i> statements.
.TAG_CPY_2		Used to manage transitive dependencies caused by <i>call</i> statements in copy files and libraries.
.LIS, .OBJ		Generated in the library directory by the <i>COMPILE</i> command or the <i>BUILD DESCRIP</i> phase.
<i>app</i> OBJ.OLB		Created in the library directory to store the resultant object module.
.MMS_INC		Generated in the library directory by the <i>BUILD RULES</i> phase.

9.10 DCL

Requirements for DCL command procedures are summarized in Table 9–15.

Table 9–15 DCL Command Procedure Requirements

File Type	Feature	Semantics
.COM_SRC		DCL command procedure source.
.COM_APP		DCL command procedure append source.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.11 DECdocument

Requirements for DECdocument are summarized in Table 9–16.

Table 9–16 DECdocument Requirements

File Type	Feature	Semantics
.SDML		DECdocument source. See Table 9–17 for file name suffix conventions.
	<CONTENTS_FILE>	Include /CONTENTS in the generated DOCUMENT command. This tag must only be used in a .SDML source - it is not recognised in a .SDML_INC include source.
	<EXAMPLE_FILE>, <FIGURE_FILE>, <INCLUDE>	Generate a dependency on the specified include file.
	<INDEX_FILE>	Include /INDEX in the generated DOCUMENT command. This tag must only be used in a .SDML source - it is not recognised in a .SDML_INC include source.
.SDML_INC		DECdocument include source.
	<EXAMPLE_FILE>, <FIGURE_FILE>, <INCLUDE>	See .SDML source.
.GRA		DECdocument graphic source.
.INX_LIST		Used during the creation of a master index.
.COM, .EPS, .HLP, .INT_ TOC, .INX, .LIS, .MAP_LIS, .TAG_INX, .TAG_SDML, .TEX		Generated in the library directory by the COMPILE command or the BUILD DESCRIP phase.
.DECW\$BOOK, .GIF, .HTML, .PS, .TXT		Generated in the documentation directory by the COMPILE command or the BUILD DESCRIP phase.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

File name suffix conventions are listed in Table 9–17.

Table 9–17 DECdocument File Name Suffixes

File Name Suffix	Usage
_CVR	Cover note source for generating PostScript and text output.
_GDE	Guide source for generating bookreader, master index and PostScript output.
_HLP	Help source.
_IDX	Master index sources for generating PostScript output. Two sources must exist, one with a file type of .SMDL and the other of a file type of .INX_LIST.
_INS	Guide source for generating bookreader, PostScript and text output.
_OVH	Overhead source for generating PostScript output.
RELEASE NOTES	Release notes source for generating bookreader, PostScript and .RELEASE_NOTES output.
_REF	Reference manual source for generating bookreader, master index and PostScript output.

9.11.1 Graphics

Because DECdocument graphics renditions may need different scaling on a per figure or format basis, SysWorks cannot use implicit MMS rules for generating them. Therefore, each application using VAX DOUMENT graphics needs to declare an explicit set of rules. A typical way of doing this is to have a line in the applications DESCRIP.MMS which includes another application MMS script.

For example:

```
!++
!
! MMS Script:
! SWDOC_GRAPHICS
!
! Purpose:
! Create graphics targets for DECdocument.
!
!--

SWRK_LIB_DIR:SWRK_DECW_SM.BRF, -
SWRK_LIB_DIR:SWRK_DECW_SM.EPS, -
SWRK_LIB_DIR:SWRK_DECW_SM.SDML -
  depends_on -
    SWRK_WRK_DIR:SWRK_DECW_SM.GRA
  @ if f$trnlnm("decw$display") .eqs. "" then set display/create
  $(docbrfeps) $(mms$source)/output=$(mms$target)/scale=(x=75,y=75,exclude=brf)

SWRK_LIB_DIR:SWRK_DISKS_FIGURE.BRF depends_on SWRK_WRK_DIR:SWRK_DISKS_FIGURE.GRA
  @ if f$trnlnm("decw$display") .eqs. "" then set display/create
  $(docbrf) $(mms$source)/output=$(mms$target)/scale=(x=67,y=67)

SWRK_LIB_DIR:SWRK_DISKS_FIGURE.EPS depends_on SWRK_WRK_DIR:SWRK_DISKS_FIGURE.GRA
  @ if f$trnlnm("decw$display") .eqs. "" then set display/create
  $(doceps) $(mms$source)/output=$(mms$target)/scale=(x=50,y=50)
```

In this example

- A line such as `.INCLUDE SWRK_WRK_DIR:SWRK_GRAPHICS.MMS` would be required in the applications DESCRIP.MMS.
- The SWRK_DECW_SM graphics file needs a 25% reduction for its postscript rendition while retaining its original size for the bookreader version. This can be achieved in a single rule as indicated. This method would also be used if neither format required scaling or both formats required the same scaling.
- The `docbrfeps` macro is defined in the standard SysWorks MMS rules file as generating both a .BRF and a .EPS rendition.
- The SWRK_DISKS_FIGURE graphics file however, requires two rules since the two renditions require different scaling.
- The `docbrf` macro is defined in the standard SysWorks MMS rules file as generating a .BRF and the `doceps` macro a .EPS one.
- The DECW\$DISPLAY definition is required to make the DOCUMENT/GRAPHICS=RENDER command work. If a node on which the documentation is to be built does not have a display device, it is necessary to create the display on a node which does.

9.12 DECforms

Requirements for DECforms are summarized in Table 9–18.

Table 9–18 DECforms Requirements

File Type	Feature	Semantics
.IFDL		DECforms source.
	form	Must be the same as the filename. Should start with the application code and finish with <code>_FRM</code> .
	copy ...	Copy in a source file. The file should reside in the work directory.
	copy ... from dictionary	Used to indicate a dependency on a CDD/Repository element. If the object is a record, its name must either start with <code>appl_RECORDS</code> , or start with <code>appl</code> and finish with <code>_REC</code> . If the object is a field, its name must either start with <code>appl_FIELDS</code> , or not finish with <code>_REC</code> .
	copy ... in ... call	Copy a source from a library. The library should reside in the library directory. It should have a file name of <code>appl/TXT.TLB</code> . Used to create a transitive dependency via <code>.TAG_EP</code> . Called program name must be in quotes.
.TAG_CDD		Used to manage dependencies required by <code>copy ... from dictionary</code> statements.
.TAG_CPY_1		Used to manage dependencies caused by nested <code>copy ...</code> and <code>copy ... in ...</code> statements.
.TAG_CPY_2		Used to manage transitive dependencies caused by <code>call</code> statements in copy files and libraries.
.FORM		DECforms intermediate binary format.
.LIS, .OBJ		Generated in the library directory by the <code>COMPILE</code> command or the <code>BUILD DESCRIP</code> phase.
<i>appl</i> OBJ.OLB		Created in the library directory to store the resultant object module.
.MMS_INC		Generated in the library directory by the <code>BUILD RULES</code> phase.

9.13 FMS

Requirements for FMS are summarized in Table 9–19.

Table 9–19 FMS Requirements

File Type	Feature	Semantics
.FRM		FMS form source.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.14 Fortran

Requirements for Fortran are summarized in Table 9–20.

Table 9–20 Fortran Requirements

File Type	Feature	Semantics
.FOR		Fortran source.
.LIS, .OBJ		Generated in the library directory by the COMPILE command or the BUILD DESCRIP phase.
<i>app</i> OBJ.OLB		Created in the library directory to store the resultant object module.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.15 Macro

Requirements for Macro-32 are summarized in Table 9–21.

Table 9–21 Macro-32 Requirements

File Type	Feature	Semantics
.MAR		Macro-32 source.
	.title	Used to determine the name of the module. This directive must be present and specify the same name as the source file name.
_MAC.MAR		Macro-32 macro only source.
	.title	Used to determine the name of the module. This directive must be present and specify the same name as the source file name.
appMAC.MLB		Created in the software directory to store macros found in _MAC.MAR macro only sources.
	.macro	Beginning of a macro in a macro only source. Nested macros are ignored.
	\$defini	If found on the line immediately following a .macro directive, the \$defini directive indicates that dependencies for an object module should to define the symbols contained within the macro should be generated. If the macro name starts with a period, no dependency to update the macro library is generated as MMS does not support objects starting with a period. If a \$defini directive is not found, only the dependencies required to insert the macro into the macro library are generated.
	.endm	End of a a macro in a macro only source.
.LIS, .OBJ		Generated in the library directory by the COMPILE command or the BUILD DESCRIP phase.
appOBJ.OLB		Created in the library directory to store the resultant object module.
.MMS_INC		Generated in the library directory by the BUILD RULES phase. The format differs between general Macro sources and macro only sources.

9.16 ObjectBroker

9.16.1 Contexts

Requirements for ObjectBroker contexts are summarized in Table 9–22.

Table 9–22 ObjectBroker Context Requirements

File Type	Feature	Semantics
.COL		ObjectBroker context source.
.CO		ObjectBroker context database.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.16.2 Repositories

Requirements for ObjectBroker repository are summarized in Table 9–23.

Table 9–23 ObjectBroker Repository Requirements

File Type	Feature	Semantics
.IDL		ObjectBroker repository source.
.IR		ObjectBroker repository database.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.17 Oracle Rdb

Requirements for Oracle Rdb are summarized in Table 9–24.

Table 9–24 Oracle Rdb Requirements

File Name Suffix and File Type	Feature	Semantics
_CK <i>n</i> .SQL		Check constraint definition.
_DOM.SQL		Domain definition.
_FK <i>n</i> .SQL		Foreign key constraint definition.
_MAP.SQL		Storage map definition.
_PK.SQL		Primary key constraint definition.
_PIDX.SQL		Primary index definition.
_RTN		Routine definition.
_SIDX <i>n</i>		Secondary (alternative) index definition.
_TBL.SQL		Table definition.
_TRG.SQL		Trigger definition.
_TRGR.SQL		
_VIEW.SQL		View definition.
_VW.SQL		
.SQLMOD		Module source.
	procedure	Used to create a transitive dependency via .TAG_EP.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

The following SQL include lists are generated in the application architecture independent library directory during the BUILD SCAN phase:

- *appl_CONSTRAINTS.SQL*
- *appl_DOMAINS.SQL*
- *appl_INDICES.SQL*
- *appl_ROUTINES.SQL*
- *appl_STORAGE_MAPS.SQL*
- *appl_TABLES.SQL*
- *appl_TRIGGERS.SQL*
- *appl_VIEWS.SQL*

The following MMS dependency list is generated in the application architecture independent library directory during the BUILD SCAN phase:

- *appl_SCHEMA.MMS*

9.18 Pascal

Requirements for Pascal are summarized in Table 9–25.

Table 9–25 Pascal Requirements

File Type	Feature	Semantics
.PAS		Pascal source.
	filename	Must start with application code. Should reside in the CMS library.
	module	Must be the same as the filename.
	environment	Should be the same file name as the source, but in the library directory and with a file type of .PEN. The .PEN file type is assumed by default.
	inherit	Should be in the library directory and with a file type of .PEN. The .PEN file type is assumed by default.
.LIS, .OBJ		Generated in the library directory by the COMPILE command or the BUILD DESCRIP phase.
<i>app</i> OBJ.OLB		Created in the library directory to store the resultant object module.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.19 Runoff

Requirements for Runoff are summarized in Table 9–26.

Table 9–26 Runoff Requirements

File Type	Feature	Semantics
.RNH		Runoff help source.
.RNM		Runoff manual source.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.20 TDMS

9.20.1 Forms

Requirements for TDMS forms are summarized in Table 9–27.

Table 9–27 TDMS Forms Requirements

File Type	Feature	Semantics
.FRM_TDMS		CDD backup of TDMS form used as a source.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

9.20.2 Requests

Requirements for TDMS requests are summarized in Table 9–28.

Table 9–28 TDMS Request Requirements

File Type	Feature	Semantics
.RDF		TDMS Request source.
	form is ..., forms are ...	Names must also exist as .FRM_TDMS sources and must either start with <i>appl_FORMS</i> . or start with <i>appl</i> and finish with <i>_FRM</i> .
	record is ..., records are ...	Used to indicate a dependency on a CDD element. If the object is a record, its name must either start with <i>appl_RECORDS</i> . or start with <i>appl</i> and finish with <i>_REC</i> . If the object is a field, its name must either start with <i>appl_FIELDS</i> . or not finish with <i>_REC</i> .
.TAG_CDD		Used to manage dependencies required by the above statements.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

The logical name *appl_TDMS_REPLACE* indicates whether to execute the .RDF source or use an appropriate RDU command. By default, this is assumed to have a value of false.

9.20.3 Request Libraries

Requirements for TDMS request libraries are summarized in Table 9–29.

Table 9–29 TDMS Request Library Requirements

File Type	Feature	Semantics
.LDF		TDMS request library source.
	request is ..., requests are ...	Used to indicate a dependency on a TDMS request definition. Its name must either start with <i>appl_REQUESTS</i> . or start with <i>appl</i> and finish with <i>_REQ</i> .
.TAG_CDD		Used to manage dependencies required by the above statements.
.RLB		Generated in the software directory by the COMPILE command or the BUILD DESCRIP phase.
.MMS_INC		Generated in the library directory by the BUILD RULES phase.

The logical name *appl_TDMS_REPLACE* indicates whether to execute the .LDF source or use an appropriate RDU command. By default, this is assumed to have a value of false.

Naming Conventions

This chapter describes the suggested application object naming conventions. Note that while some language specific issues are discussed, further information is available in Chapter 9, Language Specifics.

appMSG

The application message shareable image. Relevant file types include .EXE and .OPT. A logical name should be defined for this file.

appRDBSHR

The application database shareable image. This additional shareable image is used when an application has a substantial number of common routines and calling images which don't need the database. These would be put into the application shareable image. The routines which need to use the database would be placed in the application database shareable image. Relevant file types include .EXE and .OPT. A logical name should be defined for this file.

appSHR

The application shareable image. Relevant file types include .EXE and .OPT. A logical name should be defined for this file.

appSHRPRV

The application protected shareable image. Relevant file types include .EXE and .OPT. A logical name must be defined for this file.

appl_envr_APL

The ACMS application definition for the whole application in the specified environment. This object is a dictionary based object. Relevant file types include .ADF. A .TAG_CDD file is placed in the library directory after this source is successfully loaded into the dictionary.

appl_DATABASE

An SQL script used to create a database. Relevant file types include .SQL and .SQL_CPY .

appl_DOMAINS

The set of database domains. Relevant file types include *.SQL* .

appl_FIELDS

The set of CDD/Plus dictionary fields. There are basically three types of fields that should be defined in the common set, these being database domains, database columns, and common record fields. Fields specific to single record should be placed in the source for the record (i.e. an *appl_object-code_REC*). This object is effectively a set of dictionary objects. Relevant file types include *.CDO* , *.CDO_GEN* and *.TAG_CDD* .

appl_GRP

The ACMS task group definition for the whole application. This object is a dictionary based object. Relevant file types include *.GDF*.

appl_MESSAGES

The messages for the application. This source is separately compiled into two objects, *appl_MESSAGES.OBJ* and *appl_MESSAGES.SYM*. The first object, *appl_MESSAGES.OBJ*, is compiled with message text, but without the symbol values. It is then linked into the *applMSG.EXE* shareable image. The second object, *appl_MESSAGES.SYM*, is compiled with the symbol values, without text, and with a pointer to the *applMSG.EXE* shareable image. It is then placed into the *applOBJ* object library, from where it is linked into the various executable images, such as *appl_PROC_SRV.EXE*.

appl_MNU

The root menu for the application which should consist of entries for the top menu of each user class. Users would then be given access to the appropriate user class menu directly below the *appl_MNU* menu. Some super users may be given direct access to the root menu.

appl_PROC_SRV

An ACMS server used to perform read only transactions. Relevant file types include *.EXE*, *.OBJ* and *.OPT*. The object module is produced by ACMS when the *appl_GRP* task group is built. The options file is used to link the server image. Note that both the *appl_GRP* task group definition and the linker options file must list the procedures to be contained in the image.

appl[_type]_object-code_CC

An SQL module procedure to close the associated *appl[_type]_object-code_CSR* cursor. Note that this procedure must be called after the *appl[_type]_object-type_FC* procedure returns the *SQL_END_OF_CURSOR* status.

appl_table_CKn

An SQL script used to create a database table check constraint. Relevant file types include .SQL and .SQL_CPY .

Since there can be more than one check constraint on a table, a numeric suffix is used.

appl_object-code_COM

A report or other batch DCL command procedure. Relevant file types include .COM , .COM_CPY and .COM_SRC .

appl[_type]_object-code_CSR

An SQL module cursor. There should be an associated *appl[_type]_object-code_OC* procedure to open the cursor, a *appl[_type]_object-code_FC* procedure to fetch the rows from the cursor, and a *appl[_type]_object-code_CC* procedure (which must be called) to close the cursor.

appl_table_DIDX

An SQL script used to create a database direct (i.e. hashed) index. Relevant file types include .SQL and .SQL_CPY .

Although a database doesn't distinguish between different index uses, this naming convention is suggested to clarify the index usage for developers and database administrators.

If the direct index is not used for clustering, it is based around the primary key, and is normally stored in a mixed storage area with the data of the associated table.

If it is involved in clustering, it is based around the first segment of the primary key and is normally stored in a mixed storage area with the data of the associated and parent tables.

appl_domain_DOM

An SQL script used to create a database domain. Relevant file types include .SQL and .SQL_CPY .

appl_file-or-table_DOM

A Datatrieve script used to define a domain on a file or database table.

appl[_type]_object-code_DR

An SQL module procedure to delete rows from the table indicated by the object-code. The type variants are used when there are differing delete options for the table. When only a single row based on the primary key should be deleted there should only be one *appl_object-code_DR* procedure.

appl[_type]_object-code_FC

An SQL module procedure to fetch a row from the the associated *appl[_type]_object-code_CSR* cursor. This procedure will eventually return the *SQL_END_OF_CURSOR* status, whereupon the *appl[_type]_object-code_CC* procedure must be called.

appl_table_FKn

An SQL script used to create a database foreign key. Since there can be more than one foreign key for a table, a numeric suffix is used. There should be a foreign key definition for each relationship between tables.

Note that the use of foreign keys to reference data is encouraged, but using them largely prevents the ability to delete reference rows. This is because the table which uses the reference data would not normally have an index on the columns used to access the reference table, and hence the using table would be sequentially scanned to ensure that it didn't use the reference data which is to be deleted.

appl_object_code_FNC

The function name used within *appl_object-code_KEY* and *appl_object-code_RSP* DECforms include texts.

appl_routine_FNC

An SQL script used to create a function definition. Relevant file types include *.SQL* and *.SQL_CPY*.

The function ***appl_routine*** is normally an entry point on the *applSHR* sharable image.

appl[_type]_object-code_FRM

A form used to manipulate an object-code. Note that for most forms, there will be a corresponding ACMS task. Some common forms such as the *appl_WORKING_FRM* are shared among some or all tasks. Table 10–1 provides a list of form and task object types.

Table 10–1 Form and Task Types

Type	Usage
APPLY	Apply an operation or deferred update to the set of object-codes.
FIND	Select an object-code from the set of object-code's. Use paging where required to select from a large number of object-codes.
MNT	Maintain an object-code. Where appropriate icons would be used to select one of a predetermined set of values for fields, and calls to FIND tasks would be used to select values for validated fields.
RCV	Receive object-code data from an external source. Typically, this will submit a batch job.
RPT	Generate a report about object-codes. Typically this will submit a batch job.
SND	Send a set of object-codes to some external system.

Typically a form manipulates information about the object-code on panel (*appl_object-code_PNL*). If the user presses the list function key (i.e. Gold L) on a key field, panel *appl_object-code_PG_PNL* is used to display the set of object-code's a page at a time. In most cases this paging will need to return to the task to get the next or previous page. Scrolled pages within DECforms should only be considered when the total table size is limited by nature (rather than by arbitrary decision) to less than 2Kb. Relevant file types include .FORM, .IFDL and .OBJ.

appl[_type]_object-code_GRP A temporary ACMS task group used to assist in debugging the *appl[_type]_object-code_TSK* task. This object is a dictionary based object. Relevant file types include .GDF and .TDB.

appl[_type]_object-code_IR

An SQL module procedure to insert a row into the table indicated by the object-code. The type variants are used when there are differing partial insertions for the table. When only full insertion (i.e. no columns missing and no NULLs resulting) is required, there should only be one *appl_object-code_IR* procedure.

Table 10–2, SQL Module Procedure Suffices for Inserting a List of Byte Varying lists the suggested names for the SQL module procedures where a table has a *list of byte varying* column.

Table 10–2 SQL Module Procedure Suffices for Inserting a List of Byte Varying

Suffix	Usage
<i>_IR_OC</i>	Open the insert table cursor.
<i>_IR_IC</i>	Insert the table row into the table cursor.
<i>_IR_IC_OC</i>	Open the insert list cursor.
<i>_IR_IC_IC</i>	Insert a segment into the list cursor.
<i>_IR_IC_CC</i>	Close the insert list cursor.
<i>_IR_CC</i>	Close the insert table cursor.

appl_object-code_KEY

A copy text for defining a function in a DECforms IFDL module. For each *appl_object-code_KEY* copy text there should be a corresponding *appl_object-code_RSP* copy text. This object is a text library based object. Relevant file types include .TXT.

Examples include:

- *appl_FIND_KEY*

This copy text describes the find key definition that should be used in every DECforms form which performs a table lookup. Note that there is no standard response copy text because responses are position dependent eg. some responses are valid across a form, while others change at some fields or are only valid at some fields.

- *appl_STD_KEY*

This copy text describes the standard key definitions that should be used in every DECforms form. Note that there is no standard reponse copy test because responses are position dependent eg. some responses are valid across a form, while others change at some fields or are only valid at some fields.

appl_object-code[_type]_LST

A DECforms record list. The list would normally correspond to a panel of the name *appl_object-code_PNL*. The exception is when a separate send and receive record list are used with a tranceive external response.

Examples include:

- *appl_object-code_RCV_LST*
A DECforms record list used for receiving from a tranceive external response when its send and receive data requirements are substantially different. When the requirements are not substantially different, the *appl_object-code_LST* name should be used.
- *appl_object-code_SND_LST*
A DECforms record list used for sending to a tranceive external response when its send and receive data requirements are substantially different. When the requirements are not substantially different, the *appl_object-code_LST* name should be used.

appl[_type]_object-code_OC

An SQL module procedure to open the associated *appl[_type]_object-code_CSR* cursor. *appl_object-code[_type]_PNL* A DECforms panel. The panel may appear directly in a form definition or be included from the copy library. Relevant file types include .TXT (when used via a copy library).

appl_table_PIDX

An SQL script used to create a database primary index. Relevant file types include .SQL and .SQL_CPY .

Also, an SQL module used to access a table via its primary index. Relevant file types include .OBJ and .SQLMOD . Typically this module would include procedures of the form *appl[_type]_object-code_OC*, *appl[_type]_object-code_FC* and *appl[_type]_object-code_CC*.

Although a database doesn't distinguish between primary and secondary or alternate indices, this naming convention is suggested to clarify the index usage for developers and database administrators. The primary index is based around the primary key. It would be used by application code to perform simple selections and by the database to validate the primary key.

appl_table_PK

An SQL script used to create a database primary key. Relevant file types include .SQL and .SQL_CPY .

There should be one primary key definition for each database table. This source is kept separate from the database table definition in order to keep to the model of one object per source and to allow the key to be dropped and reapplied during large scale databa operations if necessary.

appl_object-code_[PG[n]]_PNL

A DECforms panel. The panel may appear directly in a form definition or be included form the copy library. The **PG_n** format is used when a form contains more than one panel.

appl[_type]_object-code_PROC

A 3GL procedure to use in an ACMS procedure server. Typical file types include 3GL file types and .LIS and .OBJ. Table 10–3 provides a list of common types. The record or workspace name will be of the form *appl_object-code_REC* (see below).

Table 10–3 Procedure Types

Verb	Usage
GET	Retrieve a page form a table, or a row or table from the database into a record or workspace, or from one record or workspace into another.
INI	Initialize a record, workspace, server or other object.
RET	Return (eg. reformat, sort etc) information in a record or workspace.
UPD	Updates a row or table in the database from a record or workspace.
VAL	Validate a field, row or table held in a record or workspace.

appl[_type]_object-code_PROG

A report or other batch program. Relevant file types include 3GL file types and .LIS, .MAP, .OBJ and .OPT. Table 10–4 provides a list of common types. The record or workspace name will be of the form *appl_object-code_REC* (see below).

Table 10–4 Program Type

Verb	Usage
RCV	Receive records in a flat file from another system and insert or update them in the database tables.
SND	Fetches database tables into records in a flat file and sends them to another system.
UPD	Similar to RCV, but is normally performing updates rather than inserts.

appl[_type]_object-code_SQL

An SQLMOD module. Relevant file types include .OBJ and .SQLMOD .

This naming convention is used for SQLMOD modules which are related to a 3GL modules with a name of the form ***appl[_type]_object-code***.

appl[_type]_object-code_STR* or *appl_object-code[_type]_STR

A structure definition within a record definition. Note that all records which are to be used with DECforms should normally contain a structure immediately inside the record definition, since DECforms ignores the outer record encapsulation. In general the type usage is the same as for records. Table 10–5 provides a list of extra structure types, not typically used with record names.

Table 10–5 Structure Types

Type	Usage
DTL	Contains non key fields. Useful with DECforms to perform a 'get details' transceive.
KEY	Contains fields used to contain a tables key. Useful with DECforms to perform a 'get key' transceive.

appl[_type]_object-code_REC* or *appl_object-code[_type]_REC

A record definition. This object is a dictionary based object. Relevant file types include .CDO.

Examples include:

- *appl_MSG_VEC_REC*
A record definition for an OpenVMS message vector including the option words.
- *appl_object-code_PG[n]_REC*
A record used to hold one page of rows from table object-code. See *appl_object-code_TBL_REC* for more details. Table 10–6 provides a list of the suffices for the standard fields in a page record.
- *appl_object-code_REC*
A record used to hold one row from table object-code. Typically this record is used to allow a user to directly enter the key for a row via DECforms, and than a call is made to *appl_GET_object-code_PROC* to check that the row exists, and return the non-key columns in the row. This record is typically used in conjunction with *appl_object-code_RCV_LST* record lists.
- *appl_object-code_TBL_REC*
A record used to hold all (or at least a large number) of rows from table object-code. The *appl_GET_object-code_TBL_PROC* fetches the table from the database into this record. The *appl_GET_object-code_PG_PROC* procedure moves information from this table record (eg. *appl_object-code_TBL_REC*) into a page based record (eg. *appl_object-code_PG_REC*). Generally a table record should not be passed to DECforms. Table 10–7 provides a list of record specific fields and structures that are used.

Table 10–6 Page Record Suffices

Suffix	Usage
<code>_PG_COL_MAX</code>	A field which contains the maximum number of columns that a two dimensional <i>appl_object-code_PG_REC</i> record may hold. See <i>appl_object-code_MAX</i> for one dimensional records.
<code>_PG_COL_SIZ</code>	A field which contains the number of columns used in a two dimensional <i>appl_object-code_PG_REC</i> record. See <i>appl_object-code_PG_SIZ</i> for one dimensional records. If this number is greater than the <i>appl_object-code_PG_COL_MAX</i> field, the record has overflowed.
<code>_PG_ROW_SIZ</code>	A field which contains the maximum number of rows that a two dimensional <i>appl_object-code_PG_REC</i> record may hold. See <i>appl_object-code_COL_MAX</i> for one dimensional records.
<code>_PG_ROW_SIZ</code>	A field which contains the number of rows used in the last used column of a two dimensional <i>appl_object-code_PG_REC</i> record. See <i>appl_object-code_PG_SIZ</i> for one dimensional records. If this number is greater than the <i>appl_object-code_PG_ROW_MAX</i> field, the record has overflowed.
<code>_PG_LIM</code>	A field which contains the current maximum number of rows that the current page of a one dimensional <i>appl_object-code_PG_REC</i> may hold. For each page except the last page, this field should contain the same value as the <code>_PG_MAX</code> field. On the last page, a value less than the <code>_PG_MAX</code> value, indicates the page size is not a factor of the associated <i>appl_object-code_TBL_REC</i> table size.
<code>_PG_MAX</code>	A field which contains the maximum number of rows that a one dimensional <i>appl_object-code_PG_REC</i> record may hold. See <i>appl_object-code_COL_MAX</i> and <i>appl_object-name_ROW_MAX</i> for two dimensional records.
<code>_PG_SIZ</code>	A field which contains the number of rows used in a one dimensional <i>appl_object-code_PG_REC</i> record. See <i>appl_object-code_COL_SIZ</i> and <i>appl_object-name_ROW_SIZ</i> for two dimensional records. If this number is greater than the <i>appl_object-code_PG_MAX</i> field, the page has overflowed.
<code>_PG[_n][_qual]_STR</code>	A structure within the <i>appl_object-code_PG[_n]_REC</i> record definition. This should also be used as a group within the DECforms form. The <code>_qual</code> format is used when a record has more that one structure.

Table 10–7 Table Record Suffices

Suffix	Usage
<code>_TBL_MAX</code>	A field which contains the maximum number of rows that the <i>appl_object-code_TBL_REC</i> record may hold.
<code>_TBL_SIZ</code>	A field which contains the number of rows in the <i>appl_object-code_TBL_REC</i> record. If this number is greater than the <i>appl_object-code_TBL_MAX</i> field, the record has overflowed.
<code>_TBL[_qual]_STR</code>	A structure within the <i>appl_object-code_TBL_REC</i> record definition. The <code>_qual</code> format is used when a record has more that one structure. Note that any fields specific to a record should be included within its source.

appl_object-code_RSP

A copy text for defining a function response in a DECforms IFDL module. For each *appl_object-code_RSP* copy text there should be a corresponding *appl_object-code_KEY* copy text. This object is a text library based object. Relevant file types include .TXT.

appl_table_SIDXn

An SQL script used to create a database secondary index. Relevant file types include .SQL and .SQL_CPY .

Also, an SQL module used to access a table via a secondary index. Relevant file types include .OBJ and .SQLMOD . Typically this module would include procedures of the form *appl[_type]_object-code_OC*, *appl[_type]_object-code_FC* and *appl[_type]_object-code_CC*.

Since there can be more than one secondary index for a table, a numeric suffix is used.

Although a database doesn't distinguish between primary and secondary or alternate indices, this naming convention is suggested to clarify the index usage for developers and database administrators. Secondary indices would be used by application code to perform alternative selections and by the database to validate foreign keys.

appl[_type]_object-code_SS

An SQL module procedure to select a single a row from the table indicated by the object-code. The type variants are used when there are differing selection predicates for the table. When only a single row based on the primary key is to be retrieved, there should only be one *appl_object-code_SS* procedure.

Table 10–8, SQL Module Procedure Suffices for a Singleton Select of List of Byte Varying lists the suggested names for the SQL module procedures where a table has a *list of byte varying* column.

Table 10–8 SQL Module Procedure Suffices for a Singleton Select of List of Byte Varying

Suffix	Usage
<i>_SS_OC</i>	Open the read table cursor.
<i>_SS_FC</i>	Read the table row from the table cursor.
<i>_SS_FC_OC</i>	Open the read list cursor.
<i>_SS_FC_FC</i>	Read a segment from the list cursor.
<i>_SS_FC_CC</i>	Close the read list cursor.
<i>_SS_CC</i>	Close the read table cursor.

appl[_type]_object-code_ST_RO

An SQL module procedure to start a read only transaction. All the tables reserved in the procedure are reserved for read access.

appl[_type]_object-code_ST_RW

An SQL module procedure to start a read write transaction. At least one of the tables reserved in the procedure is reserved for write access.

appl_object-code_STP

A copy text for an ACMS task step definition. Relevant file types include .TDF_INC.

appl_object-code[_qual]_STR

A structure within the *appl_object-code_REC* record definition. This should also be used as a group within the DECforms form. The *_qual* format is used when a record has more than one structure.

appl_object-code_SUB_PNL

A copy text for using within a DECforms panel. For example *appl_STD_MAIN_BOX_SUB_PNL* draws the standard outer box, current date and program information for all the full screen panels. This object is a text library based object. Relevant file types include .TXT.

Examples include:

- *appl_STD_MAIN_BOX_SUB_PNL*
The standard main box for most full screen panels. This includes the box, the standard task or program name and version information, and the current date.

appl_table_TBL

An SQL script used to create a database table. Relevant file types include .SQL and .SQL_CPY .

Also, an SQL module used to directly access a table via its primary key. Relevant file types include .OBJ and .SQLMOD . Typically this module would include procedures of the form *appl[_type]_object-code_DR*, *appl[_type]_object-code_IR*, *appl[_type]_object-code_SS* and *appl[_type]_object-code_UR*.

appl_table_[oper_]TRGR

An SQL script used to create a database trigger. Relevant file types include .SQL and .SQL_CPY .

Where there is more than one trigger required on a table, the operation qualification is used.

Suggested values for this code are as follows:

Oper	Usage
DEL	After delete
INS	After insert
UPD	After update

or the more complete:

Oper	Usage
AD	After delete
AI	After insert
AU	After update
BD	Before delete
BI	Before insert
BU	Before update

appl[_type]_object-code_TSK

An ACMS application task definition used to manipulate an **object-code**. Relevant file types include .TDB and .TDF .

Table 10–1, Form and Task Types described with *appl[_type]_object-code_FRM*, provides a list of form and task object types. This object is a dictionary based object.

appl[_type]_object-code_UR

An SQL module procedure to update a row into the table indicated by the object-code. The type variants are used when there are differing partial updates for the table. When only full update (i.e. no non primary key columns missing and no NULLs resulting) is required, there should only be one *appl_object-code_UR* procedure.

Table 10–9, SQL Module Procedure Suffices for Updating a List of Byte Varying lists the suggested names for the SQL module procedures where a table has a *list of byte varying* column.

Table 10–9 SQL Module Procedure Suffices for Updating a List of Byte Varying

Suffix	Usage
_UR_OC	Open the update table cursor.
_UR_FC	Fetch a row from the table cursor.
_UR_UC	Update the row in the table cursor.
_UR_UC_OC	Open the insert list cursor.
_UR_UC_IC	Insert a segment into the list cursor.
_UR_UC_CC	Close the insert list cursor.
_UR_CC	Close the update table cursor.

appl_table-or-view_[qualification_]VIEW or appl_table-or-view_[qualification_]VW

An SQL script used to create a database view. Relevant file types include .SQL and .SQL_CPY .

If the view is based on a single table, it should directly reflect the table name. A qualification is required when more than one view is based on a given table.

appl_object-code_VPT

A DECforms viewport. The viewport may appear directly in a form definition or be included from the copy library. Relevant file types include .TXT (when used via a copy library).

DESCRIP.MMS

The MMS script used to build the application. Relevant file types include .MMS

MESSAGEPANEL

A field used by DECforms to display a message in the traditional message line. This is a fixed DECforms name.

Application Environment Setup

This chapter describes creating an application environment manually.

When installed at the System or Turnkey level, SysWorks is normally used to create application environments using the *Application Management Menu*. This chapter also provides a guide to the actions which SysWorks takes when performing such tasks.

11.1 Directories

To set up an application environments directory structure library manually, the following steps should be used. If SysWorks is installed at the Turnkey level or at the System level where it is used to manage the system, all these steps are executed as part of the *Add an application* or the *Create an application's environment* task.

- 1 Ensure that the SysWorks root logical names have been defined. This normally done in the site specific startup procedure `SWRK_LCL_DIR:site_PRE_STARTUP.COM`. Each root logical name should be defined using a command such as:

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE DISK_envnn device:[env_nn.] -
_$ /TRANSLATION=(CONCEALED,TERMINAL)
```

- 2 If the application common structures have not yet been defined, create them as follows:

```
$ CREATE/DIRECTORY DISK_APPLnn:[appl]
$ CREATE/DIRECTORY DISK_APPLnn:[appl.SRC]
```

- 3 Create the application environment structures as follows:

```
$ CREATE/DIRECTORY DISK_envnn:[appl]
$ CREATE/DIRECTORY DISK_envnn:[appl.DAT]
$ CREATE/DIRECTORY DISK_envnn:[appl.DOC]
$ CREATE/DIRECTORY DISK_envnn:[appl.RUN]
$ CREATE/DIRECTORY DISK_envnn:[appl.SFT]
```

- 4 If the application environment is used to build software (eg. development, maintenance and testing environments) create the construction directories as follows:

```
$ CREATE/DIRECTORY DISK_envnn:[appl.LIB]
$ CREATE/DIRECTORY DISK_envnn:[appl.WRK]
```

- 5 If the application environment is used to construct a kit (eg. testing environments) create the kit directory as follows:

```
$ CREATE/DIRECTORY DISK_envnn:[appl.KIT]
```

- 6 Create other site specific directories as appropriate.

11.2 CMS Library Creation

To set up a CMS library manually, the following steps should be used. If SysWorks is installed at the Turnkey level or at the System level where it is used to manage the system, steps 1 through 6 are executed as part of application environment setup.

- 1 Create the OpenVMS directory for the common CMS library using a command such as:

```
$ CREATE/DIRECTORY DISK_APPLn:[appl.SRC.CMSLIB]
```

- 2 Create the CMS library using a command such as:

```
$ CMS CREATE LIBRARY DISK_APPLn:[appl.SRC.CMSLIB]/NOREFERENCE -  
_ $ "App1 common CMS library"
```

- 3 Create the initial CMS classes for each environment using commands such as:

```
$ cms  
CMS> CREATE CLASS appl_DEV "Development class"  
CMS> CREATE CLASS appl_DTST "Development testing class"  
CMS> CREATE CLASS appl_MNT "Maintenance class"  
CMS> CREATE CLASS appl_MTST "Maintenance testing class"
```

- 4 Create the initial elements using a command such as:

```
$ CMS CREATE ELEMENT *.* /INPUT=DISK_DEV:[appl.WRK] "Initial version"
```

This should create a set of initial generations.

- 5 If they weren't created with the initial elements above, insert template DCL command procedures using commands such as:

```
$ CMS CREATE ELEMENT ENTER.COM_SRC -  
_ $ /INPUT=SWRK_SFT_DIR:ENTER_APPL_ENV.TEMPLATE "Base procedure"  
$ CMS CREATE ELEMENT EXIT.COM_SRC -  
_ $ /INPUT=SWRK_SFT_DIR:EXIT_APPL_ENV.TEMPLATE "Base procedure"  
$ CMS CREATE ELEMENT LOGICALS.COM_SRC -  
_ $ /INPUT=SWRK_SFT_DIR:LOGICALS_APPL_ENV.TEMPLATE "Base procedure"
```

- 6 Copy the above template files into the application development environment software directory using commands such as:

```
$ COPY SWRK_SFT_DIR:ENTER_APPL_ENV.TEMPLATE appl_SFT_DIR:ENTER.COM  
$ COPY SWRK_SFT_DIR:EXIT_APPL_ENV.TEMPLATE appl_SFT_DIR:EXIT.COM  
$ COPY SWRK_SFT_DIR:LOGICALS_APPL_ENV.TEMPLATE appl_SFT_DIR:LOGICALS.COM
```

- 7 Move to the applications development environment using a command such as:

```
$ CONTEXT APPLICATION appl DEV
```

This should result in the new application development environment's LOGICALS.COM and ENTER.COM command procedures being executed. The template LOGICALS.COM should define the *appl\CMS_PATH* logical name to have a value of *appl_DEV+*.

- 8 Insert all the initial generations into the development class using a command such as:

```
$ CMS INSERT GENERATION *.* /GENERATION=1 appl_DEV "Initial version"
```

- 9 Build the initial version using a command such as:

```
$ BUILD/BATCH/FROM_SOURCES
```

- 10 Copy the minimum required DCL command procedures into the development testing directory using a command such as:

```
$ COPY appl_SFT_DIR:ENTER.COM,EXIT.COM,LOGICALS.COM -
_$ DISK_DTST:[appl.SFT]
```

This is so that the following TRIAL will work when it uses the CONTEXT command to move to the development testing environment.

- 11 Move the initial version into development testing using a command such as:

```
$ vsnctl trial
Target environment [DTST]:
Override CMS reservations (Yes/No) [No]:
Build (Yes/No) [Yes]:
From sources or work area (Source/Work) [Work]:
Execution (Batch/Detached/Online/Subprocess) [Batch]:
Batch queue [SYS$BATCH]:
Execute after [NOW]:
```

This should cause the same generations as are in the development class to be placed into the development testing class.

- 12 Move to the applications development testing environment using a command such as:

```
$ CONTEXT ENVIRONMENT DTST
```

- 13 Release the initial version of the application using a command such as:

```
$ vsnctl release
Test environment [DTST]:
Version [V1.0]:
Execution (Batch/Detached/Online/Subprocess) [Batch]:
Batch queue [SYS$BATCH]:
Execute after [NOW]:
```

This should cause the same generations as are in the development and development testing class to be placed into the release class.

- 14 Split the release into its continued development and maintenance streams using a command such as:

```
$ vsnctl split
Version: V1.0
Development environment [DEV]:
Build (Yes/No) [Yes]:
Development testing environment [DTST]:
Maintenance environment [MNT]:
Build (Yes/No) [Yes]:
Maintenance testing environment [MTST]:
Keep existing maintenance version (Yes/No) [Yes]:
Execution (Batch/Detached/Online/Subprocess) [Batch]:
Batch queue [SYS$BATCH]:
Execute after [NOW]:
```

This should result in the release class retaining its existing generation 1 members, the development and development testing classes having generation 2 members and the maintenance and maintenance testing classes having generation 1A1 members. All the generations of an element should have the same contents and modification date. The development and maintenance

streams are now in separate branches of the generation tree so they can be merged in the future.

11.3 Granting Access for Developers

To grant access to a developer manually, the following steps should be used. If SysWorks is installed at the Turnkey level or at the System level where it is used to manage the system, all these steps are executed as part of the *Grant a user access to an application* task.

- 1 Create the OpenVMS directories for the developer using commands such as:

```
$ CREATE/DIRECTORY DISK_envvnn: [appl.RUN.username]  
$ CREATE/DIRECTORY DISK_envvnn: [appl.WRK.username]
```

A

- applMSG*, 10-1
- applRDBSHR*, 10-1
- applSHR*, 10-1
- applSHRPRV*, 10-1
- appl[_type]_object-code_CC*, 10-2
- appl[_type]_object-code_CSR*, 10-3
- appl[_type]_object-code_DR*, 10-3
- appl[_type]_object-code_FC*, 10-4
- appl[_type]_object-code_FRM*, 10-4
- appl[_type]_object-code_IR*, 10-5
- appl[_type]_object-code_OC*, 10-6
- appl[_type]_object-code_PROC*, 10-7
- appl[_type]_object-code_PROG*, 10-7
- appl[_type]_object-code_REC*, 10-8
- appl[_type]_object-code_SQL*, 10-8
- appl[_type]_object-code_SS*, 10-10
- appl[_type]_object-code_STR*, 10-8
- appl[_type]_object-code_ST_RO*, 10-10
- appl[_type]_object-code_ST_RW*, 10-10
- appl[_type]_object-code_TSK*, 10-12
- appl[_type]_object-code_UR*, 10-12
- appl_CONSTRAINTS.SQL*, 5-4
- appl_DATABASE*, 10-1
- appl_DEPENDENCIES.MMS*, 5-5
- appl_DOCUMENTATION.MMS*, 5-4
- appl_DOMAINS*, 10-2
- appl_DOMAINS.SQL*, 5-4
- appl_domain_DOM*, 10-3
- appl_DOMAIN_SDML_LIST.MMS*, 5-4
- appl_envr_APL*, 10-1
- appl_FIELDS*, 10-2
- appl_file-or-table_DOM*, 10-3
- appl_FORM_LIST.MMS*, 5-4
- appl_GRP*, 10-2
- appl_HELP.MMS*, 5-4
- appl_INDICES.SQL*, 5-4
- appl_LINK_OPT_LIST.MMS*, 5-4
- appl_MESSAGES*, 10-2
- appl_MNU*, 10-2
- appl_MSGHLP.MMS*, 5-4
- appl_object-code[_qual]_STR*, 10-11
- appl_object-code[_type]_LST*, 10-6
- appl_object-code[_type]_REC*, 10-8
- appl_object-code[_type]_STR*, 10-8
- appl_object-code_COM*, 10-3
- appl_object-code_KEY*, 10-5
- appl_object-code_RSP*, 10-9
- appl_object-code_STP*, 10-11
- appl_object-code_SUB_PNL*, 10-11
- appl_object-code_VPT*, 10-13
- appl_object-code_[PG[n]]_PNL*, 10-7
- appl_object_code_FNC*, 10-4
- appl_PROC_SRV*, 10-2
- appl_PROTECTIONS.SQL*, 5-4
- appl_ROUTINES.SQL*, 5-4
- appl_routine_FNC*, 10-4
- appl_SCHEMA.MMS*, 5-4
- appl_STORAGE_MAPS.SQL*, 5-4
- appl_table-or-view_[qualification_]VIEW*, 10-12
- appl_table-or-view_[qualification_]VW*, 10-12
- appl_TABLES.SQL*, 5-4
- appl_table_CK_n*, 10-3
- appl_table_DIDX*, 10-3
- appl_table_FK_n*, 10-4
- appl_table_PIDX*, 10-6
- appl_table_PK*, 10-7
- appl_TABLE_SDML_LIST.MMS*, 5-4
- appl_table_SIDX_n*, 10-10
- appl_table_TBL*, 10-11
- appl_table_[oper_]TRGR*, 10-11
- appl_TARGETS.MMS*, 5-4
- appl_TASK_LIST.MMS*, 5-4
- appl_TRIGGERS.SQL*, 5-4
- appl_VIEWS.SQL*, 5-5
- appl_VIEW_SDML_LIST.MMS*, 5-5

B

BUILD, 3-9, 6-1

C

.CDO, 10-2

.CDO_GEN, 10-2

CHGCTL

Menu, 4-1

CLEANUP, 6-2

CMS Libraries, 3-1

.COM, 10-3

Command procedures

ENTER.COM, 1-2, 11-2

EXIT.COM, 1-2

LOGICALS.COM, 1-2, 3-2, 11-2

Commands

CONTEXT

APPLICATION, 1-2, 2-1

ENVIRONMENT, 1-2, 2-1

VERSION, 1-2, 2-1

COMPARE, 6-1

.COM_CPY, 10-3

.COM_SRC, 10-3

CONTEXT

APPLICATION, 1-2, 2-1

ENVIRONMENT, 1-2, 2-1

VERSION, 1-2, 2-1

CREATE, 3-9

D

DBCOMPARE, 6-9

DECW\$DISPLAY, 9-20

DELETE, 3-9

DESCRIP.MMS, 9-20, 10-13

DEVELOP, 6-3

DEVTOOLS

CMS, 3-5

Directories

Architecture Independent Library, 9-26

Documentation, 9-4

Library, 9-4

Software, 9-4

Work, 9-4

DIRECTORY, 3-9

E

EDIT, 3-9

ENTER.COM, 1-2, 11-2

Environments, 1-1

EXIT.COM, 1-2

F

FETCH, 3-9

Files

appl_CONSTRAINTS.SQL, 9-26

appl_DOMAINS.SQL, 9-26

appl_INDICES.SQL, 9-26

appl_ROUTINES.SQL, 9-26

appl_SCHEMA.MMS, 9-26

appl_STORAGE_MAPS.SQL, 9-26

appl_TABLES.SQL, 9-26

appl_TRIGGERS.SQL, 9-26

appl_VIEWS.SQL, 9-26

File Types

.CDO, 10-2

.CDO_GEN, 10-2

.COM, 10-3

.COM_CPY, 10-3

.COM_SRC, 10-3

.MMS, 10-13

.OBJ, 10-6, 10-8, 10-10, 10-11

.SQL, 10-1, 10-2, 10-3, 10-4, 10-6, 10-7,
10-10, 10-11, 10-12

.SQLMOD, 10-6, 10-8, 10-10, 10-11

.SQL_CPY, 10-1, 10-3, 10-4, 10-6, 10-7,
10-10, 10-11, 10-12

.TAG_CDD, 10-2

.TDB, 10-12

.TDF, 10-12

G

Generated Scripts

appl_CONSTRAINTS.SQL, 5-4

appl_DEPENDENCIES.MMS, 5-5

appl_DOCUMENTATION.MMS, 5-4

appl_DOMAINS.SQL, 5-4

appl_DOMAIN_SDML_LIST.MMS, 5-4

appl_FORM_LIST.MMS, 5-4

appl_HELP.MMS, 5-4

appl_INDICES.SQL, 5-4

appl_LINK_OPT_LIST.MMS, 5-4

appl_MSGHLP.MMS, 5-4

appl_PROTECTIONS.SQL, 5-4

appl_ROUTINES.SQL, 5-4

appl_SCHEMA.MMS, 5-4

appl_STORAGE_MAPS.SQL, 5-4

appl_TABLES.SQL, 5-4

appl_TABLE_SDML_LIST.MMS, 5-4

appl_TARGETS.MMS, 5-4

appl_TASK_LIST.MMS, 5-4

appl_TRIGGERS.SQL, 5-4

appl_VIEWS.SQL, 5-5

Generated Scripts (cont'd)

appl_VIEW_SDML_LIST.MMS, 5-5

I

INSTALL, 6-8

L

Logical names

appl_ACMS_REPLACE, 9-6, 9-7, 9-8

appl_TDMS_REPLACE, 9-29, 9-30

DECW\$DISPLAY, 9-20

Logical Names

appl_CMS_PATH, 3-2, 3-4

appl_CMS_VARIANT, 3-2

LOGICALS.COM, 1-2, 3-2, 11-2

M

MAINTAIN, 6-3

Menus

CHGCTL, 4-1

SRCCTL, 3-7

VSNCTL, 6-1

MERGE, 6-6

MESSAGEPANEL, 10-13

.MMS, 10-13

O

.OBJ, 10-6, 10-8, 10-10, 10-11

R

RELEASE, 6-8

RENAME, 3-10

REPLACE, 3-10

REPORT, 3-10

RESERVE, 3-10

S

SHOW, 3-10

SPLIT, 6-4

.SQL, 10-1, 10-2, 10-3, 10-4, 10-6, 10-7, 10-10,
10-11, 10-12

.SQLMOD, 10-6, 10-8, 10-10, 10-11

.SQL_CPY, 10-1, 10-3, 10-4, 10-6, 10-7, 10-10,
10-11, 10-12

SRCCTL, 3-5

BUILD, 3-9

CREATE, 3-9

DELETE, 3-9

DIRECTORY, 3-9

EDIT, 3-9

FETCH, 3-9

Menu, 3-7

SRCCTL (cont'd)

RENAME, 3-10

REPLACE, 3-10

REPORT, 3-10

RESERVE, 3-10

SHOW, 3-10

UNRESERVE, 3-10

Suffices

_APL, 10-1

_CC, 10-2

_CK, 10-3

_COM, 10-3

_CSR, 10-3

_DATABASE, 10-1

_DIDX, 10-3

_DOM, 10-3

_DOMAINS, 10-2

_DR, 10-3

_FC, 10-4

_FIELDS, 10-2

_FK_n, 10-4

_FNC, 10-4

_FRM, 10-4

_GRP, 10-2

_IR, 10-5

_IR_CC, 10-5

_IR_IC, 10-5

_IR_IC_CC, 10-5

_IR_IC_IC, 10-5

_IR_IC_OC, 10-5

_IR_OC, 10-5

_KEY, 10-5

_LST, 10-6

_MESSAGES, 10-2

_MNU, 10-2

MSG, 10-1

_OC, 10-6

_PIDX, 10-6

_PK, 10-7

_PNL, 10-7

_PROC, 10-7

_PROC_SRV, 10-2

_PROG, 10-7

RDBSHR, 10-1

_REC, 10-8

_RSP, 10-9

SHR, 10-1

SHRPRV, 10-1

_SIDX_n, 10-10

_SQL, 10-8

_SS, 10-10

_SS_CC, 10-10

_SS_FC, 10-10

Suffices (cont'd)

_SS_FC_CC, 10-10
_SS_FC_FC, 10-10
_SS_FC_OC, 10-10
_SS_OC, 10-10
_STP, 10-11
_STR, 10-8, 10-11
_ST_RO, 10-10
_ST_RW, 10-10
_SUB_PNL, 10-11
_TBL, 10-11
_TRGR, 10-11
_TSK, 10-12
_UR, 10-12
_UR_CC, 10-12
_UR_FC, 10-12
_UR_OC, 10-12
_UR_UC, 10-12
_UR_UC_CC, 10-12
_UR_UC_IC, 10-12
_UR_UC_OC, 10-12
_VIEW, 10-12
_VPT, 10-13
_VW, 10-12

T

.TAG_CDD, 10-2
.TDB, 10-12
.TDF, 10-12
TRIAL, 6-7

U

UNRESERVE, 3-10

V

Version Control, 4-1, 6-1

VSNCTL

BUILD, 6-1
CLEANUP, 6-2
COMPARE, 6-1
DBCOMPARE, 6-9
DEVELOP, 6-3
INSTALL, 6-8
MAINTAIN, 6-3
Menu, 6-1
MERGE, 6-6
RELEASE, 6-8
SPLIT, 6-4
TRIAL, 6-7