Technical Requirements Guide (TRG) - Reference

DOCUMENT IDENTIFIER: A-MN-ELCP790-00-0000 Rev B, 10-May-1993

ABSTRACT: This document describes the technical requirements and recommendations for developing Digital layered products.

APPLICABILITY: This reference guide is intended for software engineers, project leaders, project managers, and technical writers.

STATUS: APPROVED 10-May-1993; use VTX SMC for current status.

Technical Requirements Guide (TRG) - Reference

DOCUMENT IDENTIFIER: A-MN-ELCP790-00-0000 Rev B, 10-May-1993

REVISION HISTORY:	Rev A	11-Jun-1992	
	Rev B	10-May-1993	ECO Number NRO01

Document Management Category:	Software Manuals (TDS)
Responsible Department:	Cross-Engineering Services/Software Quality Assurance (CES/SQA)
Responsible Person:	Fred Howell

APPROVAL: This document has been reviewed and approved by the Manager of the Cross-Engineering Services/Software Quality Assurance (CES/SQA) group.

Fred Howell - Cross-Engineering Services/Software Quality Assurance (CES/SQA)

Direct requests for further information to:

GSFSYS::SQA

Use VTX SMC to order copies of this document from Standards and Methods Control. Send distribution questions to JOKUR::SMC or call DTN: 234-4423.

Preface

The Technical Requirements Guide (TRG) - Reference describes the technical requirements and recommendations for developing Digital layered products.

Intended Audience

This guide is intended for software engineers, project leaders, product managers, and technical writers. This guide is useful for anyone working with the Software Quality Assurance (SQA) group for assessing product readiness for release. See Software Quality Assurance (SQA) for more information on the SQA group.

Document Structure

The previous version of the Technical Requirements Guide contained a checklist and reference guide. The new version of this guide has been reformatted and the information is now available in the following documents:

Reference Guide - Contains detailed descriptions of technical requirements.
PC Checklist - Describes technical requirements for PC products.
OpenVMS Checklist - Describes technical requirements for OpenVMS products.
UNIX Checklist - Describes technical requirements for UNIX products.

How to use the TRG documents:

1. Obtain a copy of this requirements guide by using the pointers that follow and print this document out in the postscript format.
2. Choose the checklist you need your for product. Obtain a copy of that checklist by using the pointers that follow.
3. Use the checklist online by completing the "Answer" column (the last column on the right) for each requirement using the SYNTAX GUIDELINES listed in subhead 1.2 of the checklist.
4. Read the first column in the checklist, Requirement description, which is a brief one-sentence or two-sentence requirement statement.
5. If you have any questions about the requirements while working with the checklist, refer to the center column in the checklist, Refer to requirement.

6. Make a note of the requirement identifier. For example: prod_name_v
7. Find this requirement identifier in the requirements guide obtained in step 1 for a complete description of the requirement.

How to Obtain TRG Documents

You may copy the TRG documents from the following locations:

JOKUR::TRG:EL-CP790-00_REFERENCE_REVB.PS
JOKUR::TRG:EL-CP790-01_PC_CHECKLIST_REVA.TXT
JOKUR::TRG:EL-CP790-02_OPENVMS_CHECKLIST_REVA.TXT
JOKUR::TRG:EL-CP790-03_UNIX_CHECKLIST_REVA.TXT

The TRG Bookreader files contain all four documents, EL-CP790-00, EL-CP790-01, EL-CP790-02, and EL-CP790-03.

JOKUR::TRG:EL-CP790_TRG.DECW$BOOK
JOKUR::TRG:EL-CP790_TRG.DECW$BOOKSHELF

Use VTX SMC to order copies of any of the following documents.

EL-CP790-00, Technical Requirements Guide (TRG) - Reference
EL-CP790-01, Technical Requirements Guide (TRG) - PC Checklist
EL-CP790-02, Technical Requirements Guide (TRG) - OpenVMS Checklist
EL-CP790-03, Technical Requirements Guide (TRG) - UNIX Checklist

Appendix A — This appendix provides additional information about tools available to assist layered product groups in meeting test and registration requirements.

List of Abbreviations and Acronyms — This list provides definitions of many abbreviations and acronyms used in the TRG guide and checklists. Abbreviations and acronyms that are specific to a product or operating system are not included since it is not the intention of the TRG documents to serve as product documentation. Refer to the product documentation if you need an abbreviation or acronym definition not provided here.

Associated Documents

Many requirements have references to additional sources of information. Refer to individual requirement descriptions for names and locations of associated documents.

Software Quality Assurance (SQA)

SQA's Role

Software Quality Assurance (SQA) is responsible for assessing a product's overall readiness for release to the Software Supply Business (SSB).

History

Until January 1992, this readiness assessment was carried out by the Software Quality Operations and Information Services (SQOIS) group. In July 1992, the scope of SQA's responsibilities was enlarged to include the definition and maintenance of standards and technical requirements.

DEC STD 038-1

The Technical Requirements Guide (TRG) - Reference is the guide for implementing *DEC STD 038-1 Systems Evaluation of New Products - Software*. Meeting the requirements in this guide for your product is equivalent to being in compliance with the requirements of *DEC STD 038-1*.

SQA Responsibilities

The purpose of SQA is to be an unbiased party in performing the auditing function to determine that products satisfy Corporate requirements and minimum ship criteria (MSC).

SQA will provide advice and consultation on how to improve products and processes, including, but not limited to, Quality/Test Plans, the SQA Process, *DEC STD 038-1 Systems Evaluation of New Products - Software*, the Technical Requirements Guide (TRG) document set (EL-CP790-00 through -03), Corporate Minimum Shipment Criteria (MSC), and Total Quality Management (TQM) Initiatives.

SQA also manages the technical content and implementation of the TRG, *DEC STD 204-0 Software Correction and Distribution*, and DEC STD 038-1.

Communicating with SQA

Any questions and requests for assistance related to product release issues can be mailed to GSFSYS::SQA.

The following documents can be obtained from VTX SMC:

EL-EN797-00, *Software Quality Assurance Process Guide for Products*

EL-00197-00, *DEC STD 197-0 Legal Requirements and Guidelines for Digital Publications and Software*

Contents

1 INTRODUCTION

EL-CP790-01, *Technical Requirements Guide (TRG) - PC Checklist*, EL-CP790-02, *Technical Requirements Guide (TRG) - OpenVMS Checklist*, and EL-CP790-03, *Technical Requirements Guide (TRG) - UNIX Checklist* each present descriptions of the technical requirements that apply to that particular checklist. Detailed descriptions of the requirements are listed alphabetically in EL-CP790-00, *Technical Requirements Guide (TRG) - Reference*.

Refer to **sample_requirement_name** (page 4) for an explanation of the types of information found in each section of a requirement description.

1.1 POINTER TO THE TECHNICAL REQUIREMENT DOCUMENTS

The previous version of the Technical Requirements Guide contained a checklist and reference guide. The new reversion of this guide has been reformatted and the information is now available in the following documents:

Reference Guide -	Contains detailed descriptions of technical requirements.
PC Checklist -	Describes technical requirements for PC products.
OpenVMS Checklist -	Describes technical requirements for OpenVMS products.
UNIX Checklist -	Describes technical requirements for UNIX products.

You may copy the TRG documents from the following location:

JOKUR::TRG:EL-CP790-00_REFERENCE_REVB.PS

JOKUR::TRG:EL-CP790-01_PC_CHECKLIST_REVA.TXT

JOKUR::TRG:EL-CP790-02_OPENVMS_CHECKLIST_REVA.TXT

JOKUR::TRG:EL-CP790-03_UNIX_CHECKLIST_REVA.TXT

The TRG Bookreader files contain all four documents,
EL-CP790-00, EL-CP790-01, EL-CP790-02, and EL-CP790-03.

JOKUR::TRG:EL-CP790_TRG.DECW$BOOK

JOKUR::TRG:EL-CP790_TRG.DECW$BOOKSHELF

Use VTX SMC to order copies of any of the following TRG documents:

EL-CP790-00	*Technical Requirements Guide (TRG) - Reference*
EL-CP790-01	*Technical Requirements Guide (TRG) - PC Checklist*
EL-CP790-02	*Technical Requirements Guide (TRG) - OpenVMS Checklist*
EL-CP790-03	*Technical Requirements Guide (TRG) - UNIX Checklist*

2 HOW TO INTERPRET REQUIREMENT DESCRIPTIONS

Requirement descriptions follow a standard format. The following sample requirement is briefly annotated to explain what type of information is contained in each section. For a detailed explanation of each numbered item in the sample requirement, refer to the list immediately following the sample requirement.

sample_requirement_name

1

The requirement statement is printed in this space. 2

OPERATING SYSTEMS (ALL ARCHITECTURES):
3

	OpenVMS ULTRIX Sun OSF/1
PERFORMED BY: 4	Technical Writer
REQUIREMENTS AID(S): 5	None

OpenVMS

6

Description

7 A full description of the requirement is placed here.

Rationale

8 The reason for the requirement is placed here.

Exceptions

9 Any exceptions to the requirement are noted here.

Example

10 Examples of the requirement are placed here.

For more information

11 Pointers to additional information about the requirement are placed here.

Description of Sample Requirement

1 This is the Unique Requirement Identifier (URI) that identifies each requirement by name. URIs are usually not made up of complete words; they are a shorthand way of referring to a requirement.

2 This is the requirement statement, which briefly states what you need to comply with.

3 This identifies the operating systems for which the requirement applies. If your product is not being developed on any of the operating environments listed in this section, then the requirement does not apply to you and you can skip it. On the other hand, if your product is being developed on one or more of the operating environments listed here, then you need to comply with it.

4 This identifies the person in a development team usually concern with the requirement. Team-member functions identified in this guide are: product manager, project leader, software engineer, and technical writer. Every development team is different; in your group it could be a person different from the one listed here.

5 This identifies tools that can be used in conjunction with the requirement. Currently only the REQ CHECK tool is referenced here for certain ULTRIX requirements. When no tools are available, the word "None" is printed. For every tool identified, a short description is available in Appendix A.

6 This is an operating system banner. It identifies to which operating system the description applies. Similar banners exist for ULTRIX, OSF, and Sun. When a multiplatform requirement provides a description without showing any banners, the description provided applies to all operating systems listed.

7 This is the description section, which provides more details about the requirement.

8 This section explains why complying with the requirement is needed or recommended.

9 This section calls out any exceptions to complying with the requirement. If no exceptions exist, the word "None" is printed.

10 When available, an example illustrating the requirement is shown here. Sometimes small examples are embedded in the description rather than here. If no examples are available, the words "Not available" are printed. If the requirement cannot be illustrated, the words "Not applicable" are printed.

11 When available, this section provides you with pointers to more sources of information or assistance. When no pointers are available, the word "None" is printed.

acct_quota

The preinstallation section lists all required account quotas, including instructions for checking and changing these values.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

The preinstallation section must list all nondefault account quotas necessary for the installation. Instructions for checking and changing these values via the AUTHORIZE utility must be included.

Rationale

Complying with this requirement provides the user with the information needed to make all necessary changes before the installation. In general, it ensures a successful installation.

Exceptions

None

Example

The following example shows how to check for the current value of the account quota for user Schwartz:

```
$ set def sys$system
$ run authorize
AUTH>show Schwartz
```

The following example shows how to set the required BIOLM value of 30 for the Schwartz account.

```
$ set def sys$system
$ run authorize
AUTH>modify Schwartz/biolm=30
```

For more information

For information on installation guide templates, refer to requirement ig_templates .

added_files_log_names

An appendix of the installation guide lists all the files and OpenVMS logical names that have been added or modified by the installation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OSF/1	MS–DOS	Sun
ULTRIX	Windows NT	Windows
Mac	SCO UNIX	OpenVMS
AIX	HP-UX	

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

An appendix of the installation guide must contain a list of the files and logical names that have been added or modified by the installation. It should also indicate what special protections or acls have been set on files and directories.

If you have a localized product, you should also include a list of the files and logical names of the product variant that were added or modified by the installation.

Rationale

Enables the installers to know what logical names and files have been added to the system and where installers reside, in case the installers have a need to remove the product or encounter a problem with a particular file. It also assists in housekeeping and product maintenance. This information can also be used to repair a product in the event of accidental deletion or alteration.

Exceptions

None

Example

Not applicable

For more information

For information on installation guide templates, refer to requirement ig_templates .

alpha_com_opt

Software and/or documentation that implement, reference, or describe command language options and qualifers for the Alpha AXP architecture must be "Alpha_ AXP".

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Software and/or documentation that implement, reference, or describe command language options and qualifers for the Alpha AXP architecture must be "Alpha_ AXP".

Rationale

Using Alpha_AXP will provide a consistent and legal method to reference or describe command language options and qualifers for the Alpha AXP architecture.

Exceptions

None

Example

None

For more information None

backup_sys_disk

The preinstallation section of the installation guide should recommend a backup of the disk or file system on which the product will be installed before starting the installation procedure.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OSF/1	ULTRIX	Sun
AIX	SCO UNIX	OpenVMS
HP-UX		

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

The installation guide should recommend that the installer perform a backup of the system disk before installing the layered product.

Rationale

This protects the users' environment in case problems that cause the system to be corrupted occur during the installation.

Exceptions

None

Example

OpenVMS

Backing Up your System Disk

At the beginning of the installation, VMSINSTAL asks whether you have backed up your system disk. Digital recommends backing up your system disk prior to installing any software.

Use the backup procedures that have been established at your site. For details on backing up your disk, see the section on the Backup Utility in the OpenVMS documentation set.

UNIX-based systems

Performing System Backup

It is recommended that a system backup be performed before installing DEC C++. Refer to the UNIX System Management reference manual (*Guide to Backup and Restore - ULTRIX*).

For more information

For information on using installation guide templates, see requirement ig_templates

OPENVMS ONLY

Refer to the *Backup Utility Manual* in the OpenVMS Documentation set for detailed information.

bit_byte_interpret

The product interprets the server's bit or byte order correctly when placing and removing images on a server.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX		

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Alpha AXP, VAX, and RISC systems from Digital expect the least significant bit and the least significant byte first (otherwise known as "Little Endian"). Many other third-party machines expect some different combination, such as most significant bit and byte first ("Big Endian"). A Sun workstation is a good example of this different bit/byte ordering scheme.

Client applications that display to a server using different bit and byte order must be able to interpret the server's bit or byte order when requesting image displays.

The actual request for display is interpreted by the server but the data to be displayed is not and, therefore, it must be ordered correctly by the application.

Rationale

All products using DECWindows and DECwindows/Motif servers must fully interoperate with all servers supported by Network Application Services (NAS). For an image to be properly displayed, it is crucial that a product correctly interpret the server's bit/byte order.

Exceptions

None

Example

Not applicable

For more information

OPENVMS ONLY

Refer to *OpenVMS DECwindows Motif Guide to Application Programming* for more information on bit/byte order.

callback_acct_create

Products that create or modify accounts should use the CREATE_ACCOUNT or UPDATE_ACCOUNT callbacks.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

When layered products create or modify accounts, they must use the CREATE_ACCOUNT or UPDATE_ACCOUNT callbacks. Rights identifiers corresponding to each account created are added to the rights data base unless the /NOADD_IDENTIFIER qualifier on CREATE_ACCOUNT is used.

The OpenVMS operating system provides a file called a rights data base that contains a list of special names called identifiers, as well, as a list of the users specified as holders of the identifiers. There are three types of identifiers: UIC, general, and system-based. This list of rights identifiers will deny or grant access to the holders to objects on the system.

Rationale

The main benefit is to provide a VMSINSTAL interface to the AUTHORIZE utility and to provide a consistent output display to the installers. Callbacks support all valid OpenVMS configurations and manage synchronization of actions, as some VMSINSTAL activities are deferred and do not take immediate effect.

Exceptions

None

Example

```
VMI$CALLBACK_CREATE_ACCOUNT SMITH"/PASSWORD=FCDREG/UIC=[360,103]"
VMI$CALLBACK_UPDATE_ACCOUNT SMITH"/PASSWORD=FCDAUX"
```

For more information

If You Need	Do the Following
Information on callbacks	Refer to *Developers Guide to VMSINSTAL*.
Information on rights identifiers	Refer to the *OpenVMS System Manager's Manual*.
Information on installation guide templates	Refer to requirement ig_templates

callback_use

Layered product groups should use callbacks that are most appropriate for the situation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): None

Description

It is recommended that you use the most appropriate VMSINSTAL callbacks for product installation. The following is a list of callbacks commonly used during installations:

ASK
CHECK_LICENSE
CHECK_NET_UTILIZATION
CHECK_PRODUCT_VERSION
CHECK_VMS_VERSION
CONFIRM_LICENSE
CONTROL_Y
CREATE_ACCOUNT
CREATE_DIRECTORY
DELETE_FILE
FIND_FILE
GET_IMAGE_ID
GET_PASSWORD
GET_SYSTEM_PARAMETER
MESSAGE
PROVIDE_DCL_COMMAND
PROVIDE_DCL_HELP
PROVIDE_FILE
PROVIDE_IMAGE
RESTORE_SAVESET
RUN_IMAGE
SET IVP
SET PURGE
SET SAFETY
SET STARTUP
UPDATE_ACCOUNT
UPDATE_FILE

CONFIRM_ LICENSE callback

This callback should be included in the installation procedure to inquire whether a license has been registered. The intent is to protect Digital's intellectual investment and to force end users to take overt actions to breach the terms of their contract with Digital.

SET IVP ASK callback

The installation should use the SET IVP ASK callback to ask the users if they want to execute the Installation Verification Procedure (IVP) immediately after the installation. If the answer from the CONFIRM_LICENSE callback is negative, the installation should turn off the execution of the IVP via the SET IVP NO callback, even if the answer to the SET IVP ASK callback is yes.

The SET IVP ASK callback discourages the automatic execution of the IVP upon completion of the installation. Users may want to execute the IVP later, rather than immediately after the installation.

OpenVMS (Alpha AXP)

The following callbacks are obsolete for OpenVMS (Alpha AXP):

- SET REBOOT (Use SET SHUTDOWN instead)
- PATCH_IMAGE

The following are new callbacks for OpenVMS (Alpha AXP):

- SET FILE
- SET PRODUCT_NAME
- SET SEMANTICS

Rationale

To promote the consistent user interfaces to installation procedures, and to promote consistent installation procedure behavior. The callbacks also provide the installation procedure writer with a supported interface to the OpenVMS utilities, thus preventing installation procedures breaking due to the changes in the OpenVMS utilities, such as Licensing Management Facility (LMF), SYSGEN, and AUTHORIZE.

Exceptions

None

Example

Not applicable

For more information

Refer to the *Developers Guide to VMSINSTAL* guide for more information on the above callbacks.

case_scp

The asterisk character (*) cannot be used as a value for the ACT variable in the .scp file.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OSF/1	ULTRIX	Sun
PERFORMED BY:	Software Engineer		
REQUIREMENTS AID(S):	None		

Description

The setld group requires that the asterisk character (*) does not occur as a value for the ACT variable in the product's .scp file.

Rationale

The setld group would like to reserve the right to define new values for the ACT variable, if necessary. A product using the asterisk may fail during installation because improper functions for new values of ACT are executed.

Exceptions

None

Examples

1.
```
*) exit 0 ;;
```

   In this example, the .scp incorrectly uses an asterisk (*) as the value for ACT and instructs the setld to exit.

For more information

If You Need	Do the Following
Information on using setld	Refer to any of the following: • *ULTRIX Software Development*, Volume 2, *Guide to Preparing Software for Distribution on ULTRIX Systems* • VIRTUE::SETLD-ULTRIX-LPS VAX Notes conference

cause_of_err_msg

When an installation error occurs, a message is displayed on the screen stating the cause of the error and the recommended fix.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Technical Writer or Software Engineer

REQUIREMENTS AID(S): None

Description

The installation should output a brief description of the cause of the error and the recommended fix, and then it should point the user to the installation guide for a more detailed explanation of the problem.

Rationale

To promote consistent, uniform, and user-friendly installations.

Exceptions

None

Example

None

Note for OpenVMS

For an OpenVMS installation, the error message should be output using the MESSAGE callback, and it should specify the appropriate severity level code. This is useful to search for -S- and -E- in a long installation log.

For more information

None

cdrom

If the product is distributed on CD-ROM, all software and documentation meet the distribution requirements and are forwarded to the proper Software Quality Operations and Information Services (SQOIS) accounts.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX		

PERFORMED BY: Technical Writer or Software Engineer

REQUIREMENTS AID(S): None

Description

EIID Engineering is responsible for releasing products on CD-ROM. Product managers and project leaders receive a monthly announcement listing the products planned for release on CD-ROM the next month. The announcement also indicates the submission date by which all software and documentation files need to be in the possession of EIID.

Product Release Criteria

To release your product successfully on CD-ROM, the product should meet the following criteria:

- The product must be complete; therefore, all of the following must be in place.
  - The media kit is available.
  - SPD/SSA is in PostScript and ASCII formats.
  - The installation guide is in PostScript and ASCII formats.
  - The cover letter, read-me-first, and before-you-install documents are in PostScript and ASCII formats.
  - The release notes are in PostScript and ASCII formats (applies to ULTRIX products only).
- The Software Supply Business (SSB) kit must reside in Central Software Engineering Release Library (CSERL). Since worldwide consolidations originate from this library, the media kit in CSERL must contain the *same* binaries as the traditional media shipping from the SSB.

- The product is not retired.
- The product does not have any ITAR restrictions (that is, encryption).
- The product does not require a paper Component Product Authorization Key (CPAK). Electronic CPAKs are allowed.
- The product planner has declared submission at the SSB on time for the applicable release date.

Documentation Requirements

The following documentation criteria should be observed.

- The installation guide(s), cover letter(s), and release notes in PostScript and ASCII formats must be submitted to the local SQOIS group. SQOIS will retrieve the SPD/SSA from Software Product Description (SPD) Administration.
- The following accounts have been established at the local EIID sites to assist in documentation retrieval.

EIID Engineering US	VIRTUE::CONDIST_DOCS
EIID Engineering Japan	SQMJPN::SQM
EIID Engineering UK	SQGUK::SQM

- Mail is to be sent to the appropriate account with the following information.
  - Product name and version
  - Complete file specification of the documents (include node and device)
  - Type of document (installation guide, cover letter, release notes)

EIID will copy the respective documents and ensure they are included with the product. To accomplish this, the directories and files must have WORLD:READ file protection.

Rationale

To ensure smooth product release on CD-ROM.

Exceptions

Products not releasing on CD-ROM.

Example

Not applicable

For more information

For information on CD-ROM release dates, location of SQOIS accounts for software, and general assistance, send mail to FRMENG::CDROM.

cleanup_instr

The postinstallation section of the installation guide contains any special cleanup instructions.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX		

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

The postinstallation section of the installation guide should include any information about special system arrangements and cleanup procedures that can be performed after the installation.

The guide should also give instructions on how to clean up after an installation has aborted or failed. It should included instructons on what files, accounts, directories, and so on must be removed before another installation can be attempted.

Rationale

Special cleanup instructions ensure that the installer has the correct information to configure the system correctly so that the product runs as expected. Also, removing unwanted files after a failed or completed installation keeps the system tidy and assists the system manager in managing the system more effectively.

Exceptions

None

Example

Not available

For more information None

color

The product compensates for unavailable color map entries.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

There are multiple versions of color implementation on various servers. Your application needs to function correctly with any of the following color types:

- True Color
- Pseudo Color
- Static Color
- Grey Scale

Each color type may be supported on a variety of hardware, depending on how the hardware has been configured.

Since color values may be changed dynamically, windowing applications need to compensate for unavailable colormap entries.

Rationale

If a product is to display correctly on multiple color servers, all types of color must be considered.

Exceptions

None

Example

Not applicable

For more information

If You Need	Do the Following
More information on color types	Refer to *X Window System*, Part I, Section 3.1 "Visual Types"
More information on X calls to get information about a server screen	Refer to *X Window System*, Part I, Section 2.2 "Obtaining Information About the Display" or Section 10.8, "Determining the Appropriate Visual Type"
More information on colormaps	Refer to *X Window System*, Section 5.1, "Colormap Functions"

content

The release notes include problem fixes, problems solved, known problems and/or restrictions, and documentation updates. They are not used in place of complete user documentation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

Release notes should cover the following areas, if applicable:

- Problem fixes or problems resolved by the product

  You can make problem fixes until the beginning of the 30-day code freeze required by the Corporate minimum ship criteria. If necessary, you can list all the resolved problems in the Release Notes, as illustrated in the following example.

  ```
  When HELP was accessed from the editor, the cursor was forced to the beginning
  of the current line after returning from HELP. The cursor is now restored to
  its original position upon returning from HELP.
  ```

- Known problems or restrictions

  Known problems are not something one wants advertised, but knowledge of them may be useful to a customer, especially if a workaround can be included. You can list in the Release Notes any known problems that are not resolved when master kits are created. The following example shows a listing of a resolved problem:

  ```
  PROBLEM:    There are occasionally problems with refreshing windows.

  WORKAROUND: There are 3 ways to refresh a window:

  1.  Press Ctrl/W while the pointer is positioned in the active window

  2.  Resize the window

  3.  Shrink the window to an icon, then click on the icon to reopen the window
  ```

- Documentation changes and omissions

  Documentation errors found after the books have gone to print can be documented in the Release Notes, for example:

  ```
  Section 1.2.2 - The second sentence reads "Line numbers are not
  implemented."  It should read "Line numbers are not incremented."
  ```

- Identify changes from the last release of the product

  You can identify changes from the last release of the product. A mature product, in particular, can contain only modifications made to maintain compatibility with the base operating system or to incorporate customer suggestions. The following example shows how you might describe a change from the last release.

  ```
  The SHOW KIT command has been replaced by the SHOW SAVESET command.
  ```

Rationale

Identifying known problems can reduce the number of Quality Assurance Reports (QARs) and Software Performance Reports (SPRs) from customers.

Exceptions

None

Example

The following example shows a listing of such problems:

```
RESTRICTION: A document with a print file specification of
more than 34 characters will not print. It appears to be queued
properly, but does not print.

WORKAROUND: Enter a print file specification of 34 characters or
less, or generate only the PostScript file and print it without
using DECwrite. The name of the PostScript file when printed in
this manner is not limited to 34 characters.
```

For more information None

copyright_in_ivp

The Installation Verification Procedure (IVP) displays the copyright notice before beginning execution.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX		

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

If the IVP invokes the layered product in a nonwindows environment, the entire copyright notice needs to be displayed. If the IVP invokes the layered product in a windows environment, only the first line of the copyright notice must be displayed.

Rationale

Display of the copyright notice when the IVP is invoked protects Digital's legal rights because the notice informs the user that the product is a licensed Digital product protected by copyright laws.

Exceptions

None

Example

Not applicable

For more information

Refer to *DEC STD 197-0 Legal Requirements and Guidelines for Digital Publications and Software* for further information on display of copyrights.

cut_paste

The product's cut and paste functions work among various applications without any data loss.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX		

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

The windowing applications must demonstrate that all their supported data types, such as text and DDIF, correctly execute cut and paste functions without any data loss among various applications in a distributed, heterogeneous environment.

Rationale

This is basic functionality expected by customers.

Exceptions

None

Example

An example demonstrating this functionality would be to cut text from OpenVMS DECwrite displayed to an ULTRIX server and then paste the text into a DECpresent for ULTRIX window displayed on the same server.

For more information None

dcl_full_com

When Digital Command Language (DCL) commands are used, they must be spelled in full.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): None

Description

DCL commands used in any installation command procedure should be spelled out completely. Abbreviations are not allowed.

Obsolete DCL commands For OpenVMS (Alpha AXP)

The following DCL commands are obsolete for OpenVMS on Alpha AXP:

- MONITOR POOL
- SET FILE/UNLOCK
- UNLOCK

Rationale

VMSINSTAL redefines some DCL verbs; hence, they need to be spelled out completely. Failing to do so could create problems and cause the installation to fail.

Also, abbreviated commands will be interpreted by OpenVMS at the first alphabetical level of uniqueness. This can result in the command not being interpreted as intended or expected. Conflicts with customer-defined commands are also likely to occur.

Exceptions

None

Example

Use TYPE instead of TY

Use SHOW instead of SHO

For more information None

dcl_syntax

New Digital Command Language (DCL) syntax has been approved by the DCL Review Board.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

The DCL Review Board consists of representatives from various software engineering groups. The DCL Review Board must approve all DCL syntax (top-level verbs, keywords, and qualifiers). Refer to the help icon at the end of this requirement description for a pointer to guidelines on DCL syntax.

It takes between six and eight weeks for the DCL Review Board to review a proposed syntax. Therefore, be sure to submit your syntax early enough in the development cycle to allow for any code or documentation changes. The information you must submit is as follows:

- The official product name as shown on the Software Product Description (SPD) and a version number.
- A one-line description of the product to aid the DCL Review Board in determining if there is a similar product with similar syntax.
- A summary of the syntax and a brief description of the features, including:
  - all DCL and subcommand-level commands
  - qualifiers
  - arguments

After reviewing this information, the Board may make recommendations for changes, after which it is your responsibility to implement the approved syntax, and repost for further review.

You should be able to provide justification to the Systems Integration Manager if you did not receive syntax approval from the DCL Review Board.

Rationale

Maintains consistent and correct use of DCL syntax so that command verbs, parameters, and qualifiers are indicative of their intended function and consistent across products and platforms on which DCL is supported. This makes the product more intuitive and consistent.

The Software Quality Assurance Group may hold up product release if the product uses unapproved syntax. Refer to the For More Information section at the end of this description for a pointer to the Software Quality Assurance Group.

Exceptions

None

Example

Not applicable

For more information

If You Need	Do the Following
Current DCL guidelines and a list of current DCL Review Board Members	Refer to the VIRTUE::DCLREVIEW Notes conference
To contact the Software Quality Assurance Group.	Send mail to GSFSYS::SQA

dcl_tables_instl

DCLTABLES may be replaced through the INSTALL SYS$SHARE:DCLTABLES /REPLACE command on other nodes in a cluster. The SYSMAN utility can execute this command on any node without login.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

DCLTABLES should be replaced via the INSTALL SYS$SHARE:DCLTABLES /REPLACE command on other nodes in a cluster. The SYSMAN utility can execute this command on any node without login. Starting the product on other nodes in a cluster is also done via the SYSMAN utility.

Rationale

To allow access to newly installed products on all nodes of the cluster.

Exceptions

None

Example

Not applicable

For more information None

ddif_tag_in_instl

All .DDIF files have the .DDIF file extension restored in the event the semantic tag DDIF was stripped during the SPKITBLD procedure.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Currently, during the SPKITBLD procedure, .DDIF files lose the RMS semantic tag .DDIF because of the /INTERCHANGE qualifier. During the subsequent installation, files are placed without this required tag.

To rectify this situation, use the SET FILE/SEMANTICS=semantics-tag in the KITINSTAL.COM *after* restoring the saveset containing .DDIF file(s) and *before* moving them to the destination via the PROVIDE_FILE callback.

Rationale

To ensure a successful product installation.

Exceptions

To avoid installation failure.

Example

```
$ SET FILE /semantics=DDIF VMI$KWD: file_name.ddif
```

For more information None

decw_user_interface

If the product supports the DECwindows user interface, recommendations on installation, file destinations, and naming conventions have been considered.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK for OpenVMS

Description

Support Before OpenVMS V5.1

There is no DECwindows support for OpenVMS versions prior to V5.1. Layered products should not assume the existence of any DECwindows directories or files.

License Checking

License checking of DECwindows is not required.

File Naming

File prefixes such as DECW$, CDA$, and DDIF$ are reserved by OpenVMS.

DECwindows Installation

For products providing both a DECwindows and non-DECwindows component, it is recommended that the installation prompt the installer to determine if the DECwindows component of the product should be installed. The installation procedure should verify the existence of the components listed in the following table:

Description	DECwindows Filename
For any DECwindows component	SYS$SHARE:DECW$XLIBSHR.EXE
Compute Server Support component	SYS$SYSTEM:DECW$SESSION.EXE
Device Support component	SYS$SYSTEM:DECW$SERVER_MAIN.EXE
Programming Support component	SYS$SYSTEM:DECW$UILCOMPILER.EXE

Depending on the DECwindows functionality required by the installing product, prompting for input on the installation of the DECwindows component is necessary because the addition of the DECWindows component usually uses additional disk space.

File prefixes, such as DECW$ and CDA$, are reserved by OpenVMS and should not be used by layered products. This will insulate products from upgrades to OpenVMS where all DECwindows files are purged and deleted.

File destinations

All DECwindows files should be targeted to a .USER* directory to avoid having subsequent versions of OpenVMS DECwindows wipe out the contents of SYSTEM or other non-SYSTEM directories reserved for OpenVMS. The following table lists the directories the product development groups should pay attention to.

DECwindows Directories	Purpose
SYS$COMMON:[VUE$LIBRARY.USER]	This directory holds VUE command procedures for layered products. The parallel directory SYS$COMMON:[VUE$LIBRARY[.SYSTEM] is reserved by OpenVMS and is not available to layered products.
SYS$COMMON:[DECW$DEFAULTS.USER]	This directory holds User Interface Definition (UID) and Xresourse (Xrm) files for layered products. The parallel directory SYS$COMMON:[DECW$DEFAULTS[.SYSTEM] is reserved by OpenVMS and should not be used by layered products. The logical name "DECW$SYSTEM_DEFAULTS" should be used to locate these files, since this is a search list through the .USER and .SYSTEM subdirectories. Layered products should not put any UID or Xresource files in SYS$LIBRARY.
SYS$COMMON:[SYS$KEYMAP.DECW.USER]	This directory holds keymap files for layered products. The parallel directory SYS$COMMON:[SYS$KEYMAP.DECW[.SYSTEM] is reserved by OpenVMS and is not available to layered products.
SYS$COMMON:[SYSFONT.DECW.USER_75DPI]	This directory holds 75 dots per inch fonts for layered products. The parallel directory SYS$COMMON:[SYSFONT.DECW[.75DPI] is reserved by OpenVMS and is not available to layered products.
SYS$COMMON:[SYSFONT.DECW.USER_100DPI]	This directory holds 100 dots per inch fonts for layered products. The parallel directory SYS$COMMON:[SYSFONT.DECW[.100DPI] is reserved by OpenVMS and is not available to layered products.

DECwindows Directories	Purpose
SYS$COMMON:[SYSFONT.DECW.USER_CURSOR16]	This directory holds 16 bit x 16 bit cursors for layered products. The parallel directory SYS$COMMON:[SYSFONT.DECW.CURSOR16] is reserved by OpenVMS and is not available to layered products.
SYS$COMMON:[SYSFONT.DECW.USER_CURSOR32]	This directory holds 32 bit x 32 bit cursors for layered products. The parallel directory SYS$COMMON:[SYSFONT.DECW.CURSOR32] is reserved by OpenVMS and is not available to layered products.
SYS$COMMON:[DECW$INCLUDE]	This directory is reserved by OpenVMS DECwindows and is not available to layered products.
SYS$COMMON:[SYSHLP.EXAMPLES.DECW]	This directory is reserved by OpenVMS DECwindows and is not available to layered products. Current SQOIS guidelines recommend that layered products create their own subdirectories to [SYSHLP.EXAMPLES] and do not use the DECW$EXAMPLES directory.

Rationale

To avoid conflicts and problems during product installations.

Exceptions

None

Example

Not applicable

For more information None

default_answers

Where possible, all questions provide default answers.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

Default answers are usually the recommended answers for the questions asked during the installation.

Rationale

Providing default answers prompts the user with the recommended responses so that they can quickly move through an installation. Default answers also assist users who may be uncertain about what sort of answer to give.

Exceptions

Questions that by their nature could not have default answers are exceptions. Also, questions that request a password should not provide default answers.

Example

```
Device to be used for displaying the DECwindows IVP [:0.0]:
```

The example shows the default answer in the square brackets.

For more information None

disk_space

The preinstallation section of the installation guide contains the required amount of disk space both during and after the installation completes.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Technical Writer or Software Engineer

REQUIREMENTS AID(S): REQ_CHECK for OpenVMS

Description

All disk space information should be consistent with the disk space information listed in the System Support Addendum (SSA) and checked for by installation procedure. Make sure that if the user has different installation options, the installation procedure checks the appropriate disk space value on the user's selection.

The preinstallation section should list the required disk space needed during the installation (peak) and after the installation completes (net). Instructions for determining the amount of free disk space on the user's system should also be included.

For example, if there is a full development kit and a run time kit, disk space requirements for each kit should be listed. If there are multiple product components, each component should be listed separately with disk space information for each component.

In addition, if there are parts of the kit that are optionally installed, disk space requirements for these components should also be listed. For example, if there are example files that can be installed optionally, there should be a separate disk space value listed for the example files.

International Products

For international products, be sure to include the required disk space for each product variant.

Rationale

To help the installer plan correctly on systems where disk space may be at a critical point. The installer will know up front how much disk space will be needed to install and run the product.

Exceptions

None

Example

Not available

For more information

None

dollar_sign

No dollar sign ($) is used as a file delimiter.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OSF/1	Sun	ULTRIX
SCO UNIX	AIX	HP-UX

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

Do not use the dollar sign ($) as a file delimiter. It is a special character to UNIX-based shells.

Rationale

The use of the dollar sign ($) as a file delimiter causes the operating system to evaluate it as a symbol for an unintended operation.

Exceptions

None

Example

Not applicable

For more information

See also the note on the dollar sign ($) in requirement regis_software_attrib_vms .

env_test

If the product uses a windowing user interface, the interface should be tested in a heterogeneous environment comprising two or more of the platforms listed in the Software Product Description (SPD) and System Support Addendum (SSA).

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Layered products that use the windows service should test for interoperability of their product in the key areas listed below. The environment for testing should take into consideration the various types of color management, bit/byte order, and key competitive platforms. A windows product should functionally interoperate with the following NAS-supported platforms.

OpenVMS
ULTRIX/RISC
ULTRIX/VAX
MS-Windows running Excursion
MS–DOS running PC DECwindows
MIT X11 server
Macintosh running MacX
Sun workstations running OpenWindows

Rationale

Testing interoperability ensures that the product displays properly in other environments where they are commonly used and in a manner consistent with the customer's expectations.

Exceptions

None

Example

The following steps are an example of how to test interoperability of a layered product. However, these steps do not guarantee that the product functions correctly from an interoperability point of view.

DISPLAY INTEROPERABILITY

Relates to the interoperability with the window manager running on the X server.

1. Start the windows application on its native server.
2. Shrink the window to an icon.
3. Expand the icon.
4. Maximize the window.
5. Minimize the window.
6. Move the window arbitrarily.

COLOR FONT

Screen Aspect ratio testing.

1. Start the windows application on the server under test.
2. Verify initial mapping.
3. Verify initial "look and feel."
4. Open all first-level dialogue boxes.
5. Choose a dimmed object.
6. Choose an invalid object (such as Print if no printer is available).
7. Check input focus by pushing and popping windows.
8. Recreate any previously known problems.
9. On the native server, recreate problems found on the remote server to verify whether there is an interoperability error.
10. Execute one function option, such as create, and send a mail message.
11. Exit from the DECwindows application using the pulldown menu.
12. Verify the icon image is correct.
13. Verify the title bar is correct.

For more information

If You Need	Do the Following
To obtain a copy of the DECwindows programming documentation	Refer to the directory BULOVA::DECW$PUBLIC:[V3.DOCS] for V1.0 or BULOVA::DOCD$:[DECW$MOTIF011] for V1.1
Up-to-date information on DECwindows	Consult the BULOVA::DECWINDOWS Notes conference

err_and_fixes

An Appendix of the installation guide contains possible error messages and recommended fixes.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	MS–DOS	Windows
Windows NT	Mac	

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

An Appendix of the installation guide should list possible error messages that may appear during the installation, as well as the appropriate error recovery mechanism. A list of possible causes for errors should also be included. Some examples of these causes are incorrect operating system version, incorrect prerequisites, invalid license, and so on.

Rationale

To help the installer react correctly to installation error messages and to minimize calls to service centers.

Exceptions

None

Example

OPENVMS

The following is an example of an error recovery statement in the installation guide:

Error Recovery

If errors occur during the installation or when the Installation Verification Procedure (IVP) is running, VMSINSTAL displays failure messages. If the installation fails, the procedure displays the following message:

```
%VMSINSTAL-E-INSFAIL, The installation of DECforms V1.4 has failed.
```

Errors might occur during the installation if any of the following conditions exists:

- The version of the operating system is incorrect.
- A required software version is incorrect.
- Quotas are insufficient.
- System parameter values are insufficient.
- The OpenVMS Help library is being used.
- The product license has not been registered and loaded.

For descriptions of the error messages generated by these conditions, see the *OpenVMS System Messages and Recovery Procedures Manual: Part I* or the *OpenVMS System Messages and Recovery Procedures Manual: Part II*. If any of these error messages is displayed, take the appropriate action as described in the message. (You might need to change a system parameter or increase an authorized quota value.) For information on installation requirements, see the preinstallation Section.

UNIX-BASED SYSEMS

Refer to the diagnostics section of the dms(8) reference page if an error occurs while you are attempting to load subsets and the error is not discussed in this guide.

Make the correction and enter the dms command again to restart the installation.

For more information

Refer to pertinent operating system manuals, layered product installation guides, release notes, online help, that is, man pages, for recovery procedures.

error_free_dialogue

The installation dialogue is free of text errors, incorrect use of grammar, and spelling errors.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

See the requirement statement.

Rationale

To avoid any confusion about the intent of the installation, to promote user-friendly installations, and minimize calls to service centers. Also, poor quality in user display will be interpreted as evidence of poor quality software, and this will reduce the customer's confidence in the product as a whole.

Exceptions

None

Example

ULTRIX EXAMPLE:

```
Would you like to run the IVP, after the installation?
```

PC EXAMPLE:

```
Insert disk 1 and press the <RETURN> key.
```

For more information None

fac_mnemonic_in_instl

All global symbols and logical names used during the installation should be prefixed with the product's facility mnemonic.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): REQ_CHECK

Description

Place the mnemonic in front of the dollar sign ($).

Rationale

To prevent conflicts of symbols used in VMSINSTAL or any other global symbols used on the system.

Exceptions

None

Example

```
BASIC$VMS_OK
```

For more information

See also the following requirements

- regis_prod_attrib
- regis_software_attrib_vms

file_naming

File naming conventions should be adhered to for reasons of consistency, organization, and ease of file identification.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): REQ_CHECK

Description

It can be extremely difficult to identify which files are part of a particular software product. Files that serve a specific purpose and are common among many software products (for example, startup files) are not named in a consistent manner. This random method of file naming creates a difficult environment for someone, such as a system manager, to maintain many software products and their associated files. Therefore, following file naming conventions is highly recommended.

Defining a standard file naming convention ensures compatibility with third-party and customer applications. Digital software products are required to use the dollar sign ($) (which is reserved for Digital use) in their file names. Since these products can use from 1 to 39 characters in file names and file types, and the dollar sign ($) is reserved, it is possible to associate files with the owner of the product, the purpose of the files, and their Digital origin.

Layered product groups planning product variants should make sure that the separate variants are easily distinguishable, for example:

```
QUAL$SERVER_GERMAN.DIR
QUAL$SERVER_FRENCH.DIR
```

In all file specifications for a local-language variant, include the language qualifier (for example, German, French) in either the directory name or in the file name.

File Naming For Multi-platform Products

If your product will be ported to the other platforms where the dollar sign ($) in the file name is not accepted, you can substitute the underscore character for the dollar sign ($) and continue to use the registered facility name.

Refer to the *Guide to Portable Software* for more details. See the For More Information section at the end of this requirement description for a pointer to obtaining a copy.

File Name Components

All file names should consist of the following components:

- The product's registered facility name
- The dollar sign ($), which is reserved for Digital use
- An identifier string (optional)
- A string that identifies the file's purpose

Using this convention, construct file names in the following format:

```
facility_name$identifier_purpose.file_type
```

The optional identifier string adds the flexibility of having multiple files serving the same purpose.

Naming of Common Files

Software products often contain files that are common among many products. Some examples of commonly used files are Help files, message files, and run time libraries. To promote consistency in the naming of these files, all products shipping common files must adhere to the following naming conventions:

Command Definition Files	= facility_name${identifier_}.CLD
Data Files	= facility_name${identifier_}.DAT
Data File Definition Files	= facility_name${identifier_}.FDL
Help Files	= facility_name${identifier_}.HLB
Initialization Files	= facility_name${identifier_}.INI
Journal Files	= facility_name${identifier_}.JOU or = facility_name${identifier_}.FAC$JOURNAL
Link Answer Files	= facility_name${identifier_}.ANS
Log Files	= facility_name${identifier_}.LOG
Main Images	= facility_name${identifier_}MAIN.EXE
Message Files	= facility_name${identifier_}MSG.EXE
Object Libraries	= facility_name${identifier_}.OLB
Option Files	= facility_name${identifier_}.OPT
Release Notes	= facility_name{version}.RELEASE_NOTES
RTL Images	= facility_name${identifier_}RTL.EXE
Shareable Images	= facility_name${identifier_}SHR.EXE
Start-up Files	= facility_name${identifier_}STARTUP.COM

If G float and H float run time libraries are appropriate, use the following naming conventions :

G Float RTL Images	= facility_name${identifier}_RTL_G.EXE
H Float RTL Images	= facility_name${identifier}_RTL_H.EXE

If client/server products have node-specific files that need to be named appropriately, use the following naming conventions:

Client Images	= facility_name$CLIENT_{purpose}
Server Images	= facility_name$SERVER_{purpose}

Not all products need to use the identifier string. Products that do not have multiple files serving similar purposes (for example, more than one shareable image library, help, and startup files) may have no need to include the identifier string. In the case of VAX Notes, which has only one startup file, NOTES$STARTUP.COM (without an identifier string) is clearly sufficient.

The following examples use the hypothetical product VAX Quality to illustrate proper naming of client and server files:

```
QUAL$CLIENT_MAIN.EXE
QUAL$SERVER_MAIN.EXE
QUAL$CLIENT_STARTUP.COM
QUAL$SERVER_STARTUP.COM
QUAL$CLIENT_SHR.EXE
QUAL$SERVER_SHR.EXE
QUAL$CLIENT_MSG.EXE
QUAL$SERVER_OPTIONS.OPT
QUAL$RTL.EXE
QUAL$OBJLIB.OLB
QUAL$HELP.HLB
QUAL$INIT.INI
QUAL010.RELEASE_NOTES
```

Directory Names

Top-level directory names should remain consistent with the file naming convention. A correct directory name contains the product's registered facility name, the dollar sign ($) reserved for Digital (optional), and an identifier string (optional).

Appropriately named directories and files will increase organization and consistency within Digital software products.

Account Names

The accounts created by the product must also be consistent with the file naming convention. The account name should contain:

- The product's registered facility name
- The dollar sign ($) reserved for Digital
- An identifier string.

For example, a server account that maps to a DECnet object would have "QUAL$SERVER" as account name.

Other System Entities

In general, any systemwide entities created or required by a product, such as any rights identifiers or logical names, must include the facility name and the dollar sign in the prefix of the entity name.

An example of a rights identifier is QUAL$IMAGE.

Use of Image ID Field Name

OpenVMS layered products should use the File Image ID Area field to identify the name and version number of their product. These fields can be shown by performing an ANALYZE/IMAGE on any .EXE files. The field is 15 characters long and should be constructed as follows:

```
[PRODUCT NAME] [VERSION NUMBER]
```

In the above example, the product name should be substituted by the facility mnemonic.

Because products are being developed to support international markets and can be re-engineered to support localization (for example, Asian markets such as Japan and China) it is important that you reserve at least one character in the version number so that re-engineered images can be identified by language.

The standard format of a version number consists of a compound string constructed of the following discrete items:

```
[SUPPORT][VERSION].[UPDATE] - [EDIT][PATCH][LANGUAGE_IDENTIFIER]
```

The following list describes each of these discrete items:

- [SUPPORT]—Single capital letter (or null) identifying the support level of the program:

  B = Benchmark version
  D = Demonstration version
  S = Special customer version
  T = Field test version
  V = Released or frozen version
  X = Unsupported or experimental version

- [VERSION]—Major release, generation, or base level of a program.
- [UPDATE]—If present, period followed by a single decimal digit indicating a minor release.
- [EDIT]—If present, minus sign followed by a maintenance number; identifies any alteration of the source code.
- [PATCH]—If present, single capital letter identifying an alteration of the program's binary object form.
- [LANGUAGE_IDENTIFIER]—If the product is re-engineered to create a localized version (language variant) of a product (for example, Japanese FMS), this field should be used to indicate the language of the product variant. This allows products that are dependent on a specific language variant to check for the language variant's existence and version number. The language code should comply with the conventions specified in *DEC STD 012-4 Unified Numbering Code (UNC) - Software Numbering Conventions*. Refer to the For More Information section at the end of this requirement description for more information on how to obtain a copy.

Note

Since the international product model supports the localization of most non-Asian languages without the need for re-engineering, do not use this field for the basic international version of products.

The following examples show some uses of the language identifier:

```
EXA V3.0-00     for VAX Example (Basic International Version)
EXA V3.0-00J    for VAX Example Japanese
EXA V3.0-002    for VAX Example Hanzi
EXA V3.0-003    for VAX Example Hanyu
EXA V3.0-004    for VAX Example Hangul
```

You can set the image ID field by using a linker option (for example, IDENT= "15 byte string"). In this case, you must use the double quotation marks to delimit the string.

Rationale

Using file naming conventions promotes an added sense of consistency, organization, and ease of identification to the files of the OpenVMS product family.

Exceptions

None

Example

Not applicable

For more information

If You Need	Do the Following
To obtain a copy of the *Guide to Portable Software*	Copy the following file: PIPE::AIA$INFO:GD_TO_PORT_SW.PS
To obtain a copy of DEC STD 012-4 and DEC STD 204-0	Consult the Corporate VTX data base. At the system prompt, enter: VTX SMC [Return], and follow the instructions.
Information on the use of image ID field, and image ID of shareable images	See requirement image_id_in_prod_name

file_placement_during_instl

Files should be placed in the SYS$COMMON area of the system disk and should be referenced using the VMI logicals.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): REQ_CHECK

Description

Unless there is a specific reason for placing files in a system-specific area, files should be placed in the SYS$COMMON area of the system disk.

File placement during the installation should use the VMSINSTAL logical top-level directory specification, followed by the physical directory name. The logical VMI$ROOT points to the system top-level directory, and the logical VMI$SPECIFIC points to the system-specific top-level directory. No reference should be made to system logical names. For example, VMI$ROOT:[SYSLIB] should be used, not SYS$LIBRARY.

Rationale

The use of the SYS$COMMON area gives a product the capability of being accessed from all nodes in a cluster without the need to re-install the product on each of the cluster nodes. Also, the VMI logical references should be used as they allow the top-level directory specification to be optionally changed from the system's current system disk by using VMSINSTAL alternate root (R) option.

Exceptions

When the nature of the file is such that it is node-specific, or when the product is installed in an alternate location using VMSINSTAL option (R).

Example

```
VMI$CALLBACK PROVIDE_FILE A1$ A1WPSPLUS_LOGIN.COM VMIS$ROOT:[SYSMGR]
```

For more information None

final_prod_in_cserl

The final product must be included in the Central Software Engineering Release Library (CSERL).

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Product Manager

REQUIREMENTS AID(S): None

Description

Once your product is complete, it is required that it be included in the Central Software Engineering Release Library (CSERL). (The CSERL is maintained by Electronic Information Integration & Delivery (EIID) Engineering in support of the EIID Program.)

The product components required in the library are:

- Binaries, for example, subsets, savesets, tar files and so on
- SPD/SSA in PostScript(tm) and ASCII text format
- Installation guide in PostScript and ASCII text format
- Release notes and ASCII text format
- Cover letter (READ_ME_FIRST, BEFORE_U_INSTALL) in PostScript and ASCII text format

The components must be the SAME as those in Software Supply Business (SSB) that are shipping via traditional media.

For information contact FRMENG::CDROM to arrange to have the components inserted.

Rationale

Software product binaries and their related information (On-Line Documentation, PostScript images of Books, Installation Guides, and so forth.) are stored in this library and are made available for delivery to update service customers on CD-ROM consolidations and on traditional media. It also allows software development organizations to integrate their products along with marketing /merchandising information on "Marketing Demo," Industry/Market Segment, and Geographic Segment consolidated CD-ROMs. Also, the use of the CSERL eliminates the need for submitting physical masters for building traditional media.

Additionally, this library provides an environment for proactive product integration and interoperability testing of point products or groups of products by testing engineers.

The CSERL is the primary delivery/release mechanism of software products into the SSB. Products must reside in this library in order to participate in consolidated CD-ROM programs as well as ship traditional media (under full implementation).

Exceptions

It is the general rule that all software products be inserted into the CSERL. However, if a product development group believes that there is compelling business or technical justification to waive this requirement, discussions regarding waiver can be initiated by contacting EIID Engineering (FRMENG::CDROM).

Example

Not applicable

For more information

For information regarding the CSERL or EIID, contact Paul Hague (VIRTUE::HAGUE).

font_fallback

The product provides for font fallback to the MIT X Server font set when displayed to a non-Digital Equipment Corporation server.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OpenVMS OSF/1 Sun ULTRIX
PERFORMED BY:	Software Engineer
REQUIREMENTS AID(S):	None

Description

When a windowing layered product is displayed to a non-Digital X server, it is possible for the application to request a font that does not exist on that server. An application must check to see if the requested font is available on a server and, if not, fall back to a font available with the industry standard MIT X server.

Rationale

Requesting nonexistent fonts may cause the application to be unusable or to crash.

Exceptions

None

Example

Not applicable

For more information For more information on fonts, refer to *X Window System*.

foreign_com

Foreign commands are not used to invoke the product.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

The use of foreign commands is strongly discouraged and may be the cause of a delay in product sign-off. Before defining and using foreign commands in your product, obtain the approval from the Digital Command Language (DCL) Review Board.

A “real” command is guaranteed to be available to, and consistent for, every user.

Since a foreign command is just a DCL symbol, users are required to add it to their LOGIN.COM file. Users would have to know the file name of the program image, since that must be specified as part of the foreign command symbol definition. Also, it is likely that each user will have their own name for a foreign command, which is not conducive to consistency.

By contrast, a real command is defined once when the product is installed and is automatically available to all users thereafter. Since the product location is also specified as part of the installation procedure, users do not need to know anything about the image file. With real commands, users can always define DCL symbols to create aliases (for example, $ CD := = SET DEFAULT) or change defaults (for example, $FOR*TRAN := = FORTRAN /LIST"), but there is at least a guaranteed default command and behavior.

Rationale

Foreign commands are not a standard approach to file execution.

Exceptions

Approval granted by the DCL Review Board.

Requesting DCL Review Board Approval

To request approval from the DCL Review Board to use foreign commands:

1. Enter a note in the VIRTUE::DCLREVIEW Notes conference
2. Provide the following details:

   Product name
   Product version number
   Reason(s) for using foreign commands instead of DCL syntax

The DCL Review Board posts its decisions as a reply to the original entry.

Example

Not applicable

For more information

Refer to the VIRTUE::DCLREVIEW Notes conference to view requests submitted by layered product groups and the DCL Review Board's decisions.

hardcoded_saveset_names

For ease of translation, it is recommended not to use hardcoded saveset names, but to use the logical VMI$PRODUCT instead.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Hard-coded saveset names in the KITINSTAL.COM can create problems if the product is to be translated. This is because the translated product is required to use saveset names according to the International Product Model.

For ease of translation Software Quality Operations and Information Services (SQOIS) recommends *not* to hard-code saveset names, but to use the logical VMI$PRODUCT instead.

Product variants produced in Europe, and re-engineered products produced in Asia *must* have unique saveset names. Translated products are required to use saveset names according to the International Product Model.

Rationale

Hard-coding saveset names in the KITINSTAL.COM creates problems if the product is subsequently translated.

Exceptions

Products built according to the International Product Model may contain savesets that do not have the same name and therefore cannot all be referenced using the VMI$PRODUCT logical.

Example

None

For more information

If You Need	Do the Following
A description of the International Product Model	See requirement i18n_prod_model
A copy of *DEC STD 204-0*	Consult the Corporate VTX data base. At the system prompt, enter: VTX SMC [Return] and follow the instructions.

i18n_prod

The product should be designed to be international.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Layered products should incorporate the following design principles for international products regardless of the current localization plans for the product.

- Separate input and output text in the user interface from the code that presents it.
- When appropriate, use existing internationalization services for functions such as collating, display of formatted data, and data input and output.
- If the above does not apply, use table-driven and modular replacement techniques to design code for the international base and market-specific components that can be easily adapted to international requirements.
- Design the product for possible extension of support to the following:
  - Multilingual user interfaces and functionality
  - Communication between multilingual processes
  - Varying terminals, keyboards, and keyboard selection
- Use the translatability checklist from EL-SM498-00, *Producing International Products Reference Set* to check your product's adaptability into a local-language variant. In addition, contact the appropriate localization center early in the development cycle to jointly plan a prototype translation.

Note that many of these requirements can be accomplished using technologies such as XPG/3, XPG/4, Posix, CDA, Unicode, and Native Language Services (NLS) for nonwindowing interfaces, and User Interface Language (UIL) files for windowing interfaces.

Rationale

Digital is committed to move as quickly as possible towards making its products simultaneously available in Japanese as well as other languages. Incorporating international design principles from the beginning allows easier translation of the product into local-language variants, reduced time-to-market, and reduced translation costs.

Exceptions

None

Example

See *Producing International Products* for many examples of how to internationalize a product.

For more information

If You Need	Do the Following
More information on designing internationalized products	Refer to the *Producing International Products Reference Set* Refer to *DEC STD 066-3 Policy for Designing Products for all Countries Designated as Strategic Markets* Refer to the *Digital Guide to Developing International Products* (Digital Press Order Number: EY-F577E-DP) Refer to the *OSF/Motif Style Guide* Send mail to I18N::HOTLINE
More information on localization centers	Contact ISE at I18N::HOTLINE (if an agreement has been negotiated)
Information on designing for portability	Refer to the *X-Open Portability Guide* V3.0 (XPG /3) for European products, or V4.0 (XPG/4) for European and Asian products.
Descriptions of related requirements	Refer to requirements i18n_uil and i18n_rtl .

i18n_prod_model

Products that will be localized should be designed according to the International Product Model.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

The International Product Model defines an international product to consist of the following components:

- International Base Component (A Component)
- User Interface Component (B Component)
- Market-Specific Component (C Component)
- Country-Specific Information Component (D Component)

Understanding, separating, developing, and packaging these components is essential for building good international products. Separating the product into components makes it easier to develop language, country, and market variations. It also reduces both the development and maintenance costs of the product variations.

The International Base Component (A Component)

The International Base Component (A Component) is the invariant part of a product and is sold worldwide without modification. Note that while the A Component is itself invariant, it may feature built-in variants that are selected by a user, perhaps by switch selection in the case of hardware or by parameter setting in the case of software.

The User Interface Component (B Component)

The User Interface Component (B Component) is the language and text processing component. It must be localized to the language and culture requirements of a specific country. The B Component typically contains the user interface, including messages, text and language processing, formats, online help, and documentation.

The Market-Specific Component (C Component)

The Market-Specific Component (C Component) is any part added to meet special requirements of a specific market, which could be a country, an industry or business, or a group of customers sharing a language and set of cultural conventions. Note that C Components add specialized functions to the A Component, which must include interfaces to the C Components in its design. C Components provide a way to extend the A Component without requiring any changes to the A Component.

Country-Specific Information Component (D Component)

The Country-Specific Information Component (D Component) is a set of mandatory documentation, packaging and labeling, warranty, support, and product description information added in order to meet all the regulations required to sell the product in a specific country. Note that this meaning of country-specific does not include a special function or code supporting a country's unique requirements, which would be included in the C Component.

Rationale

Products designed to conform to the International Product Model can be easily and efficiently localized because:

- Elements that must be changed for some markets have been isolated in the B and C Components. Elements that are inappropriate for a market can be eliminated or exchanged for elements designed for that market.
- User-visible text has been eliminated from the A Component. As a result, only text in the B Component and perhaps in the C Component needs to be translated. For software, this means that no functional or user interface code needs to be rewritten to change languages or other attributes of the locale.

Exceptions

None

Example

Not applicable

For more information

If You Need	Do the Following
Information on producing international products	Refer to the *Digital Guide to Producing International Products*
Assistance on internationalizaton issues	Contact I18N::HOTLINE (if you have negotiated an agreement).

i18n_rtl

To increase the level of international support available in the international base component of a product, it is strongly recommended that the product be designed to use the appropriate international services available on each platform.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OpenVMS	OSF/1	Sun	ULTRIX

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK for OpenVMS

OpenVMS

The following sections are specific to OpenVMS layered products.

Description

If you are programming in C, you must use the XPG/4 internationalization routines available with DEC C. These are X/Open-compliant routines that provide a consistent set of internationalization services with UNIX-based environments.

Although bindings for these routines are not yet, available in other OpenVMS programming languages, it is possible to call them from programs written in other OpenVMS programming languages.

Rationale

Incorporating international design principles from the beginning allows easier translation of the product into local-language variants.

Exceptions

DCL command files cannot use the XPG/4 routines, but they must be able to handle internationalized error messages.

Example

Refer to the EL-SM498-00, *Producing International Products Reference Set* for many examples of designing an international product.

ULTRIX, OSF, Sun

The following sections are specific to ULTRIX, OSF, and Sun layered products.

Description

To design nonwindowing products, whether they are being localized or not, you must use Native Language Service (NLS) routines.

NLS is part of the base ULTRIX operating system. It is an X/Open-compliant mechanism that provides a consistent set of internationalization services in the UNIX environment.

All output, including log files, standard error, and standard output should use NLS routines. Basically, any type of "print" statement included within the source code should make use of NLS.

Rationale

Incorporating international design principles from the beginning allows easier translation of the product into local-language variants.

Exceptions

If the product existed before NLS or run time libraries existed, it would be less costly for the product to keep its existing routines than to migrate to NLS or run time libraries.

Due to backwards compatibility issues and historical reasons, Japanese ULTRIX (also known as J-ULTRIX) contains the NLS message catalogs in the directory /usr/jsy/lib/nls/msg.

Asian ULTRIX provides NLS in /usr/asy/lib/nls and Asian OSF/1 provides it in /usr/lib/nls. The implementation of NLS on ULTRIX for Asian products is the same as for Japanese products.

Example

Refer to the *Digital Guide to Producing International Products* for many examples of designing an international product.

For more information

If You Need	Do the Following
Information on NLS implementation on ULTRIX and OSF/1	Refer to the VAX Notes conference SMURF::I18N.
More information on designing internationalized products	Refer to the *Producing International Products* Reference Set Refer to *DEC STD 066-3 Policy for Designing Products for all Countries Designated as Strategic Markets*

i18n_test

Products that are designed to be international run correctly in a heterogeneous hardware and software environment.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Products that are international and translated need to demonstrate the following:

- Interoperability of the various product variants, for example, share, transfer, and store data, including data from different character sets.
- The capability to run multilingual systems, if supported.
- Correct behavior on market-specific devices, such as display of multinational character sets on international keyboards.
- The ability to display data on various devices.
- The ability to support various language specific keyboards.
- Public Telephone Telegraph (PTT) compliance, where applicable.

It is strongly recommended that the internationalization ability be demonstrated before the Basic International Version (BIV) is fully developed and shipped. To ensure that this type of testing is carried out, contact the appropriate localization center early in the development cycle to plan a prototype translation.

Rationale

To demonstrate the international character of the product as defined in the Internationalization Plan.

Exceptions

None

Example

Not applicable

For more information Contact the I18N::HOTLINE for consulting services on internationalization issues.

i18n_uil

Products that use a windowing user interface are designed to be international by separating all text into User Interface Language (UIL) modules for ease of translation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS OSF/1 Sun ULTRIX

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Place all text for the user interface into a User Interface Language (UIL) module, which will be translated into a User Interface Definition (UID) file. Changing the definition of the UID hierarchy in the application program allows changing the interface language.

Rationale

Using separate modules allows changing the user interface for different languages. Localization groups can then easily modify the code without affecting the bulk of the product's functions.

Exceptions

None

Example

For examples of conformance to this requirement, refer to the section "Customizing a Motif Interface Using UIL and MRM" in the *OSF/Motif Programming Guide*.

For more information

If You Need	Do the Following
More information on programming with UIL	Refer to the *OSF/Motif Programmer's Guide*

If You Need	Do the Following
More information on designing an internationalized product	Refer to the *OSF/Motif Style Guide*

ig_templates

Using installation guide templates is highly recommended to ensure consistency among products.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
Windows NT	Windows	Mac
MS–DOS		

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): None

OpenVMS The following sections are specific to OpenVMS layered products.

Description

Installation guides provide users with all the information necessary to install a particular software product. They vary from product to product, but they have many components in common, such as the following:

- Front matter
- Preface
- Introduction
- Preinstallation requirements
- Procedure to install the product
- Procedure to run the installation verification procedure
- Postinstallation requirements
- Error recovery
- Appendices

Refer to the For More Information section at the end of this requirement description for more information on the filename and location of installation guide templates.

Rationale

Installation guide templates promote consistency in installation descriptions for all Digital products. They also contribute to successful product installations by covering all necessary installation aspects, such as prerequisite products or system parameters.

Exceptions

If business requirements dictate a different installation guide format, you are not required to use the available installation guide templates. You should ensure that your product's installation guide covers at least the various aspects outlined in the description section of this requirement.

Example

Not applicable

The following sections are specific to ULTRIX, OSF, and Sun layered products.

Description

An installation guide provides a user with all the information necessary to install a particular software product. Installation guides vary from product to product, but they have many components in common, such as the following:

- Front matter
- Preface
- Introduction
- Procedure to prepare product for installation
- Procedure to install the product
- Installation Verification Procedure (IVP)
- Diskless management services (ULTRIX and OSF/1 only)
- Remote installation services (ULTRIX and OSF/1 only)
- Postinstallation requirements
- Error recovery
- Appendixes

Installation guide template files for OpenVMS, OSF, and SUN operating systems can be copied from BOOKIE::PUBLIC:[PUBLIC].

Rationale

Use of the installation guide template helps to ensure that all ULTRIX layered product installation guides will have similar appearances and contents.

Exceptions

Business requirements may occasionally necessitate the use of alternative installation formats. While it is recommended that alternatives be minimized where possible, other implementations are acceptable to meet business needs provided all operating system integration environments covered in the respective installation guide template are adequately described.

Example

Not applicable

For more information None

image_deinstl

If the installation procedure needs to invoke the INSTALL utility to install an image needed to complete the installation, that image must be deinstalled using the INSTALL utility before exiting the installation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

See the requirement statement.

Rationale

Because customer environments are so varied, the installation should prevent forcing undesirable changes to the system.

Exceptions

None

Example

Not applicable

For more information None

image_id_in_prod_name

OpenVMS layered products should use the File Image ID Area field to identify the name and version number of their products.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): None

Description

OpenVMS layered products should use the File Image ID Area field to identify the name and version number of their product. These fields can be shown by performing an ANALYZE/IMAGE on any .EXE files. The field is 15 characters long and should be constructed as follows:

```
[PRODUCT NAME] [VERSION NUMBER]
```

In the previous example, the product name should be substituted with the facility mnemonic.

Because products are being developed to support international markets and can be re-engineered to support localization (for example, Asian markets such as Japan and China), it is important that you reserve at least one character in the version number so that re-engineered images can be identified by language.

The standard format of a version number consists of a compound string constructed of the following discrete items:

```
[SUPPORT][VERSION].[UPDATE] - [EDIT][PATCH][LANGUAGE_IDENTIFIER]
```

The following list describes each of these discrete items:

- [SUPPORT]—Single capital letter (or null) identifying the support level of the program:

  B = Benchmark version
  D = Demonstration version
  S = Special customer version
  T = Field test version
  V = Released or frozen version
  X = Unsupported or experimental version

- [VERSION]—Major release, generation, or base level of a program.
- [UPDATE]—If present, period followed by a single decimal digit indicating a minor release.

- [EDIT]—If present, minus sign followed by a maintenance number; identifies any alteration of the source code.
- [PATCH]—If present, single capital letter identifying an alteration of the program's binary object form.
- [LANGUAGE_IDENTIFIER]—If the product is re-engineered to create a localized version (language variant) of a product (for example, Japanese FMS), this field should be used to indicate the language of the product variant. This allows products that are dependent on a specific language variant to check for the language variant's existence and version number. The language code should comply with the conventions specified in *DEC STD 012-4 Unified Numbering Code (UNC) - Software Numbering Conventions*. Refer to the For More Information section at the end of this requirement description for more informatin on how to obtain a copy.

Note

Since the International Product Model supports the localization of most non-Asian languages without the need for re-engineering, do not use this field for the basic international version of products.

The following examples show some uses of the language identifier:

```
EXA V3.0-00     for VAX Example (Basic International Version)
EXA V3.0-00J    for VAX Example Japanese
EXA V3.0-002    for VAX Example Hanzi
EXA V3.0-003    for VAX Example Hanyu
EXA V3.0-004    for VAX Example Hangul
```

You can set the image ID field by using a linker option (for example, IDENT= “15 byte string”). In this case, you must use the double quotation marks to delimit the string.

Shareable Images

Some layered products use images that are shareable or are supplied by another group. These images should follow the standard described in this section. The group supplying the image should set the File Image ID area to reflect the current version of the shareable image so that the image ID does not contain information about any product layered on top of it or bundled with it. It would only contain information about the base product to which it belongs. For example, if the DATATRIEVE development group were to use the image EDTSHR.EXE from the EDT group, the IMAGE ID area of that executable image would contain the following:

```
IMAGE FILE ID:   EDT V3.5-2
IMAGE NAME:      EDTSHR
```

These fields would not be changed to reflect the DATATRIEVE version that uses it, but they should be left to provide the information on the version of the EDT image itself.

Rationale

To promote an added sense of consistency, organization, and ease of identification to the files of the OpenVMS product family.

Exceptions

None

Example

Not applicable

For more information

If You Need	Do the Following
To obtain a copy of the *Guide to Portable Software*	Copy the following file: PIPE::AIA$INFO:GD_TO_PORT_SW.PS
To obtain a copy of DEC STD 012-4	Consult the Corporate VTX data base. At the system prompt, enter: VTX SMC [Return], and follow the instructions.
Information on file naming conventions	See requirement file_naming .

instal_code_exec

No code is executed after the Installation Verification Procedure (IVP) is run.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): None

Description

The KITINSTAL.COM must not execute any code after the Installation Verification Procedure (IVP) is run. VMSINSTAL does not expect any commands after the IVP runs.

Rationale

VMSINSTAL cannot run any code after the IVP has been performed. If any code runs after the IVP and fails to execute successfully, it may leave the product and the system in an unstable condition with no further verification to warn the installer.

Exceptions

None

Example

Not applicable

For more information See also the *Developers Guide to VMSINSTAL*.

instal_comments

Comments in the installation procedure are consistent and accurate and do not mention any unreleased Digital products.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

All comments written in installation procedures must be accurate and consistent with the code.

This applies also to all command procedures shipped with the product.

Do not include any comments that would be inappropriate for the customer to read.

No mention should be made of yet unreleased Digital products (hardware or software, including unannounced versions of operating systems).

Rationale

Products will be more consistent and enforce Digital's security policies.

Exceptions

None

Example

For example, a comment stating that a layered product is checking for OpenVMS V5.0 or higher, while the code shows that the product is checking for OpenVMS V5.2 or higher, is unacceptable.

For more information None

instal_space_check

The installation procedure checks for sufficient free space.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

OpenVMS

All installation kits must make sure that there is enough disk space to install the layered product. Use the CHECK_NET_UTILIZATION callback to perform this function.

The value given for the number of free blocks to be checked must be the value required for peak blocks (for installation) rather than net (after installation) blocks. In addition, this block value must be consistent with the value given for peak blocks in the System Support Addendum (SSA) and installation guide. The peak block usage can be obtained by using option S in the VMSINSTAL procedure. To determine peak disk on an alternate disk, use the command procedure named VMSINSTAL_STATS.COM located in {VIRTUE, SQMUK, SQMJPN}::SYS$INFO:.

Products that offer installation options for installing a full or partial kit should compute the required total block space for the chosen option(s) and pass this number to the callback as the peak value.

For example:

Full kit: 30000
Base kit: 25000
Example files: 3000
Help files: 2000

If the installer chooses the base kit and example files to be installed, the PEAK value would be 28000.

Block space requirements for each option must be consistent with the values listed in the SSA and installation guide.

Note that CHECK_NET_UTILIZATION is intended to check the available space on the VMI$ROOT disk or an alternate working device if the VMSINSTAL option AWD has been specified. This check will not check for disk space on target devices other than the system and VMI$KWD disks. If files are to be provided to other disk locations, the installation procedure should check separately for file space on those disks.

Rationale

The installation will fail if there is not sufficient space.

Exceptions

None

Example

```
$ VMI$CALLBACK CHECK_NET_UTILIZATION CHECK_SPACE 20000 12000 10000
```

For more information

Refer to the following:

- The kitinstal_set_safety requirement
- *Developers Guide to VMSINSTAL*.
  Copies can be obtained from: {VIRTUE, SQMUK, SQMJPN}::SYS$INFO:DEC_VMINSTAL_AUG90.*
- The document *VMSINSTAL_STATS_GUIDE.MEM*. Copies can be obtained from {VIRTUE, SQMUK, SQMJPN}::SYS$INFO:

instl_dialogue

Installation guide should contain the exact dialogue as it will appear on the screen.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

Installation guide should contain any informational or warning messages that may appear during installation. All questions should be explained and all installation paths should be presented. The Installation Verification Procedure (IVP) dialogue (if implemented) should also be included after the installation dialogue.

Rationale

To make the installer aware that the installation procedure could have options that may assist during the installation, help prepare for installation, or allow alternative installation choices.

Exceptions

None

Example

Not available

For more information None

instl_exception

Thorough installation exception testing should be performed.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

It is recommended that layered-product groups perform exception testing during installation of their products. Exception testing tests the installation /reinstallation with incorrect or insufficient data to make sure that it fails gracefully.

The following tests are recommended:

- Install the product on the supported versions of operating systems. Pay attention to unfamiliar operating systems version-naming standards; for example, V3.1a. Field test versions of operating systems must be considered, since many customers act as field test sites for operating systems.

  Exception Test: Install on versions of the operating systems that the product does not support.

- Install the product with the proper prerequisite software on the system.

  Exception Test: Install without prerequisite software and with prerequisite software that is the wrong version. If your product depends on the prerequisite software being installed with various options, install prerequisite software without the required options.

- Install the product with optional software on the system.

  Exception Test: If the product optionally links or accesses optional software during the installation procedure, install the product without optional software to ensure that no dependencies exist.

- Exercise the various code paths of the installation procedure.

  Exception Test: Try entering nonsensical input in response to installation questions. This will exercise the error-checking subroutines.

- Install the product on a system that has the previous versions or multiple versions of the software, to ensure that a customer's upgrade path will be successful.

  Exception Test: Install the product without any previous version of the product on the system. Internal development systems often depend on files that are difficult to detect; this testing removes those dependencies.

- For operating systems supporting the Licensing Management Facility (LMF), install the product with LMF_PAK installed.

  Exception Test: Install the product without LMF_PAK installed.

Rationale

Exception testing often invokes previously unexercised routines and can uncover routines that do not work, routines with no labels, or routines that do not properly modify status codes.

Exceptions

None

Example

Not applicable

For more information

For advice on installation exception testing techniques, refer to the VIRTUE::TESTING VAX Notes conference.

instl_file_rename

During the preinstallation and postinstallation phases, files should not be modified. If files have to be modified, the user should be notified.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

UNIX

The /usr/kits files (for ULTRIX) and /usr/opt files (for OSF/1) are intended to be read-only directories.

PC

The following options are available:

1. Do not modify files in read-only directories
2. Give users the choice
3. Give a sample file so the user can see what has changed
4. User reviews and modifies files during the installation

Rationale

To prevent surprises, files should not be modified without the users knowing about it.

Exceptions

Macintosh installations will perform file updates automatically

Example

Not applicable

For more information

- For ULTRIX products, *Guide to Disk Maintenance*
- For OSF/1 products, *Guide to Programming Support Tools*

instl_proced_phases

The installation procedure should be designed around three phases: the preinstallation phase, the actual installation phase, and the postinstallation phase.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

The installation procedure should be designed around three phases:

Preinstallation
This phase should perform all environment checking for product requirements. All necessary user input is obtained at this stage.

Actual installation
In this phase the actual installation work is carried out. Examples of activities taking place are: product directories and accounts are created, files are moved to their target directories, libraries are built, and so on.

Postinstallation
In this phase the Installation Verification Procedure (IVP) is executed, if available and if selected by the user, and any cleanup work is performed. An exit status is returned to the installation procedure.

SCO UNIX

For SCO UNIX products, the division in the three phases follows the guidelines set up by SCO:

Preinstallation is referred to as 'Initialization'
'Actual installation' is the extraction of the product from the media
'Postinstallation' is referred to as 'Preparation'

No IVP will be supplied; testing correct installation will be left to standard SCO UNIX tooling. No functional testing is performed.

Rationale

To promote consistency of product installation procedures across all products.

Exceptions

None

Example

Not applicable

For more information

None

SCO UNIX

Refer to: Open Desktop Product Engineering Toolkit Guide, Version 4.0.0.

instl_standard_notices

Standard notices must be displayed during installation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): *DEC STD 197-0*, REQ_CHECK

Description

All the standard legal notices (copyright, government data rights, and license notices) listed in Section 2.3.1 of *DEC STD 197-0 Legal Requirements and Guidelines for Digital Publications and Software* must be displayed during installation of layered products.

Rationale

Use of appropriate legal notices and labels protects the security and ownership of Digital's intellectual property.

Exceptions

None

Example

For examples of standard notices, refer to *DEC STD 197-0* for a list of standard legal notices that must be displayed during installation.

For more information

Refer to *DEC STD 197-0* for exact listings of Standard Legal Notices that must be displayed during installation.

instl_status_msg

During lengthy installations, the installation periodically outputs messages to the screen regarding the installation status.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

Many layered product installations are lengthy. In these cases, make sure the installation procedure periodically outputs messages to the screen regarding the status of the installation.

Rationale

Periodic status messages help the user to verify that the installation is progressing successfully.

Exceptions

None

Example

```
Starting to perform all of the installation work...
  .
  .
  .
Now linking driver modules.
  .
  .
  .
Building TSSSHR.EXE...
  .
  .
  .
Updating the ACMS parameter file...
```

For more information None

instl_time

The preinstallation section of the installation guide contains the approximate time required to complete the installation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

See the requirement statement.

Rationale

Allows the installer to schedule any required product downtime and any other resources that may be required to perform or support the installation. It also gives the installer an idea of how long the procedure will take.

This information is also essential to support personnel to enable them to cost installation services.

Exceptions

None

Example

The following is an example documenting the installation time:

```
Installing DECforms and running the Installation Verification Procedure (IVP)
requires approximately 5 to 35 minutes, depending on your system configuration
and the distribution media you are using.
```

For more information None

instl_time_msg_display

The installation procedure should display a message indicating the approximate time needed to complete the installation or the percent completion.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Since the time required to complete installations varies from system to system, a high-end and low-end system should be used as benchmarks.

Rationale

Feedback from customer service centers indicates that this requirement is a high priority with Digital customers. It promotes consistent, uniform, and user-friendly installations.

Exceptions

None

Example

```
The installation will complete in 1 to 4 hours depending on processor type.
```

For more information None

instl_with_kitinstal

The product is installed with the use of KITINSTAL.COM.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): None

Description

The primary installation command procedure for any OpenVMS layered product is KITINSTAL.COM, which is called by VMSINSTAL.COM. The installation process can be broken down into three phases:

- Preinstallation
- Installation
- Postinstallation

Rationale

Using KITINSTAL.COM is required by VMSINSTAL.COM. VMSINSTAL supports all valid system configurations, promotes installation consistency among all Digital layered products, and guards against compromising system security.

Exceptions

None

Example

Not applicable

For more information

If You Need	Do the Following
Information on installation and callbacks	Refer to *Developers Guide to VMSINSTAL*.

If You Need	Do the Following
To organize the installation procedure	Refer to requirement instl_proced_phases

invoking_vmsinstal

The installation section of the installation guide contains instructions for invoking the install program.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer
REQUIREMENTS AID(S): None

Description

The first part of the installation section should include instructions for invoking VMSINSTAL, with an explanation of the different options available, including OPTION N for displaying or printing online release notes.

The command line for invoking VMSINSTAL must contain the saveset name. An asterisk (*) must not be used for savesets because unexpected results could occur when the distribution media contains more than one kit. For multiple component products, be sure to include the saveset name for each component. For international products, include the saveset name for each product variant and any language-specific parameters that may be required. You may also include the command to invoke each product variant, if applicable.

Rationale

Tells the installer how to retrieve the release notes.

Exceptions

None

Example

OPENVMS

The following example illustrates the use of OPTION N in the installation guide.

OPTION N

An optional parameter that indicates you want to see the release notes. If you do not include the OPTION N parameter, VMSINSTAL does not prompt you to display or print the release notes. You are strongly encouraged to read the release notes before proceeding with this installation.

The following example displays the command to invoke VMSINSTAL to install DECvoice from TK70 cassette tape drive XDELTA$MUA0: and the system response. This example uses the OPTION N release notes parameter.

```
$ SYS$UPDATE:VMSINSTAL VOX021 XDELTA$MUA0: OPTION N
VAX/VMS Software Product Installation Procedure V5.4-2

It is 01-SEP-1992 at  5:09
Enter a question mark (?) at any time for help.
```

If you do not supply either the product name or the device name, VMINSTAL prompts you for this information later in the installation procedure. VMSINSTAL does not prompt you for any options, so be sure to include OPTION N on the VMSINSTAL command line to access the release notes during the installation.

For more information

Refer to the *Developers Guide to VMSINSTAL*.

ivp_instr

The postinstallation section of the installation guide contains instructions for the installer to run the Installation Verification Procedure (IVP) separately.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OpenVMS	OSF/1	Sun	ULTRIX

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

The postinstallation section of the installation guide should include instructions for running the Installation Verification Procedure (IVP) separately from the installation. Include a statement explaining that the IVP is run separately to ensure the integrity of the installed files in case system problems occur.

Rationale

To provide the user with a way to verify the installation at any point after the installation completes.

Exceptions

None

Example

The following is an example of the statement regarding running the IVP separately:

OPENVMS

The Installation Verification Procedure (IVP) is usually run at installation. If you want to run the IVP separately to verify the integrity of the installed files, execute the following command procedure:

```
@SYS$TEST:ARA$IVP.COM
```

UNIX-BASED SYSTEMS

The Installation Verification Procedure (IVP) is usually run at installation. If you want to run the IVP separately to verify the integrity of the installed files, execute the following command procedure:

```
% setld -v [SUBSET_NAME]
```

For more information None

ivp_stat_msg

The Installation Verification Procedure (IVP) outputs successful and unsuccessful messages to show the IVP results clearly.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OpenVMS OSF/1 Sun ULTRIX
PERFORMED BY:	Software Engineer
REQUIREMENTS AID(S):	None

Description

Status Messages

Status comments should be displayed while the Installation Verification Procedure (IVP) is executing. For example, a description of the IVP tests being performed can be printed out for the customer as well as comments (such as "Linking of test program has begun") when the IVP enters the corresponding phase.

Starting and Ending Messages

The IVP must display messages that identify the start and completion of the IVP. For online identification purposes, the message must include the layered product name and version number, for example:

```
Beginning the DEC QUALITY for ULTRIX V2.1 Installation Verification Procedure
End of the DEC QUALITY for ULTRIX V2.1 Installation Verification Procedure
```

Success and Failure Messages

The IVP must display a success or failure message upon completion of the IVP. The message must include the layered product name and version number for online identification purposes.

Error and Warning Messages

If the IVP does contain error/warning messages, they must be documented in the installation guide and possible corrective actions should be clearly identified.

Rationale

Status messages allow the customer to observe what is going on during an IVP, and they can act as a vehicle of assurance that the product is working. Software Quality Operations and Information Services (SQOIS) strongly discourages displaying known error/warning messages during the IVP because they can lower the user's confidence that the installation completed successfully.

Exceptions

None

Example

See the examples provided in the description section of this requirement.

For more information None

keyboard

The product provides support, or documents nonsupport, for non-Digital Equipment Corporation keyboard KEYSYMs.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OpenVMS OSF/1 Sun ULTRIX
PERFORMED BY:	Software Engineer
REQUIREMENTS AID(S):	None

Description

All client applications must provide support for servers using non-Digital keyboard KEYSYMs. Some servers, such as the PC or Macintosh, may emulate a Digital keyboard, although the protocol does not require that of the server. The emulation in the server may confuse or conflict with the DECwindows application of key events.

Rationale

If a windowing product makes use of function keys as accelerators, such as Do, Insert Here, and F7, which are not available on a non-Digital keyboard, the functionality mapped to these keys is not present.

Exceptions

Complex operations that are unavailable on non-Digital keyboards must be documented in the user's guide.

Example

Not applicable

For more information

For more information on using keyboard KEYSYMs, refer to *X Window System*, Part I, Section 7.9, "Keyboard Encoding."

kitinstal_com

The KITINSTAL.COM command procedure should be written in such a way that the input from the installer is minimized. All required input should be prompted for at the beginning of the installation procedure.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

The installation procedure should display the message "No further questions" after all questions have been asked.

Prompting for user input should be kept to a minimum. Reprompting for information already given, or available through automated detection, should be avoided. Additionally, the installation path should be structured so that any questions dependent upon a previous installation question are not asked when the response is negative. In addition, all questions must include the recommended default answer.

Rationale

This allows the users to start the installation, supply all needed information to the installation procedure, and then leave the installation to be completed without further user intervention. It also promotes consistent, uniform, and user-friendly installations.

Exceptions

None

Example

```
%VMSINSTAL-I-RESTORE, Restoring product saveset A...
%VMSINSTAL-I-REMOVED, Product's release notes have been moved to SYS$HELP
$ Does this product have an authorization key registered and loaded (Y/N) ?
$ Would you like to purge files replaced by the installation (Y/N) ?
$ Where will files be placed? [SYS$SYSDEVICE:]
$ No further questions will be asked by the installation procedure
```

.
.
.

For more information

Refer to *DEC STD 041-0 Customer Installability Product*. To obtain a copy from the VTX data base, enter VTX SMC at the system prompt and follow the instructions displayed on the screen.

kitinstal_dcl_use

The installation procedure uses callbacks where available; direct use of Digital Command Language (DCL) and OpenVMS utilities should be avoided where possible.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): REQ_CHECK

Description

Whenever possible, callbacks should be used. Examples of VMSINSTAL callbacks that replace DCL commands are:

- SET ACL
- SET ASK_CASE
- DELETE_FILE
- SET PURGE
- FIND_FILE
- RENAME_FILE
- SECURE_FILE

Purging Files

The installation procedure should never automatically purge or delete files from a previous installation. The installer should be prompted to specify whether or not files replaced during the installation should be purged. This is accomplished with the SET PURGE option. SET PURGE will not work for files that have been renamed since the previous version of a product.

Do not include duplicate files, or different versions of the same file, in the software kit. If there are two files of the same name in the kit, then VMSINSTAL fails if the duplicate file is restored into VMI$KWD while the other copy is still there. The other copy will still be there if the installation is performed in safety mode, and also in non-safety mode if the installation procedure has not moved the file.

If renamed files from previous versions need to be removed they should not be searched for with the FIND_FILE callback.

If an entire previous version of a product is to be deleted, including files created by a user for that version, a separate command procedure should be provided. Instructions for using this procedure should be documented in a postinstallation section of the Installation Guide. This provides the installer with a sure and secured means of deleting the previous version of the product.

Rationale

The use of callbacks increases the integrity of the installation and insulates the layered product from future change to OpenVMS. If callbacks are used, critical operations, such as file deletion or replacement on a system disk, are deferred until the final stage of VMSINSTAL. If a fatal problem occurred before the final stage, VMSINSTAL would perform the appropriate cleanup operations and make the system as it was before the start of the installation.

Exceptions

Using DCL commands to handle files is allowed if the target directory is a kit working directory (VMS$KWD:). You can also use the DCL COPY command when you use product-specific callbacks (VMS$CALLBACK PRODUCT) to copy your own callbacks from VMS$KWD to the SYS$UPDATE: directory.

Actions required to install or set up the product that are not provided by a suitable VMSINSTAL callback can be executed with DCL commands.

Example

Not applicable

For more information See also the *Developers Guide to VMSINSTAL*.

kitinstal_driv_loc

The installation procedure places all drivers in the [SYS$LDR] directory.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

For systems running OpenVMS V5.0 or later, place drivers in [SYS$LDR] and remove explicit references to [SYSEXE] if it is loaded through SYSGEN in STARTUP. System directories should be referenced using VMI$ROOT or VMI$SPECIFIC.

Rationale

For OpenVMS V5.0 or higher the operating system expects to find the drivers in [SYS$LDR], but for earlier versions of OpenVMS, drivers are expected to be found in [SYSEXE].

Exceptions

For systems running earlier OpenVMS versions, place the drivers in [SYSEXE].

Example

Not applicable

For more information Also refer to requirement callback_use .

kitinstal_gbl_values

The installation procedure checks for the minimum values of free GBLSECTIONS and GBLPAGES.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

With the advent of OpenVMS V5.0, layered products requiring minimum values for global pages and global sections must check for the required amount in KITINSTAL.COM.

Use the GET_SYSTEM_PARAMETER callback with the parameters FREE_GBLSECTS, FREE_GBLPAGES, and CONTIG_GBLPAGES to calculate the existing values for global sections and global pages.

Rationale

To ensure the system has enough GBLSECTIONS and GBLPAGES to run the layered product.

Exceptions

If the product does not make use of GBLSECTIONS or GBLPAGES, then it doesn't need to make the checks.

Example

```
$ VMS$CALLBACK GET_SYSTEM_PARAMETER VAXFMS$FREE_GBLPAGES FREE_GBLPAGES
```

For more information See also the *Developers Guide to VMSINSTAL*.

kitinstal_in_proc

The first two commands contained in the installation procedure are ON WARNING and ON CONTROL_Y.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

The first two commands contained in the installation procedure (KITINSTAL.COM) are ON CONTROL_Y and an ON WARNING. All command procedure files called by KITINSTAL.COM must also have an ON CONTROL_Y and ON WARNING command.

Rationale

CONTROL_Y performs housekeeping chores, returning a warning status that executes the ON WARNING statement. If this statement is omitted, and the installer enters Ctrl/Y, the results will be unpredictable.

Exceptions

If you need a different level of severity to control, ON ERROR or ON SEVERE ERROR may be used instead of ON WARNING.

Example

```
$ ON CONTROL_Y THEN VMI$CALLBACK CONTROL_Y
$ ON WARNING THEN EXIT $STATUS
```

For more information See also the *OpenVMS DCL Dictionary* and the *Developers Guide to VMSINSTAL*.

kitinstal_ivp_stat

The command procedure KITINSTAL.COM exits to VMSINSTAL.COM indicating the status of the Installation Verification Procedure (IVP).

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

All KITINSTAL.COM procedures exit back to VMSINSTAL.COM with a status after executing the Installation Verification Procedure (IVP). The status must be either VMI$_SUCCESS (if the IVP was successful) or VMI$_FAILURE (if the IVP failed). Do not include the VMI$ symbols as part of the IVP code because this would cause a failure when the IVP is run outside of VMSINSTAL.

Rationale

The IVP status allows the user to know if the installation was successful or not.

Exceptions

None

Example

```
$ EXIT VMI$_SUCCESS
```

For more information See also the *Developers Guide to VMSINSTAL*.

kitinstal_process_check

The installation procedure checks for the minimum required process quotas using the F$GETJPI function.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

Occasionally a product installation may depend on a process being started up or shut down. A check to determine the state of the process must be performed before continuing the installation.

Rationale

To ensure that the required minimum process quotas required for successful installation are met.

Exceptions

None

Example

```
$ NSDS$PAGE_FILE_QUOTA = F$GETJPI("","PGFLQUOTA")
```

For more information

This section points to further sources of information discussed in this requirement.

See the *OpenVMS DCL Dictionary*.

kitinstal_program_check

The installation procedure checks for non-default system parameter values using the GET_SYSTEM_PARAMETER callback.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

If the layered product requires nonstandard system quotas and limits, you must check these quotas to ensure that at least the minimum values are set on the system before installing the product. Check system parameters by using the GET_SYSTEM_PARAMETER callback. This callback can be used with the Table Driven option, OPTION T.

Rationale

Ensures that the minimum system quotas and limits are met to run the product.

Exceptions

The product uses only default system parameters.

Example

```
$ VMI$CALLBACK GET_SYSTEM_PARAMETER TST_MAX WSMAX
```

For more information See also the *Developers Guide to VMSINSTAL*.

kitinstal_set_safety

The installation procedure does not disable the SET SAFETY callback.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

The SET SAFETY option establishes the safety level of the installation, that is, the installation's ability to recover from a system failure. Since implementation of the safety feature requires a higher peak utilization of disk space, parameters should be selected only after weighing the need for installation safety against anticipated space availability.

The SET SAFETY option also records the level of safety in the marker file to establish appropriate crash recovery procedures.

Note the following:

- The use of SET SAFETY NO is strongly discouraged.
- If changing the safety mode cannot be avoided, SET SAFETY CONDITIONAL should be used.
- If SET SAFETY NO cannot be avoided, its use should be reported and justified to SQOIS .

Rationale

The SET SAFETY callback provides a recovery mechanism if an unexpected event occur, such as a system crash or an aborted installation. If you omit this callback, VMSINSTAL automatically specifies safety mode unconditionally, that is, without regard to available disk space. If the system does not have sufficient disk space to support it, the installation terminates with failure status.

Exceptions

None, unless justified to SQOIS .

Example

```
$ VMI$CALLBACK SET SAFETY CONDITIONAL 1000
```

For more information

Refer to requirement instal_space_check for information on a related requirement.

See also the *Developers Guide to VMSINSTAL*. Copies can be obtained from: {VIRTUE, SQMUK, SQMJPN}::SYS$INFO:DEC_VMSINSTAL_AUG90.*

kitinstal_startup

The installation procedure executes the SET STARTUP callback before invoking the Installation Verification Procedure (IVP) if the product needs to be started before running the IVP.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Layered products supplying a startup command procedure must use the SET STARTUP callback.

Rationale

The use of the SET STARTUP callback ensures that the command procedure is executed before the Installation Verification Procedure (IVP).

Exceptions

This requirement need not be complied with if the product does not need to be started before running the IVP.

Example

```
$ VMI$CALLBACK SET STARTUP CHECKTRANS$STARTUP ""
$ !
$ ! Move data file
$ !
$ VMI$CALLBACK PROVIDE_FILE CHECKTRAN_DATA CHECKTRAN.DAT VMI$ROOT"[SYSEXE]
$ !
$ ! Installation completed, exit
$ !
$ EXIT VMI$_SUCCESS
$ !
$ ! Verify installation
$ !
$ CHECKTRANS$IVP:
$ !
$ ! Run the IVP
$ !
$ @VMI$ROOT:[SYSTEXT]CHEKTRAN$IVP.COM
```

For more information

See the *Developers Guide to VMSINSTAL*. Copies can be obtained from: {VIRTUE, SQMUK, SQMJPN}::SYS$INFO:DEC_VMSINSTAL_AUG90.*

kitinstal_sys_startup

The layered product startup procedure is placed under the SYS$COMMON:[SYS$STARTUP] directory.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

For systems running VMS V5.0 or higher, place the startup file(s) in the SYS$COMMON:[SYS$STARTUP] directory. VMI$ROOT: should be used in the STARTUP callback.

Rationale

This provides a central directory for all layered products to place their startup command procedures, which provides consistency among all Digital products.

Exceptions

Systems running versions prior to VMS V5.0 where the startup files should be placed in SYS$MANAGER.

Example

```
[SYS0.SYSCOMMON.SYS$STARTUP]MCC_TCPIP_STARTUP.COM
```

For more information

See also the *Developers Guide to VMSINSTAL*. Copies can be obtained from: {VIRTUE, SQMUK, SQMJPN}::SYS$INFO:DEC_VMSINSTAL_AUG90.*

kitinstal_v_check

The installation procedure checks for the required OpenVMS and Alpha AXP versions as listed in the System Support Addendum (SSA) and the installation guide, using the CHECK_VMS_VERSION callback.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

The KITINSTAL procedure must perform an OpenVMS version check to make sure that the product is being installed on a supported version of the operating system. Layered products supporting OpenVMS V5.0 and higher must use the CHECK_VMS_VERSION callback to check for the minimum OpenVMS version supported. This callback also accepts a maximum version.

To perform OpenVMS version checking, layered-product groups must define a minimum OpenVMS version to be supported. This version must be consistent with the minimum version listed in the SSA and the installation guide.

Rationale

The use of this callback limits the installation of a product to specified versions of the OpenVMS operating system.

Exceptions

None

Example

```
$ VMS$CALLBACK CHECK_VMS_VERSION FOR_VERSION 5.0 "" 5.2
```

For more information

See also the *Developers Guide to VMSINSTAL*. Copies can be obtained from: {VIRTUE, SQMUK, SQMJPN}::SYS$INFO:DEC_VMSINSTAL_AUG90.*

kitinstal_vmi_logicals

The logical VMI$ROOT is used to reference all system directories.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

All references to system directories must use the logical name VMI$ROOT: and an explicit directory (for example, VMI$ROOT:[SYSMGR]) rather than SYS$ logicals such as SYS$MANAGER. The use of VMI$ROOT handles any current and future OpenVMS configurations; reliance on SYS$ logicals cannot guarantee future success.

Rationale

VMI$ logicals should be used because VMSINSTAL automatically handles any legal OpenVMS configuration, including a redefinition of system search lists to provide correct file placement and access. Also, the VMI$ROOT logical contains the correct alternate root name if the alternate root (R) option has been chosen with VMSINSTAL.

SYS$ logicals are not a proper choice because they are search lists, and layered product file placement may be performed incorrectly on common system disks.

Exceptions

System logicals (SYS$nnn) are used only for file referencing when VMI$ logicals cannot be used.

Example

```
$ VMI$ROOT:[SYSMGR]
```

For more information

See also the *Developers Guide to VMSINSTAL*. Copies can be obtained from: {VIRTUE, SQMUK, SQMJPN}::SYS$INFO:DEC_VMSINSTAL_AUG90.*

legal_notice_runtime

Legal notices must be displayed at run time.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): *DEC STD 197-0*

Description

All the standard notices (copyright, government data rights, and license notices) listed in Section 2.3.2 of *DEC STD 197-0 Legal Requirements and Guidelines for Digital Publications and Software* must be displayed at run time.

Rationale

Use of appropriate legal notices and labels protects the security and ownership of Digital's intellectual property.

Exceptions

Exceptions to this requirement are listed in *DEC STD 197-0*.

Example

Refer to *DEC STD 197-0* for a list of legal notices that must be displayed at run time.

For more information

Refer to *DEC STD 197-0* for a list of legal notices that must be displayed at run time.

legal_notices_windows

Standard notices must be displayed in a windowing environment.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): *DEC STD 197-0*

Description

In most windowing systems there is a menu item that will display legal notices. For example, in Motif, legal notices are displayed by selecting the On Version item under the Help menu and in Microsoft Windows, legal notices are displayed by selecting the About item under the Help menu.

All of the standard legal notices (copyright, license, and government data rights) must be displayed when the user selects a menu item that will display legal notices.

In addition, the first line of the copyright notice must be included in the title bar that is displayed when the application is started. The notice must be displayed until a user action occurs. If there is not sufficient room for the entire notice, it may be abbreviated as follows:

```
© Digital Equipment Corp.  199N. All rights reserved.
```

or

```
©  DEC  199N. All rights reserved.
```

Rationale

Use of appropriate legal notices and labels protects the security and ownership of Digital's intellectual property.

Exceptions

None

Example

Not applicable

For more information

For more information about where to place copyright information, consult the appropriate style guide for a particular windowing environment. Refer to section 2.3.2e in *DEC STD 197-0 Legal Requirements and Guidelines for Digital Publications and Software* for the examples legal notices that must be displayed in a windowing environment.

lmf_descr

The preinstallation section of the installation guide contains a brief description of Licensing Management Facility (LMF).

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OpenVMS OSF/1 Sun ULTRIX
PERFORMED BY:	Technical Writer
REQUIREMENTS AID(S):	None

Description

The preinstallation section of the installation guide must contain only a brief description of the LMF. See the *Software Licensing and License Management Facility Guide* for complete details. Refer to the For More Information section at the end of this requirement description for information on how to obtain a copy.

OpenVMS Specifics

The need to register and load the license on the system before the installation must be stated and instructions included. The following two methods for registering and loading the license should be stated:

- Invoke the SYS$UPDATE:VMSLICENSE.COM procedure and when prompted for information, respond with the data from your license Product Authorization Key (PAK).
- Enter the LICENSE REGISTER command at the Digital Command Language (DCL) prompt with the qualifiers that correspond to your license PAK information. Then enter the LICENSE LOAD command with the appropriate qualifiers.

Some products bundle other software and incorporate Component Product Authorization Key (CPAK) handling in their KITINSTAL procedure. Note that CPAKs are being phased out and can only be used with special approval. Any product that ships a CPAK in their software *must* contact MRKTNG::SNPC for approval. This approval must occur at every iteration of the product. If your product has been approved for CPAK use, the installation guide should include a statement of use and an explanation of what is being implemented.

ULTRIX Specifics

A License Product Authorization Key (License PAK) must be registered in the License Data base (LDB) to use your product on a newly-licensed node. Users should be given the following information for registering and loading a license:

- To register a license under the ULTRIX system, first log in as superuser.
- The registration procedure followed depends on whether the Installation Verification Procedure (IVP) is automatically run during the installation of your product. Detailed steps, as outlined in the installation guide template, should be provided for the appropriate situation.
- After the license is registered, the user should be instructed to use the LMF RESET command to copy the license details from the License Data base (LDB) to the kernel cache.

Rationale

This serves to remind the user that a PAK is needed prior to installing and/or running the product.

Exceptions

None

Example

Not applicable

For more information

To obtain a copy of the *Software Licensing and License Management Facility Guide*, copy the following file: MRKTNG::SNPC.

To obtain a copy of the ULTRIX installation guide template, copy the following file:
BOOKIE::PUBLIC:[PUBLIC]INSTALL_GD_OSF.SDML

To obtain a copy of the VMS installation guide template, copy the following file:
BOOKIE::PUBLIC:[PUBLIC]INSTALL_GD_VMS.SDML

To obtain a copy of the Sun installation guide template, copy the following file:
BOOKIE::PUBLIC:[PUBLIC]INSTALL_GD_SUN.SDML

lmf_pak_install

Licensing Management Facility (LMF) Product Authorization Keys (PAKs) that are intended for distribution to customers should be installation tested by the development group prior to distribution of the PAK to the customer. Testing should include attempts to functionally execute the product with the PAK applied and with it deleted.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OpenVMS	OSF/1	ULTRIX
PERFORMED BY:	Software Engineer		
REQUIREMENTS AID(S):	None		

Description

Licensing Management Facility (LMF) Product Authorization keys (PAKs) that are intended for distribution to customers should be installation tested by the development group prior to distribution of the PAK to the customer. Testing should include attempts to functionally execute the product with the PAK applied and with it deleted.

This testing is applicable for all cases where a PAK will be distributed to customers, and it covers both field test and revenue shipment situations. Testing is applicable on all platforms supporting LMF.

Rationale

SQOIS has recently detected a number of defective LMF PAKs that were the final PAKs intended for the customer. From our testing, it is clear that the engineering group developing the product must not have tested the product with the final PAK. With Alpha AXP(tm), the business rules for PAKs have become more complex, supporting more opportunity for error in PAK creation, therefore, it makes sense that the development group has tested the final PAK intended for customer use before it is made available to the customer.

Note

There is also awareness in some quarters of the need to consider enhancing the PAK generation system to more robustly support evolution of licensing business requirements. If implemented, these enhancements

could proactively prevent defects in PAK creation; however, no date has been set for implementation.

Exceptions

None

Example

None

For more information

MRKTNG::SNPC
Send mail to contact the software New Products Committee

mail_commands

The installation procedure does not include any commands to send electronic mail to the development group.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

While mail commands are occasionally included in field test software kits as a way to monitor installations, they should never be included in Software Supply Business (SSB) software kits.

Rationale

The development group is not the appropriate contact for customer problems and questions. Additionally, since the development group is probably not accessible via electronic mail from the customer's network, attempts to send mail would result in an error.

Exceptions

Internal field test software is allowed to include mail commands in its installation procedure.

Example

Not applicable

For more information None

mouse

Product functionality is available from a Xserver using one-button or two-button mice, or when there is no mouse available.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac		

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Product functionality is available from an Xserver using one-button or two-button mice, or keyboard, when there is no mouse available.

Rationale

In a distributed, heterogeneous environment, a user may attempt to use an application from a display server that has only a one-button or two-button mouse.

Exceptions

When no mouse is available, as with keyboard independence, missing functionality must be documented.

Example

DECwrite is an example of a product that provides common functions for different mice. See the *DECwrite User's Guide*.

For more information

If You Need	Do the Following
Information on mouse and keyboard functionality	Refer to the *OSF/Motif Programmer's Guide*, Section 11.1, "Getting User Input."

If You Need	Do the Following
Information on mouse button bindings	Refer to the *OSF/Motif Programmer's Guide*, Section 14.5, "Mouse Button Bindings."

PATHWORKS for SCO UNIX API Programmer's Reference	AA-PQEMA-TE
Microsoft Windows Software Development Kit for MS–DOS and PC-DOS Operating System Tools (or the appropriate programing guide)	Third-Party (get from any bookstore)

multi_display

The product displays to a selected screen of multidisplay systems.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OpenVMS	OSF/1	Sun	ULTRIX

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Windowing applications must display to the selected windowing terminal of a multi-display system as defined by a user's environment, or as defined at execution time by the user.

The windowing application must not make assumptions about screen numbers by displaying on the primary display.

Rationale

Customers expect to be able to display to the display screen they have selected.

Exceptions

None

Example

Not applicable

For more information None

non_std_sys_params

The preinstallation section of the installation guide lists any nonstandard system parameter requirements with specific values or mathematical formulas for calculating the required values.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

The preinstallation section should list any nonstandard system parameter requirements. All requirements must contain an explicit value or a mathematical formula used to calculate the required value. For example, it is not enough to state that "adequate global pages" are required; a specific value must be included. This section must also include instructions for checking the current system parameter values via the SYSGEN utility, changing dynamic system parameters via the SYSGEN utility, and changing nondynamic parameters by editing SYS$SYSTEM:MODPARAMS.DAT and invoking SYS$UPDATE:AUTOGEN.COM.

To change a parameter value already listed in the MODPARAMS.DAT file, delete the current value and insert the new value.

To add a new parameter value, add a line to MODPARAMS.DAT that includes the parameter name and the new value (for example, WSMAX = 1024).

To modify incremental parameters, such as GBLPAGES and GBLSECTIONS, use ADD_. For example, to increment GBLPAGES by 2000, include the following line to MODPARAMS.DAT: ADD_GBLPAGES = 2000

If there is a requirement for global pages and/or global sections, the following method for determining AVAILABLE global pages and global sections should be stated (for OpenVMS V5.2 and later):

```
WRITE SYS$OUTPUT F$GETSYI("CONTIG_GBLPAGES")

WRITE SYS$OUTPUT F$GETSYI("FREE_GBLSECTS")
```

Rationale

Allows the user to plan any changes to the existing system parameters, which is important because changing nondynamic system parameters requires that the system be rebooted for them to take effect.

Also, the user can assess any impact on system performance caused by installing the product.

Exceptions

None

Example

```
WSMAX must have a value of at least 16400
```

The above example indicates that WSMAX must have a value of at least 16400.

```
$ RUN SYS$SYSTEM:SYSGEN

SYSGEN>SHOW WSMAX
Param Current Default Min Max Unit Dynamic
WSMAX 14800 1024 60 200000 Pages
SYSGEN>EXIT
```

The example above shows how to check the present value of WSMAX.

```
Edit SYS$MANAGER:MODPARAMS.DAT to include WSMAX=16400
run AUTOGEN
@SYS$UPDATE:AUTOGEN GETDATA REBOOT
```

This example illustrates how to change the present value of WSMAX

For more information None

nonsys_acct_instl

The installation allows for product installation from accounts other than the SYSTEM account.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

There are two reasons customers may require the ability to install from a non-SYSTEM account:

- The SYSTEM account may be set /NOINTERACTIVE, with privileges assigned to alternate accounts, to avoid the problem of having a well-known account on a system, where one can guess the password to log in. A SYSTEM account may only be used to establish the context for certain processes, such as OPCOM and Job Controller. Therefore, installations must be supported from non-SYSTEM accounts.
- System management at a site may rotate or may be done by groups of people who all install products. To avoid having many people access one system account, separate user accounts may be established, one for each system manager. Installations would then be done from non-SYSTEM accounts.

Note

If the files need to be given the SYSTEM UIC, the installation procedure should determine the SYSTEM UIC and assign it to the files. It should not be necessary for the user to either be forced to use the system account or to change file ownership after the installation.

Rationale

For security reasons, layered products must support installation from suitably privileged accounts other than the SYSTEM account.

Exceptions

None

Example

Not applicable

For more information

None

online_help

Online help is available to the user during product installation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

If an installation question is complex, the user must be able to request help during product installation. The user should then be presented with a description of expected input.

Rationale

Providing online help during an installation makes it easier for the user to get immediate information if there is a problem.

Exceptions

Simple, straightforward questions need not have help information associated with them.

Example

```
Would you like to run the IVP during the installation?  ?

This kit contains an Installation Verification Procedure (IVP) to verify the
correct installation of DEC QUALITY for ULTRIX.

Would you like to run the IVP during the installation?
```

In this example, the installation asks if the user wants to run the IVP. The user entered a question mark to prompt for an explantion of the question. After the help information is displayed, the initial question is repeated.

For more information None

online_man_pages

Online manual pages are included with product installation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OSF/1	Sun	
ULTRIX	SCO UNIX	AIX
HP-UX		

PERFORMED BY: Software Engineer or Technical Writer

REQUIREMENTS AID(S): REQ_CHECK

Description

Manual pages are required for all UNIX-based layered products. For example, such things as executable files and scripts must have manual pages describing what they do and how to properly use them. Note that optional subsets are required for a user to use the man pages.

Rationale

Man pages represent online help for a product in the form of a detailed description for the command or file. It is important to provide as much immediately available online help to a user as possible.

Exceptions

None

Example

Not applicable

For more information

For more information on manual pages, refer to the software development manuals for the relevant operating system or execute a "man man" command on the system you are using.

online_rn

It is recommended that release notes be available online.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer or Project Leader

REQUIREMENTS AID(S): None

OpenVMS

The following sections are specific to OpenVMS layered products.

Description

Online release notes are required for all OpenVMS products. They are included in the kit media. If appropriate, you can choose to distribute a hardcopy of the release notes with the product.

The VMSINSTAL print release notes option (Option N) provides the ability to print the online release notes at the terminal (and on SYS$PRINT if desired) before or instead of installing the product. However, the installation does not have to be completed nor do all product files have to be restored from the kit. The installer has the option to continue or abort the installation after printing the release notes.

Online Release Notes Filename

Provide the online release notes as a machine-readable, printable file, which follows this filenaming convention:

```
facvvu.RELEASE_NOTES
```

where:

- "fac" is the facility mnemonic of the product registered with the Product Registrar
- "vvu" is the version number

For example:

```
FORT042.RELEASE_NOTES
```

where "FORT" is the facility mnemonic for VAX FORTRAN and "042" refers to version 4.2.

This format permits VMSINSTAL to construct the name of the release notes file so it can be restored and copied to the SYS$HELP directory. This is done without adding a callback statement to KITINSTAL to provide the name of the release notes file.

A consistent filename is also necessary to allow VMSINSTAL to purge the release notes files if the installer chooses the purge option during the installation. In addition, if these files have similarly configured names, anyone can get a listing of all release notes files in SYS$HELP by setting default to SYS$HELP and entering $DIR *.REL*, or they can get a listing of all files for a product by entering $DIR fac*.* Again,"fac" represents the registered facility mnemonic for the product.

Release Notes in KITINSTAL

The text of the release notes should not be part of the product's help files. Instead, each product should include the release notes file with the KITINSTAL procedure, and any other command procedures invoked by KITINSTAL, on Saveset A.

This permits the user to view and print the information at the beginning of the installation.

Help Library

If you deem it necessary, your product's Help library can also contain *selected* release notes topics. However, the entire text of the release notes should still be supplied in a separate file.

Your product's Help file should contain at least one reference that points to the release notes file, for example:

```
$ HELP Product_name RELEASE_NOTES

Release notes for product_name are contained in this file:
 SYS$HELP:facvvu.RELEASE_NOTES

You can type or print this file to read the release notes information
```

The product-specific Help file could also include hints for using the DIRECTORY command to find the file (and earlier versions of it).

To eliminate the need to reload the Help library with each point release, text for help on release notes can be written generally, that is, without specific reference to version number.

Rationale

Having release notes available online is beneficial to the customers because the notes are readily available. They are also useful for layered product groups, since they can be modified until just before the Software Supply Business (SSB) submission date.

Exceptions

None

Example

Not applicable

UNIX

The following sections are specific to UNIX layered products.

Description

Online release notes are desirable but are not required at this time. Products providing online release notes are not required to provide hardcopy release notes. Layered product groups have the option of providing both online and hardcopy release notes.

If online release notes are going to be supplied, then ensure the following:

1. Online release notes are a separate, installable entity (that is, the end user is provided with a Release Notes option during installation). This allows the user to install and read the release notes before the rest of the product is installed.
2. Online release notes are placed in the same subdirectory as the layered product files.
3. Instructions for displaying and printing the release notes should be contained in the cover letter, the read-me-first letter, or the installation guide.

Rationale

Compliance with this requirement makes the release notes easily available to users who may not have easy access to the hardcopy document and to users of UNIX ConDist for ULTRIX layered products.

Exceptions

None

Example

Not applicable

PC

Release notes are contained in the read.me file.

For more information

None

optional_prod_comp_instl

The installation of optional product components at a later date should not require the reinstallation of the product.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

A product may include components that can be optionally added to the system either at initial installation or at a later date. In the latter case, the installation procedure should ensure that these optional product components can be added without necessitating the reinstallation of the entire product. Examples of optional product components are:

Demo programs
Example files
Client software for various platforms
Help libraries
Several product savesets bundled under one software entity

Rationale

To enable customers to add extra product components without the need to reinstall the entire product, which may be both time-consuming and irritating.

Exceptions

Products that require no installation of optional product components.

Example

Not applicable

For more information

Also refer to *DEC STD 041-0 Customer Installability Product*. This standard is available from the Corporate VTX data base. At the system prompt, enter: VTX SMC [Return]. Follow the instructions to obtain a copy.

pathworks_env_var

PC applications that work in the PATHWORKS environment should use environment variables instead of hardcoded device/pathnames in their installation procedure.

OPERATING SYSTEMS (ALL ARCHITECTURES):

MS–DOS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

PC applications that work in the PATHWORKS environment should use environment variables instead of hardcoded device/pathnames in their installation procedure.

Use of environment variables allows network administrators to set up PCs to use USE /ENV.

Rationale

Use of a hardcoded device/pathname name restricts PC clients to connect using that same device. Where many clients exist, and multiple servers, it is probable that any chosen device for a new application server will already be in use by at least one server, thus creating a conflict for a client that wished to connect to both servers.

Use of an environment variable would allow the PC client to connect any available drive to the file server where the desired application is installed.

Exceptions

None

Example

None

For more information

Ref to PATHWORKS documentation references in Subhead 1.4 in the PC TRG checklist (EL-CP790-01)

See Topic 5073 in notes file RANGER::PWDOS4

pc_client_server_security

PC applications that work in a client/server environment should ensure the security of users data. User specific files, including temporary files, should not be accessible by other clients.

OPERATING SYSTEMS (ALL ARCHITECTURES):

MS–DOS Windows NT Windows

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

None

Rationale

Applications need to ensure that they do not create any security risks.

Exceptions

None

Example

If temporary files are placed in a special STAGING area, these should be protected so that they don't create any potential security risk.

For more information

Ref to PATHWORKS documentation references in Subhead 1.4 in the PC TRG checklist (EL-CP790-01)

See Topic 5073 in notes file RANGER::PWDOS4

pc_network

Installation procedures of PC products should allow the product to be installed in a networked environment.

OPERATING SYSTEMS (ALL ARCHITECTURES):

MS–DOS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

When a product is installed on a PC in a PATHWORKS environment it is possible to install the product on an available disk or file service. The installation procedure should therefore offer the choice of the location where the installation will take place.

Rationale

Restriction of the installation to a physical place, such as the C drive on a PC, limits the interoperability that is expected between products in a PATHWORKS networked environment.

Exceptions

None

Example

None

For more information None

pc_revenue_prot

PC products should include an appropriate revenue protection scheme.

OPERATING SYSTEMS (ALL ARCHITECTURES):

MS–DOS Windows NT Windows Mac

PERFORMED BY: Development Group/Product Management
REQUIREMENTS AID(S): None

Description

Currently there is no recognized method for implementing a revenue protection scheme for products that install on MS–DOS. This is especially serious for products that can be installed on a central server and be available to any client on the network.

A proposed minimum revenue protection scheme is to implement the method used by Microsoft, where at installation a question is asked for a registered owner and Organisation of the software being installed. This is written to the installation media for future reference and to act as a deterrent against future illegal copying of the software.

For software that can exist on a Network server, a standard revenue protection scheme is still required. Ideally, standard software would limit the use of the software to the number of licenses bought. This software does not exist today, so products should document how users can remain "legal" in their installation guides by limiting access to a network service to the number of licenses bought.

Rationale

Presently there is no Digital standard for protecting the revenue for PC software.

Exceptions

When/if software is provided on a CD-ROM, a revenue protection scheme that/if does not involve writing to the CD-ROM will be required.

This will also be true if the media is a read only service on a network server.

Example

None

For more information

None

pc_temp_fil

Temporary files must be dealt with correctly so that they are deleted when they are no longer useful.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	MS–DOS	Windows	Windows NT
PERFORMED BY:	Software Engineer		
REQUIREMENTS AID(S):	None		

Description

It is important for PC users that they do not suffer from temporary files being created that are not automatically deleted by the system when no longer required.

Products should therefore be organised about where they place these files and that they are deleted.

Products should also place temporary files in the location stated in the AUTOEXEC.BAT by the SET TEMP command where this is defined. (Providing this does not create any security problem.)

Rationale

It is unprofessional to write software that puts temporary files all over a person's file system and then leaves them there in perpetuity. Users do not wish to fill their disks with temporary files.

Exceptions

None

Example

None

For more information None

performance_design_impl

The project leader is responsible for designating individuals to focus on performance. The project team is responsible for tracking the performance status/issues of the product in the design and implementation phases.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Product Team/Product Manager

REQUIREMENTS AID(S): None

Description

The project leader should ensure that there is a plan of action for the design and implementation of performance in the product. The project leader is responsible for designating an individual(s) to focus on performance. The project team is responsible for tracking the performance status/issues in the design and implementation.

The product team should conduct performance design review(s) to examine performance hazards and trade-offs, which could be done using performance design reviews and/or performance prototype testing. To assist in the support of performance goals, the product team might want to consider providing performance knobs, and so forth, and/or monitoring capabilities within their product.

The product team tracks, measures, and compares the performance from BL to BL against the goals, (that is, throughput, response time, and resource utilization).

The product manager tracks the performance of the competition.

Rationale

Performance and price/performance ratio are critical user apparent features for most products. Integrating performance work into design implementation and tracking performance are the keys to success.

Exceptions

None

Example

TPSYS::SW_PERFORMANCE:[PRODUCTS.TRG.DESIGN_AND_IMPLEMENTATION]

For more information

Send Mail to: TPSYS::SW_PERF

TPSYS::SW_PERFORMANCE:[HANDBOOK], TPSYS::SW_PERFORMANCE:[REPORTS], TPSYS::SW_PERFORMANCE:[TPC]

Software Performance Engineering, 3-day course offered regularly.

Eyal Zimran and David Butchart, "Performance Engineering Throughout the Product Life Cycle", DEC-TR 868.

Eyal Zimran, "Integrating Performance Engineering into the SQA Process", January 1993, Presentation.

Performance Notes files: TURRIS::DECSPEC, DATABS::DECTRACE_V11, NAC::NAC_PERFORMANCE, HDLITE::SPEC, CLT::DECSET, WORDS::ULTRIX_TOOLS HDLITE::ULTRIX_TUNING, TURRIS::PCA, BYTECH::VAXSPM, VMSDEV::VMSTUNING

performance_planning

If performance is critical to your product, performance goals and minimum shipment performance criteria have been established. Performance test and documentation plans have been created.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Product Manager/Product Team

REQUIREMENTS AID(S): None

Description

The product team should identify the competitors and their performance and collect performance requirements from potential customers; customers who are using an earlier version of the product; or customers who are using similar products. The product team defines workloads and identifies relevant benchmarks.

The product manager and project leader establish minimum shipment performance criteria. The product team establishes performance goals and creates performance testing and evaluation plans. The product team and technical writers create the performance documentation plan.

Rationale

Good performance contributes to high customer satisfaction, high profits, and access to large share of potential market. Early reputation of product's poor performance is costly and difficult to change.

Exceptions

None

Example

TPSYS::SW_PERFORMANCE:[PRODUCTS.TRG.PLANNING]

For more information

Send Mail to: TPSYS::SW_PERF

Software Performance Engineering, 3-day course offered regularly.

Eyal Zimran and David Butchart, "Performance Engineering Throughout the Product Life Cycle", DEC-TR 868.

Eyal Zimran and Jim Behymer, "Guidelines for Development of Performance Requirements, Goals and Test Plans", DEC-TR 869.

Eyal Zimran, "Integrating Performance Engineering into the SQA Process", January 1993, Presentation.

VTX PERFORMANCE

performance_release

The product team continues to track the performance of the released product and identifies areas for performance improvements.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Product Manager/Product Team

REQUIREMENTS AID(S): None

Description

The product team continues to track the performance of the released product, identify areas for improvements, and perform testing and analysis on additional hardware platforms. They also provide special tuning and recommendation to customers. The product team measure or estimate the impact of new releases of related software, operating system, or hardware products.

Rationale

Continuous improvement and understanding of customer requirements are the key to protect investment in product features, marketing, and performance.

Exceptions

None

Example

TPSYS::SW_PERFORMANCE:[PRODUCTS.TRG.RELEASE]

For more information

Send Mail to: TPSYS::SW_PERF

Software Performance Engineering, 3-day course offered regularly.

Eyal Zimran and David Butchart, "Performance Engineering Throughout the Product Life Cycle", DEC-TR 868.

Eyal Zimran, "Integrating Performance Engineering into the SQA Process", January 1993, Presentation.

performance_testing

The product team is responsible for conducting performance testing and characterization, and for assessing the minimum shipment performance criteria versus the planned criteria.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Product Manager/Product Team

REQUIREMENTS AID(S): None

Description

The product team conducts a performance test and characterization of the final version of the product. The following reports are generated (when relevant):

- Performance Specification Summary - a submission to the Software Performance Handbook Reference is encouraged (2-5 pages).
- Performance Characterization Report (5-50 pages).
- Performance Tuning and Management Guide (A book).
- Performance training material for field test, customer support and system sizing.
- Performance Marketing Report.

The product's performance should be tested on one of the processors from each supported processor class and on all supported operating systems (that is, VAX, AXP, MIPS, 486, OSF, ULTRIX, OPENVMS).

The product's performance should be tested with multiple versions of the prerequisite and/or optional layered products, and a sustained load performance test performed.

The product manager assesses the minimum shipment performance criteria versus the planned criteria. The product team documents performance tuning knobs/parameters, performance instrumentation, and monitoring capabilities.

Rationale

Performance characterization reports are necessary for system sizing, system integration projects, and to support customers.

Exceptions

None

Example

TPSYS::SW_PERFORMANCE:[PRODUCTS.TRG.TESTING]

For more information

Send Mail to: TPSYS::SW_PERF

TPSYS::SW_PERFORMANCE:[HANDBOOK], TPSYS::SW_PERFORMANCE:[REPORTS], TPSYS::SW_PERFORMANCE:[TPC]

Software Performance Engineering, 3-day course offered regularly.

Eyal Zimran and David Butchart, "Performance Engineering Throughout the Product Life Cycle", DEC-TR 868.

Eyal Zimran, "Integrating Performance Engineering into the SQA Process", January 1993, Presentation.

prereq_hw

The preinstallation section of the installation guide lists all prerequisite hardware.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

The preinstallation section of the installation guide must include all prerequisite hardware information. If the installation guide is not updated with each release of the product, a pointer to the System Support Addendum (SSA) should be included rather than specific hardware information. Also, it is sometimes difficult to maintain hardware support tables in the installation guide, so a pointer to the SSA is more efficient. The hardware information in this section must be consistent with the hardware information in the SSA.

Rationale

To provide the user before the actual installation begins with the information needed to perform a successful installation. This way, the user makes all necessary preparations before starting the installation.

Exceptions

None

Example

Required Hardware

A supported MicroVAX II, MicroVAX 3000 series, or a VAXstation II computer system. Refer to the MicroVAX/DRQ3B Device Driver SPD/SSA for a complete list of processors.

For more information Refer to Appendix C Reference Documents for pointers to Installation Guide templates.

prereq_info

The preinstallation section of the installation guide lists all prerequisite and optional software information, including supported versions, or it references the System Support Addendum (SSA) for supported versions.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

This section must include any prerequisite and optional layered products, as well as required operating systems. If the installation guide is not updated with every release of the product, specific versions of the prerequisite and optional products should not be listed. Instead, a pointer to the SSA should be included.

The prerequisite and optional information listed in the installation guide must be consistent with the prerequisite and optional software information listed in the SSA.

Rationale

To provide the user before the actual installation begins with the information needed to perform a successful installation. This way, the user makes all necessary preparations before starting the installation.

Exceptions

None

Example

To install DECforms you must have installed on your system:

- A valid OpenVMS operating system configuration, OpenVMS V5.0 or higher.
- CDD/Repository for OpenVMS/Plus V4.1 or higher (if you intend to store definitions in VAX SCC/Plus software).

- VAX Language Sensitive Editor V2.3 or higher (if you intend to use VAX Language Sensitive Editor templates for the DECforms Independent Form Description Language).

For a complete list of products that are compatible with this version of DECforms, refer to the SPD/SSA.

For more information

Refer to Appendix C Reference Documents for pointers to Installation Guide templates.

prereq_optional_prod_test_msg

The installation procedure checks for the minimum version of prerequisite and optional products as listed in the Software Product Description (SPD) and System Support Addendum (SSA), that is, operating systems versions numbers, processor class/architecture, and any prerequisite hardware.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer or Project Leader

REQUIREMENTS AID(S): None

Description

Installation procedures should check for the existance of prerequisite and optional products. If prerequisite products do not exist, a message should be displayed indicating the legal product name, minimum version, and binary name, that is, zip file, tar file, subset, and so on, needed for successful execution of the product. The installation should abort at this time.

A prerequisite product is one that is required for the operation of the layered product that is being installed. An optional product is one that can integrate with the layered product that is being installed, but is not required for the layered product's successful operation.

Rationale

Failure to provide this information could produce unpredictable results for the remainder of the installation and at execution of the product.

Exceptions

None

Example

```
DECQUALBASE requires the existance of prod_name, Version x.x or later.
Subset DECQUALBASE will not be installed at this time.
```

For more information

- *ULTRIX Software Development*, Volume 1, *Guide to Preparing Software for Distribution on ULTRIX systems*
- *Guide to Programming Support Tools*
- *DEC STD 041-0 Customer Installability: Product Requirements*
- Developers Guide to VMSINSTAL
- Or the operate Programing Guide/Manual

prev_v_compatib

The product has been tested for compatibility with previous product version.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

A product must be tested to verify that a user's upgrade to a new version will not adversely affect the system environment or user files associated with that product.

Rationale

To protect a customer's investment in past use of a product, it is in the product's best interest to do compatibility testing with previous versions.

Exceptions

First versions of products are not required to perform this testing. Nor are products for which a formal decision has been made by product management not to support upgrades from previous versions.

Example

Not applicable

For more information None

priv_req

The preinstallation section of the installation guide lists all privileges required for the product installation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS MS–DOS Windows NT Windows mac

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

The preinstallation section of the installation guide should list all privileges required for product installation.

OpenVMS

It does not suffice to say that the installation should be made from the SYSTEM account, since not all users will be installing from the SYSTEM account. Required privileges should be explicitly stated. A note should also be included that states VMSINSTAL turns off BYPASS at the start of the installation.

Rationale

To ensure a successful product installation.

Exceptions

None

Example

OpenVMS

To install DECforms, you must be logged into an account that has SETPRV or the following privileges:

- CMKRNL
- WORLD
- SYSPRV

Note that VMSINSTAL turns off BYPASS privileges at the start of the installation.

PC

Installing PC software to a network server will require network operating privileges and should be documented.

For more information

None

probl_report

The postinstallation section provides instructions for reporting problems to Digital.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

The postinstallation section of the installation guide should advise the installer to report any software problems to Digital. The user should be instructed to take one of the following actions depending on the nature of the problem and the type of support they have:

- Call Digital if your software contract or warranty agreement entitles you to telephone support.
- Submit a Software Performance Report (SPR).
- Fill out and submit a Reader's Comments form if the problem has to do with the layered product documentation.

  This form is located at the back of each manual. The installer should use the form from the documentation in which the error was found and include the page and section numbers.

Rationale

Provides the user with a method for reporting problems.

Exceptions

None

Example

Determining and Reporting Problems

If an error occurs while DECforms is being used, and you believe that a problem with DECforms is causing the error, do one of the following:

- If you have a BASIC or DECsupport Software Agreement, call your Customer Support Center.
- If you have a Self-Maintenance Software Agreement, submit a Software Performance Report (SPR).
- If you have purchased DECforms within the past 90 days, submit an SPR.

If you find an error in the DECforms documentation, fill out and submit the Reader's Comments form at the back of the document in which you found the error. Then indicate on the form the section and page number of the error.

For more information

Refer to Appendix C Reference Documents for pointers to Installation Guide templates.

proc_sup

The product has been tested on one of the processors from each supported processor class.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

OpenVMS

The following sections are specific to OpenVMS layered products.

Description

All OpenVMS layered products are required to support the full VAX processor range. You should test your layered product once on at least one processor for each class of processor. For example, testing on a VAX 8300 or VAX 8200 will qualify a product for both.

Following the initial release of a product, testing is recommended on different processors and configurations to verify operation over the range of processors. If your product has specific hardware or software constraints or is "hardware sensitive," more frequent evaluation of the product is recommended.

Software Quality Operations and Information Services (SQOIS) maintains a processor support table that lists the VAX processors that can be used to test a family of processors. Refer to the For More Information section at the end of this requirement description for information on the location of this Central Processing Unit (CPU) table.

Symmetric Multiprocessing

Your product should be tested on a processor that runs symmetric multiprocessing.

Some layered products work properly in a single-processor environment, yet fail in a multiprocessor environment.

VAXcluster Systems

Clustering is a key feature in marketing VAX processors. Clustered systems offer high processing capabilities in a distributed environment, which provides something that a single mainframe cannot offer—availability. For details on the services and configurations available in a VAXcluster environment, refer to the VAXcluster Software SPD, 29.78.xx.

Your layered products should adhere to the full support cluster model.

Failure to adhere to this model should be addressed early in the review cycle.

Rationale

Digital Equipment Corporation has a legal obligation to ensure that its software products sold for use on supported processors will function properly on those processors.

Exceptions

All OpenVMS layered software products are required to support the full VAX processor range unless a specific architectural or hardware restriction exists.

Example

Not applicable

UNIX

The following sections are specific to UNIX based layered products.

Description

A product must be tested once on at least one processor for each class of processor. For example, testing on a DECsystem 2100 or a DECsystem 3100 qualifies a product for both. Refer to the Help icon at the end of this requirement description for further information on classes of processors.

Layered product groups must also test their products on a processor that runs symmetric mutliprocessing. Some layered products work properly in a single-processor environment, yet they fail in a multiprocessor environment.

Rationale

Digital Equipment Corporation has a legal obligation to ensure that its software products sold for use on supported processors will function properly on those processors.

Exceptions

A processor support table does not exist within Digital for third-party processors. It is recommended that software be fully tested on all third-party processors supported by a product when access to these processors is available.

Example

Not applicable

For more information

If You Need	Do the Following
In VIRTUE::UNIXINFO note #16.1 has the info on processors supported by ULTRIX, 16.2 contains the Alpha processor support table. The alpha table contains processor info for all of the operating systems that alpha CPU's support.	
In VIRTUE::VMSINFO note 358.2 there is the VMS processor support table. Note 358.3 contains an alpha cpu table that is a duplicate of what is in UNIXINFO #16.2.	

prod_name_v

The installation outputs the correct layered product name and version number at the beginning and end of the installation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

The installation procedure should display the correct layered product name and version number at the beginning and end of the installation.

Rationale

This requirement promotes consistent, uniform, and user-friendly installations.

Exceptions

None

Example

```
* Beginning installation of DECforms V1.4 at 12:03
.
.
.
*Installation of DECforms V1.4 completed at 12:10
```

For more information None

prod_startup_com

A product startup command procedure is provided when users need to issue more than one command to start the product.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Products that require users to issue more than one command to start the product should provide a startup command procedure. The startup file should be easily identifiable by using a naming convention such as `product_identifier$startup.com` (for example, DECW$STARTUP.COM).

If the layered product supplies a startup command procedure, be sure to define all system-wide logical names in SYSTEM/EXECUTIVE mode. Also be sure to provide the startup file to the system common SYSMGR directory.

Rationale

A startup command procedure reduces the chances of user error, promotes consistency with other layered products, and results in an easier installation. Startup command procedures are also useful for restarting a product that has been stopped either deliberately or by system failure.

Exceptions

When one command procedure to start up the product is sufficient.

Example

Not applicable

For more information None

prod_variants

OpenVMS testing requirements for product variants were negotiated and documented in the "Product Variant Evaluation Strategy" document.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Early in the development stage, you should negotiate with ESQG and the Localization Center the OpenVMS testing requirements for your product variant. This does not apply to Asian-based product variants. The agreement should be documented in the “Product Variant Evaluation Strategy.” Refer to the For More Information section at the end of this requirement description for information on obtaining copies.

Product variants should also be tested on a processor that runs symmetric multiprocessing.

Rationale

Ensures that the product can be installed and used in all supported product language and market variants.

Exceptions

None

Example

Not applicable

For more information

To request a copy of the “Product Variant Evaluation Strategy” send mail to SQMUK::SQM.

prod_variants_instl

If the installation procedure cannot find language-specific savesets, it should exit gracefully and display an error message.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Each product variant of a Basic International Version (BIV) contains certain language-specific savesets, but uses the same installation procedure as all other product variants. Therefore, knowledge of the required language is not hardcoded into the installation procedure and hence the need to ask for the language. After inputting the language, the installation procedure will use this information to search for language-specific savesets that are named accordingly. If these savesets cannot be found, the installation procedure should fail gracefully and display a suitable error message.

Rationale

Displaying an error message lets the user know that the installation was not completely successful.

Exceptions

Layered products that do not have product variants.

Example

Not applicable

For more information None

prod_variants_instl_paths

If the product is a product variant, only valid installation paths are successful.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

If the product being installed is a product variant, you must check that only valid installation paths are successful. For example, if the product is a Norwegian-language product variant, it may not be installed as any other language.

Rationale

Prevents installers from accidentally selecting and installing invalid product variants, which could potentially break the existing parent product.

Exceptions

None

Example

Not applicable

For more information None

q_prompts

The Installation Verification Procedure (IVP) that prompts the user for input can recover from invalid input and reprompt the question.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX		

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

The Installation Verification Procedure (IVP) confirms that the user input is complete and correct. The IVP cannot check all user input but should provide error handling capabilities when appropriate. For example, when input such as a Yes/No choice is requested, the IVP should verify that either Yes or No was entered.

Rationale

Reprompting for input after invalid answers helps to ensure that the IVP is executed correctly and efficiently.

Exceptions

Products that do not prompt the user for input during IVP execution do not have to comply with this requirement.

Example

```
Is this the correct hardware address (Y/N): X
Is this the correct hardware address (Y/N):
```

In this example, the IVP prompts the user for Yes or No (Y/N) input, but the user responds with an invalid answer. The IVP determines that the input is incorrect and reprompts the user.

For more information None

reboot_after_instl

A reboot is not required before or after the installation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ_CHECK

Description

A reboot should not be required either before or after an installation.

Rationale

Reboots should be avoided whenever possible as this is noted as a cause for customer dissatisfaction.

Exceptions

A reboot may be required before or after the installation to reset system parameters for the product to run or to maintain system performance.

Additionally, any required changes to the operating system kernel will necessitate a system rebuild and reboot.

Example

Not applicable

For more information None

regis_prod_attrib

The product name has been defined in accordance with the recommended naming conventions and registered in the product registration data base.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): APRS

Description

Product Registration Defined

Product registration is the process whereby specific product and software attributes are assigned and reserved for the exclusive use of one product. In this document product attributes include the product name, product description, product type, and the Software Product Description (SPD) number. Software attributes are defined to include various registration classes and mnemonics.

Product Attributes to be Registered

The following table lists the product attributes you need to register for products layering on any of the operating systems listed in the "Operating Systems (All Architectures)" section of this requirement.

Product Attributes
Product name
Product description
Type of product
SPD number
Operating system supported
UPILV
Software Quality Operations and Information Services (SQOIS) (formerly SQM) site to which you will submit kits

When to Register Attributes

Register all relevant product attributes for your product as early as possible, preferably during the design and implementation phases of the development cycle. However, all attributes should be registered before the product is released to the Software Supply Business (SSB) group.

Also, you need not wait until you possess the "final" attributes data. For example, you can register a product under its development code name and later update the registration data base with the product's final name.

How to Register

To register or update product attributes, follow these steps:

1. Use the APRS tool described in Appendix A.
2. Register new attributes or update existing ones by using the APRS tool.

Refer to the For More Information section at the end of the requirement description for a pointer to the software and documentation location and how to obtain assistance from the Product Registrar.

The following section provides you with directions on defining a product's name.

Registering Product Name

The product name is the legal name of the layered product. If the product has an SPD, then the product name as shown on the SPD should be registered. If the product does not have an SPD, then the most formal name of the product should be registered.

Follow this format when defining a product name:

[DEC / DECwindows] Product_name[/Lang_var1] for [Operating System] [/Lang_var2]

where:

- The terms “DEC” or “DECwindows” should precede the product name. For example:

  ```
  DEC GKS
  DECwindows DECnet/SNA 3270 Terminal Emulator
  DECwrite for ULTRIX
  ```

- “Product_name” is spelled out.

  Do not use acronyms unless the acronym is well known in the industry. If you do use acronyms, first write the name in full and then follow with the acronym in parentheses.

  If the acronym is used within the documentation, call out the full name in the preface of the document with a note indicating that future references to the product name will use the acronym.

  Try to choose a product name that will not become too cryptic if it has to be shortened to approximately 30 characters for media labels.

- "/Lang_var1" is the qualifier used for an international product to represent the language in which the product is available. For example, the word "Japanese" in the following product name.

  DEC Quality**/Japanese for ULTRIX**

  The following list shows some of the available language variants:

```
/American          /Hanyu
/Arabic            /Hanzi
/British           /Hebrew
/Canadian          /Italiano
/Can-Français      /Japanese
/Dansk             /Nederlands
/Deutsch           /Norsk
/Español           /Portugues
/Français          /Suomi
/Hangul            /Svenska
```

  Use the country spelling of the language in the Roman character set. Do not use all capital letters when spelling out the language variant.

- "Oper_system" is the operating system on which the product runs. For example:

  DEC Quality/Japanese for **ULTRIX**

  The operating system is not platform-specific. For example, the word "ULTRIX" describes a product that executes on either the VAX or RISC version of the ULTRIX operating system.

  If it is necessary to differentiate between the VAX and RISC platforms, do this in:

  - The SPD when describing order numbers for the kits
  - The system manager and user documentation

- "/Lang_var2" is the qualifier used for products that run only on a local language variant of the operating system. For example:

  DEC Quality/Japanese for ULTRIX**/Japanese**

  If the product variant runs on both the original operating system and the product variant of the operating system, it is not necessary to add the second language qualifier to the name. For example:

  DEC Quality/Japanese for ULTRIX

  The available language variants are the same as those listed for "Lang_var1."

Rationale

Once registered, an attribute reserved by one product group cannot be used by another product group. The registration of product and software attributes allows Digital software products to co-exist in the same environment by ensuring that all product-specific programs, data, and devices are identified in a unique manner.

Exceptions

If your product is used only at Digital or layers onto the Sun platform, you are not required to register the product. However, the reasons explained in the Rationale section, you are strongly encouraged to do so.

Example

Not applicable

For more information

If you need	Do the following
Assistance on product registration issues	Contact the Product Registrar at VIRTUE::PRODUCT_REGISTRAR, or use the Feedback function of the APRS software.
The APRS software and user guide	Copy the files located in this directory: AURORA::WORK1$:[CUI_TOOLS.PUBLIC]
Information on the process of product naming	Refer to the *Corporate Product Introduction Guide*.
A copy of *DEC STD 095-0*	Refer to the Corporate VTX data base. At the system prompt, enter: VTX SMC [Return]. Follow the instructions to obtain a copy by mail.
Information on ISO guidelines for international products	Refer to *ISO 639, Codes for the Representation of Names and Languages* and *ISO 3166, Codes for the Representation of Names of Countries*.
OpenVMS	
Information on using the dollar sign	Refer to the *NAS Guide to Developing Portable Software* available on OpenVMS OLD CD-ROM, order # AA_PE81A-TK.
Information on naming conventions	Refer to the *Guide to Creating OpenVMS Modular Procedures*.
Information on error message status codes	Refer to the *OpenVMS Message Utility Manual* of the OpenVMS documentation set.
ULTRIX, OSF	
Further information on environment variables	Issue the command, `man 7 environ`

regis_software_attrib_ux

All registration classes and mnemonics have been defined and registered in the product registration data base.

OPERATING SYSTEMS (ALL ARCHITECTURES):

ULTRIX	OSF/1	Sun
SCO UNIX	AIX	HP-UX

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): APRS

Description

Product Registration Defined

Product registration is the process whereby specific software attributes are assigned and reserved for the exclusive use of one product. In this guide, software attributes are defined to include various registration classes and mnemonics.

Attributes to be Registered

The following table identifies the software attributes that you need to register for layered products operating in a UNIX-based environment.

Software Attributes
Product Code
Subset name
Man pages
Man page subsection
Environment variable
Daemon name
Device driver Mnemonic
Major number
Kernel option
Kernel pseudo-Device
Facility Prefix

Software Attributes

Term Cap Entries

The registration of software attributes applies to new products, or whenever changes occur to any of these attributes.

When to Register Attributes

Register all relevant software attributes for your product as early as possible, preferably during the design and implementation phases of the development cycle. However, all attributes should be registered before the product is released to the Software Supply Business (SSB) group.

How to Register

To register or update the software attributes, follow these steps:

1. Use the APRS tool described in Appendix A.
2. Register new attributes or update existing ones by using the APRS tool.

Refer to the For More Information section at the end of this requirement description for a pointer to the software and documentation location and information how to obtain assistance from the Product Registrar.

The following sections provide guidelines on the various software attributes that need registration.

Facility Prefix Names

Facility prefix names must be unique and should be registered with the Product Registrar.

The example shows the facility prefix name that is registered for a fictional product DEC Quality for ULTRIX:

```
qual
```

A facility prefix name is an alphanumeric string beginning with an alphabetic character (A-Z) which will prefix all of the product's APA-compliant programming interfaces. This string is used as a prefix to identify the product and its components in a unique manner. A facility name is required to comply with the Application Integration Architecture and Application Portability Architecture (AIA/APA). Facility names and facility numbers provide a critical link for layered product developers to synchronize event data across distributed applications and data bases running simultaneously on multiple platforms.

The product's facility name should be used to prefix the following components:

- Public procedures
- Private procedures
- Program modules
- Global variables
- Status codes and condition values
- Constants
- Macros

- Platform-specific conventions that require publicly visible names, such as ULTRIX environment variables or daemon names.

If your product operates on both the OpenVMS and UNIX platforms, you should register one facility prefix name only. If a facility prefix or facility number has already been registered for one platform, it will be cross-registered for the other platforms. This means that most UNIX layered products will be registered with their companion OpenVMS product's facility prefix and facility number. If your UNIX product does not have an equivalent OpenVMS product, you can choose a unique facility prefix name.

For example, if your UNIX product has a companion product called DEC Quality for OpenVMS, which has "qual" as its registered facility prefix name, you should register "qual" as the facility prefix name for your UNIX product.

man pages

ULTRIX subsections can only be registered after it has been requested. Mail must be sent to GORGE::QAR_INTERNAL, PW-QAR.

All man pages added by a layered product must be unique and should be registered. This means that a layered product should register all files it adds to the man directory structure. Any file that modifies an existing man page should also be registered.

Note

If a layered product needs to add a man page with the same name as a man page already registered, a man page subsection should be used.

A man page is a page from an ULTRIX reference manual that a user can display online by using the man command.

The expected format of a man page name is *name.d* where *name* is the man page name (not including the path) and *d* is a digit between 1 and 8 that indicates the section to which the man page belongs. Man pages are stored in eight sections according to usage:

1 - contains descriptions of commands that are available to all users.
2 - contains descriptions of system calls (entries into the kernel) that are used by all programmers.
3 - contains descriptions of the functions available in the system libraries.
4 - contains descriptions of special files, device driver functions, and network support.
5 - describes system file formats and their use.
6 - not used.
7 - contains descriptions of miscellaneous information that does not belong to any other section.
8 - contains descriptions of system operations and maintenance commands.

Man pages are placed in the following directory structures (where ? stands for one of the eight man page sections):

- ULTRIX man pages—/usr/man/man?
- OSF man pages—/usr/share/man/man?

- Japanese ULTRIX man pages—/usr/jsy/man

The following guidelines should be used when registering man pages:

- Man page names should be the same as the names of the commands or files being described. They should contain 36 or fewer characters. If a man page is associated with a subsection, the combined length of the man page name and subsection name must not exceed 36 characters.
- Man page names should use lowercase characters except when uppercase is required for the command or routine being described.
- Characters in man page names should not be significant to the shell. For example, $, #, and ? characters should not be used.

The following are examples of man pages that might be registered for the fictional product DEC Quality for ULTRIX:

/man1/qualchk.1
/man1/qualtest.1
/man8/qualsetup.8

man page Subsections

If a product uses a man page subsection, the name of the subsection should be registered.

Man page subsections are used to add an extra level of hierarchy to the man subsystem. Within each man? directory (for example, /usr/man/man1) man page subsections can group man pages that belong to a particular product or service.

Man page subsections are necessary only when more than one product intends to supply a man page with the same name and in the same section directory.

The following guidelines should be used when naming a man page subsection:

- The combined length of a man page and its associated subsection must not exceed 36 characters.
- The first character in a subsection must be a character (not a digit).
- A subsection must not end in ".digit".
- Subsection names must not contain shell characters.

The following are examples of subsections currently being used:

```
yp
nfs
X11
```

Note

One of the differences between the OSF/1 and the ULTRIX man macro is that in OSF/1, subsections are no longer hardcoded. Therefore, on OSF/1, use of a man subsection by a layered product does not require that the man macro be changed. You can register OSF/1 man page subsections directly into the registration data base with the APRS software. Send mail to the Product Registrar to register ULTRIX man page subsections.

Product Code

Product codes for ULTRIX can NOT begin with the letters U or O, regardless of whether or not the product is supporting the RISC or VAX platform. These letters are reserved for the ULTRIX operating system.

Register the product code used by your product. If it uses more than one product code, register all of them.

A product code is a 3-letter mnemonic used to prefix the subsets of a product.

The following guidelines should be used when registering product codes:

- The letters must be uppercase.
- If your product runs on multiple UNIX-based platforms, you can use one character of the product code to identify the platform associated with each product code. By convention, the character "V" represents VAX, "R" represents RISC, and "O" represents OSF/RISC.

For example, if the fictional product DEC Quality for ULTRIX includes the subsets `QTYDECTEST` and `QTYDECQUIZ`, the product code registered is `QTY`.

Subset Names

The product code and mnemonic portion of any subset names installed with the setld utility should be registered.

A subset name is a character string composed of the following parts, in the order listed:

- The three character product code (should be in uppercase)
- A mnemonic identifying the subset (should be in uppercase)
- The three digit version code

The entire subset name (product code, mnemonic, and version code) must not exceed 15 characters.

The following is an example of a subset name:

```
QTYTEST100
```

where:

→ “QTY” is the product code

→ “TEST” is the identifying mnemonic

→ “100” is the version code

The product code and the subset mnemonic are what you register with the Product Registrar. The version code does not get registered since it will differ for each release of the product.

For example, the fictional product DEC Quality for ULTRIX would register the subset name `QTYTEST` without a version code.

The setld utility requires subset names for installation. To avoid conflict during an installation, no two products should have the same subset name.

Layered products can have more than one subset. This may be necessary for larger products that have language variants or that are modularized into optional subsets that can be tailored by the user.

Environment Variables

For each environment variable added by the product, the name, planned use, and a pointer to more information should be registered.

Environment variables are used to associate a value with a name in a user's home shell. They can be passed to subshells.

Environment variables can be set up for a product by the export command and "name=value" arguments in /bin/sh or by the setenv command if /bin/csh is to be used.

The following guidelines should be used when registering environment variables:

- Environment variable names should be in uppercase.
- Planned use of the variable should be described in 80 or fewer characters.
- Pointers to more information (for example, more extensive documentation on the use of the variable) should be provided in a format of 80 or fewer characters.
- Since there is limited memory reserved for environment variables, the following recommendations are made to layered products:
  - Use configuration files rather than environment variables when appropriate. For example, use a configuration file when a sizable amount of customization is required and when the configuration information does not need to change regularly.
  - Reuse existing environment variables that define the entity to be used. To facilitate this, each registered environment variable is associated with a brief description of the use of the variable and a pointer to further information.
  - New environment variables should be created only when the information being conveyed differs on a timescale similar to the duration of a users login session or shorter.

For example, the fictional product DEC Quality for ULTRIX would register the following environment variable:

Name:	`QUAL_QUALITY`
Description:	`Identifies the location of quality information for the system. Refer to documenation or consult the project leader.`

Daemon Names

Register the full path name of any daemons started by your product.

Daemons run as background processes on behalf of a product.

The following guidelines should be used when registering daemon names:

- Daemon names should end with a "d" when there is a possibility of conflict with a command.
- Daemon names usually consist of lowercase characters.

The following are example daemon names for the fictional product DEC Quality for ULTRIX:

```
/usr/lib/qty/qualchkd
/usr/lib/qty/qualtstd
```

Device Driver Mnemonics

Any device driver mnemonic that is referenced by the MAKEDEV command, and is added by the layered product, must be registered with the Product Registrar. This does not apply to OSF layered products.

Device drivers are programs that provide communication between a device and the kernel. The kernel is the central program that runs the computer. Mnemonics are assigned to these device drivers, and they are used to create the special files usually located in the /dev directory. They can exist in either the /dev/MAKEDEV or /dev/MAKEDEV.local file. Examples of types of devices that use a mnemonic include terminals, pseudo-terminals, disk drives, and tape drives.

Major Numbers

Major numbers and device classes must be unique and should be registered with the Product Registrar. This does not apply to OSF layered products.

A major number is a numeric identifier that links a device to its appropriate device driver. The major number determines which device driver will interface with a device.

Devices are broken down into two classes:

1. Block (also known as structured). For example, disks and tapes.
2. Character (also known as unstructured). For example, terminals, communications lines, and line printers.

Block devices work with addressable 512-byte blocks. Character devices handle data in all formats other than 512-byte blocks.

For each of the two device classes there is an array of device configuration tables. The major number reflects the position of the device driver routines within the proper table.

The following example shows the registered major number and device class for a fictional product DEC Quality for ULTRIX:

```
23 (block)
```

where:

→ "23" is the major number

→ "(block)" is the device class

Kernel Options

Kernel options must be unique and should be registered with the Product Registrar. This does not apply to OSF layered products.

Kernel options are any system facilities that are compiled into the system configuration and are not considered mandatory. Kernel options would be included within a system's configuration file under the options description.

Examples of some system-level kernel options include local area transport connections (LAT), internet communication protocols (inet), and support for downline loading (DLI).

Kernel Pseudo-Devices

Kernel pseudo-devices must be unique and should be registered with the Product Registrar. This does not apply to OSF layered products.

A kernel pseudo-device is an operating system component for which there is no associated hardware. To enable the operating system to recognize these components, a pseudo-device definition is required within the system's configuration file.

Examples of pseudo-devices are the pseudo-terminal driver (pty) and the network file system protocol support.

Refer to the For More Information section at the end of this requirement description for more information pointers.

Rationale

Once registered, an attribute reserved by one product group cannot be used by another product group. The registration of attributes allows Digital software products to co-exist in the same environment by ensuring that all product-specific programs, data, and devices are identified in a unique manner.

Exceptions

If your product is used only at Digital, or layers onto a Sun platform, you are not required to register attributes although it is strongly recommended that you do so.

Example

Not applicable

For more information

If you need	Do the following
Assistance on product registration issues	Contact the Product Registrar at VIRTUE::PRODUCT_REGISTRAR, or use the Feedback function of the APRS software.

If you need	Do the following
The APRS software and User's Guide	Copy the files located in this directory: VIRTUE::WORK1$:[CUI_TOOLS.PUBLIC]

The following identifies the APRS software files you need for your working environment:

aprs.com
aprs.exe
aprs_users_guide.ps

Refer to the APRS User's Guide for details on installing and running the APRS software to register product and software attributes.

Information on the process of product naming	Refer to the *Corporate Product Introduction Guide*.
A copy of DEC STD 095-0	Refer to the Corporate VTX data base. At the system prompt, enter: VTX SMC [Return]. Follow the instructions to obtain a copy by mail.
Information on ISO guidelines for international products	Refer to *ISO 639, Codes for the Representation of Names and Languages* and *ISO 3166, Codes for the Representation of Names of Countries*.
A complete list of file types	Refer to {VMS,ULTRIX}_GENERIC_FILE_TYPES.TXT available from {VIRTUE, SQMUK, SQMJPN}::SYS$INFO.
Further information on environment variables	Issue the command, `man 7 environ`

regis_software_attrib_vms

All registration classes and mnemonics have been defined and registered in the product registration data base.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OpenVMS	MS–DOS
PERFORMED BY:	Software Engineer	
REQUIREMENTS AID(S):	APRS	

Description

Product Registration Defined

Product registration is the process whereby specific product and software attributes are assigned and reserved for the exclusive use of one product. In this guide software attributes are defined to include various registration classes and mnemonics.

Attributes to be Registered

The following table identifies the software attributes that you need to register for layered products operating in the OpenVMS, MS–DOS environments.

Software Attributes
Facility name
Saveset name
Logical name
File type extension
Shareable images
Print form definition number
Library format number
Executive system service number
Kernel system service number
Device driver name
Command verb

The registration of software attributes applies to new products or whenever changes occur to any of these attributes.

When to Register Attributes

Register all relevant software attributes for your product as early as possible, preferably during the design and implementation phases of the development cycle. However, all attributes should be registered before the product is released to the Software Supply Business (SSB) group.

The following sections provide guidelines on the various software attributes that need registration.

How to Register

To register or update product attributes, follow these steps:

1. Use the APRS tool described in Appendix A.
2. Register new attributes or update existing ones by using the APRS tool.

Refer to the For More Information section at the end of the requirement description for a pointer to the software and documentation location and how to obtain assistance from the Product Registrar.

Facility Name

All Digital software products should use their facility name followed by a dollar sign ($) to identify the software as supplied by Digital.

A facility name is an alphanumeric string used as a prefix to identify a product and its components in a unique manner. The alphanumeric string includes:

The character set A to Z
The numbers 0 to 9

Use the facility name as a prefix for any software entity that must be unique. For example: QUAL$ represents the facility name for a hypothetical product called VAX Quality.

When registering the facility name, indicate if an OpenVMS error message status code number is to be assigned. These are used to signal errors with the OpenVMS Message Utility.

The following table shows the product components for which you should use the facility name as a prefix, the naming format, and an example.

Component	Format	Example
File names	facility_name$string	QUAL$CLIENT_ MAIN.EXE
Rights Data base identifiers	facility_name$rights_identifier	QUAL$IDENT
Data structure definitions	facility_name$K_CLASS_S	QUAL$K_CLASS_S
	facility_name$K_DTYPE_T	QUAL$DTYPE_T
Global symbols	facility_name$symbol_description	QUAL$EDIT
Entry points	facility_name$procedure_name	QUAL$TEST
Status codes and condition values	facility_name$status_code	QUAL$STATUS_1

Component	Format	Example
Program section (PSECT) names	facility_name$psect_name	QUAL$TEST_ DISPLAY
Queue names	facility_name$queue_name	QUAL$QUEUE_1
System lock identifiers	facility_name$systemlock	QUAL$LOCK_1

Note

On using the dollar sign ($):

Some controversy exists regarding use of the dollar sign ($) in file names, entry points, and identifiers. Applications written to be portable to other platforms (for example, portable from VAX/OpenVMS to VAX/ULTRIX, or OSF/1, or SUN) cannot include the $ because this character has restrictions in platforms other than VAX/OpenVMS. If your product will be ported to other platforms, substitute the underscore character (_) for the $ in cases where the $ is not appropriate, and continue to use your registered facility name. For more information, refer to *NAS Guide to Developing Portable Software*, which is available on OpenVMS OLD CD-ROM, order number AA-PE81A-TK.

Send mail to VIRTUE::PRODUCT_REGISTRAR if you have questions about the current guidelines for using the dollar sign ($).

Refer to the For More Information section at the end of this requirement description for more pointers to information on file naming conventions.

Logical Name Prefixes

- Register any logical name prefixes that will be used. It is recommended to use the product's facility code as an identifying prefix in the following format: `facility_name$string`

  For example: `QUAL$MAIN`.

- For system-resident processes in a layered product, use unique registered names. Use the facility code as the prefix to a system-resident process name. This will ensure that another Digital product does not run a process with the same name.

  A system-resident process is one that is always in the system. It is usually in hibernation or in a wait state.

- Use the prefix in assigning logical names to permanent mailboxes.

  Permanent mailboxes are entered into LNM$PERMANENT_MAILBOX, which is usually set to the system logical name table LNM$SYSTEM. Other processes can then access the mailbox simply by knowing its name and using the $ASSIGN directive to assign their channel to it. If a process were to create a mailbox ($CREMBX) by using a name that already existed, it would get a channel to the existing mailbox and no error indication. This can be prevented by adopting the prefix.

File Type Extensions

New file types should be registered in the registration data base.

A file type extension is a character string from 1 to 39 characters that usually identifies the file in terms of contents.

Several file type extensions are already registered as generic (for example, .EXE and .COM) and need not be registered. Use generic file type extensions whenever possible.

When determining a file type, evaluate the use of a default file type before creating your own. Refer to the For More Information section at the end of this description for a pointer to default filetypes.

If your product uses a file type that has been registered by another product group, you should coordinate the use of the extension with the other product group.

Shareable Images

If a product ships a shareable image, you should register the image name. The recommended naming format is:

```
facility_name$stringSHR.EXE
```

Print Form Definition Numbers

When registering print form definition numbers with the APRS software for products shipping a predefined form definition, the APRS software will generate the number when you press the appropriate key sequence.

Print form definition numbers permit unique form definitions and prevent conflicts among the family of OpenVMS layered products.

Library Format Numbers

If your product uses the OpenVMS Library Utility, and it has a unique library format, the APRS software will generate a library format number when you press the appropriate key sequence. The library format number is needed only if you are creating a new library format. Library format numbers are internal codes used by the OpenVMS Library Utility.

The OpenVMS Library Utility currently supports the following library formats:

object
text
help
image

Supporting a new library format requires a new release of the OpenVMS operating system.

Executive and Kernel System Service Numbers

If your product requires executive or kernel mode system service numbers, you will need to register each executive and kernel system service number. When registering with the APRS software, you will register each service number by pressing the appropriate key sequence.

System service numbers are vectors on the executive used by products that add system services to OpenVMS. Products that create a new system service require a system service number.

Saveset Names

Register the names of the savesets that package the product and from which VMSINSTAL installs the product.

Saveset names can have up to 39 characters—3 to 27 of these characters are allotted to the facility code.

Saveset names for magnetic-tape distribution are limited to 17 characters, including the period and the saveset letter.

Follow this format for naming savesets:

`facvvu.s`

where:

- "`fac`" is the facility code
- "`vv`" is the current version number
- "`u`" is the current update number
- "`s`" is the sequence identifier

Each saveset must have a unique sequence identifier.
Savesets are assigned an alphanumeric sequence identifier based on the order in which they should be installed: ".A" for the first saveset to be installed, ".B" for the second saveset to be installed, and so on.

International products should have:

- A common international base (A) component
- A variety of language-specific (B) components
- A variety of market-specific (C) components

For example, the primary saveset for the hypothetical product VAX Quality has the following format and name for each component of the international product model:

Component	Format	Example
Base (A)	[fac][vvu].s	QUAL040.A
Language (B)	[fac]Lxx[vvu].s	QUALLDA040.A
Market (C)	[fac]Cxx[vvu].s	QUALCDK040.A

The examples illustrate the following points:

- "`fac`" is represented by the facility code "QUAL" which is used as the saveset mnemonic
- "`xx`" in the language and market components is represented by ISO codes. "DA" is the ISO language component code for Danish/Dansk, and "DK" is the ISO market component code for Denmark
- "`L`" in the language component signifies a language variant
- "`C`" in the market component signifies a country-specific product variant.

Device Driver

The registration of devices, drivers, and pseudo devices is now a joint effort between the OpenVMS development group and Software Quality Operations and Information Services (SQOIS). For developers, registration consists in selecting a two-character mnemonic and contacting the appropriate registry. Two-character device mnemonics can be a device specification prefix, a driver name prefix, or a pseudo device. Device names and short descriptions are recorded as are the buses supported.

To get help in selecting a two-character mnemonic, refer to the following file:

{VIRTUE, SQMUK, SQMJPN}::SYS$INFO:DEVICE_DRIVER.MASTER

The file contains a list of registered mnemonics and some suggestions for selecting characters for the mnemonic. Please abide by the suggestions and choose something that is not used by another product.

The term "device" is used in two situations. Device name refers to the official registered name for a piece of hardware (for example, DSB32). A device specification, which is used by the OpenVMS operating system as the device address, is made up of the registered two-character mnemonic followed by the controller letter and unit number (for example, DBA0:).

The term "driver" is used to identify the "Driver name" which is made up of the registered two-character mnemonic followed by the word "DRIVER", for example, TTDRIVER.

If a layered-product developer wants a driver or pseudo device to be bundled with the OpenVMS operating system, support must be negotiated with the OpenVMS development group. The terms used to represent a driver or pseudo device that is shipped as part of the OpenVMS product are "OpenVMS Supported," "Bundled with OpenVMS," and "Shipped on the Master Pack."

The Registries

The following groups record the mnemonics and make assignments:

- The Hardware Registry is responsible for the Q–bus, UNIBUS, VAXstation 2000, MicroVAX 2000, VAXBI, and SCSI lists.
- The OpenVMS Product Registry, administered by SQOIS, is responsible for the two-character mnemonics.

The registration procedures vary depending on whether you are registering a device, a device with a driver, a driver only, or a pseudo device. The following table shows the appropriate registry contacts for each situation:

To register:	Contact:
A device without a driver	The Hardware Registry for a device specification, The OpenVMS Product Registry for a device mnemonic.
A device with a driver	The Hardware Registry for a device specification, the OpenVMS Product Registry for a device mnemonic and a driver mnemonic.

To register:	Contact:
A driver only	The OpenVMS Product Registry for a driver mnemonic.
A pseudo device	The OpenVMS Product Registry for a pseudo device mnemonic.

Refer to {VIRTUE, SQMUK, SQMJPN}::SYS$INFO:DEVICE_DRIVER_REGISTRATION.TXT for details and contact names.

Rationale

Once registered, any attributes reserved for your product cannot be used for another product, unless negotiated with you.

The registration of software attributes allows Digital software products to co-exist in the same environment by ensuring that all product-specific programs, data, and devices are identified in a unique manner.

Exceptions

If your product is used only at Digital, you are not required to register the product information. For the reasons exlained in the Reationale section, you are strongly encouraged to do so.

Example

Not applicable

For more information

If you need	Do the following
Assistance on product registration issues	Contact the Product Registrar at VIRTUE::PRODUCT_REGISTRAR, or use the Feedback function of the APRS software.
The APRS software and User's Guide	Copy the files located in this directory: VIRTUE::WORK1$:[CUI_TOOLS.PUBLIC]

The following identifies the APRS software files you need for your working environment:

aprs.com
aprs.exe
aprs_users_guide.ps

Refer to the APRS User's Guide for details on installing and running the APRS software to register product and software attributes.

If you need	Do the following
Information on the process of product naming	Refer to the *Corporate Product Introduction Guide*.
A copy of DEC STD 095-0	Refer to the Corporate VTX data base. At the system prompt, enter: VTX SMC [Return]. Follow the instructions to obtain a copy by mail.
Information on ISO guidelines for international products	Refer to *ISO 639, Codes for the Representation of Names and Languages* and *ISO 3166, Codes for the Representation of Names of Countries*.
A complete list of file types	Refer to {VMS,ULTRIX}_GENERIC_FILE_ TYPES.TXT available from {VIRTUE, SQMUK, SQMJPN}::SYS$INFO.
Information on using the dollar sign ($)	Refer to the *NAS Guide to Developing Portable Software* available on OpenVMS OLD CD-ROM, order number AA_PE81A-TK.
Information on naming conventions	Refer to the *Guide to Creating OpenVMS Modular Procedures*.
Information on error message status codes	Refer to the *OpenVMS Message Utility Manual* of the OpenVMS documentation set.

regress

Regression tests have been performed.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Regression testing is one of the most common techniques for testing software. It involves running established software tests for which the results are known, and comparing the new results with the known results.

If the new results do not conform to the previously verified results, the software being tested can contain errors. If errors do exist, the software is said to have regressed. Thus, regression testing is a method of ensuring that a program being developed runs consistently and that new features added to a program do not affect the correct execution of previously tested features.

Rationale

Regression tests help to ensure that new features added to a program do not affect the correct execution of previously tested features.

Exceptions

None

Example

The following is a typical sequence of steps for regression testing:

1. Write test scripts to test the software.
2. Organize the tests and create a mechanism for the team to readily access the tests as needed.
3. Run the tests.
4. Examine the test results:
   a. Compare the results of each test to the expected results. Note any differences.

b. For incorrect test output, revise the program code to correct the problem. Repeat steps 3 and 4 until the test output is correct.

c. Save the correct output as the validated test results.

5. Repeat steps 3 and 4 whenever the program is modified. If the current and validated test results match, the program being tested is working as expected.

   If unexpected changes are found in the test results, the program being tested can contain errors. Correct the program and rerun the tests whose results did not match. Repeat this cycle until all results are valid.

   For future test runs, use these validated test results for comparison.

For more information

For hints on regression testing techniques, refer to the VIRTUE::TESTING VAX Notes conference or the *Digital Guide to Software Development*.

released_ft_os

The product has been tested on the latest release and field test versions of the operating system on which it is layered.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer/Product Manager

REQUIREMENTS AID(S): None

Description

The layered product should be installed and tested on each supported version of the operating system on which it is layered, including the current field test version, if available. "Supported versions" means the current shipping version and the previous release. System Support Addendum (SSA) or Software Product Description (SPD) should not mention field test versions of operating systems.

Current ULTRIX OS Information

To obtain the current field test version of the ULTRIX or OSF/1 operating system, use the Online Registration and Distribution System (ORDS). Refer to the Help icon at the end of this requirements description for a pointer to further information on ORDS.

Current OpenVMS OS Information

To obtain the current field test version of the OpenVMS operating system, send mail to VIRTUE::INFORMATION.

Rationale

Compliance with this requirement verifies the layered product's legal support of the operating system.

Exceptions

The Sun operating system field test versions are not likely to be available.

Example

Not applicable

For more information

None

robust_ivp

The Installation Verification Procedure (IVP) is robust—it compiles, links against libraries, adds or deletes records, performs conversions, and so on—and it demonstrates adequately that the product operates successfully.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX		

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

OpenVMS The following sections are specific to OpenVMS layered products.

Description

When designing Installation Verification Procedure (IVP)s, layered products are divided into basic classes geared toward its particular class. These classes and recommendations are listed below.

Interactive IVPs

IVPs that ask questions should not be executed as part of the installation procedure. The ASK callback cannot be used within the IVP since IVPs should be executable outside of the VMSINSTAL command procedure.

The auto-answer file when used during an installation would not record any answers and subsequent installations using an auto-answer file would fail.

The installation should not prompt the user as to whether or not the IVP should be run. Instead, the installation should output a message stating that an IVP is available and explain why it cannot be executed as part of the installation. A description should be provided of where the residual IVP will be located and how to invoke the IVP separate from the installation.

The IVP should provide the user with a Help option for each question, along with default answers when possible. The user should be informed before running the IVP about any required pre-configuration, which should also be documented in the installation guide.

Non-Interactive IVPs

If the IVP can be run as part of the installation (non-interactive), you should use the SET IVP callback with the product's KITINSTAL.COM to ask the users if they want to execute the IVP immediately after the installation. For example:

```
$ VMI$CALLBACK SET IVP ASK
```

The automatic execution of the IVP upon completion of the installation is strongly discouraged.

Set Verify

Layered products should not automatically turn off the ability to run the installation and the IVP with SET VERIFY turned on. Without the ability to run the IVP with VERIFY turned on, it is much more difficult to determine the cause of any installation failure.

Compiler Languages

Compiler products should compile a robust program, link against Run Time Library (RTL), and confirm that it works.

Serious problems caused by RTL changes in new OpenVMS releases affecting layered products can be uncovered by this exercise. Compiling, linking, and running a small image that uses some features of the libraries or interacts with prerequisite products can easily demonstrate that the product is installed and functions properly.

Data bases

The IVP for data base products should be more complex and complete than the average layered product to establish product integrity. Data base products should supply an example data base with data in it. The layered product should touch the data base, enter a new record, delete an old record, and so on, to confirm. If data base conversions are required, these should also be performed on the sample data base.

Data base and data query products are designed for controlling, maintaining, and manipulating information applications. Due to their complex nature, these products tend to exercise OpenVMS more than the average product, and therefore should be more robust.

Forms and Graphics

The IVP should exercise the components of the operating system with which the product interfaces, such as RMS, Digital Command Language (DCL), and DECwindows. Additionally, since these products are often directed toward the general user, and since their main feature is visual, the IVP must offer the ability to display the usability and practical applications. Products supporting the DECwindows user interface should include an IVP that exercises that interface.

Utilities and Applications

The IVP should check the main attributes of the product to ensure the soundness of the operating system interface.

Device Drivers

Because of the unique individual nature of driver products and their reliance on special hardware, the IVP should verify correct installation, even when special hardware is not available.

In other words, some action can be performed on the drive, even if it is as basic as the manipulation of register values. The test should consider the ability to perform more complex driver operations.

Communications Software

The user should be able to perform simple look-up testing and should have the ability to invoke, upon request, more complex testing procedures that show common product operations.

Product Variant Considerations

The IVP of product variants should check that local language screens are displayed correctly and local language responses are accepted.

IVP Execution

IVPs should be executed only after files have been moved to their target directories to avoid IVP failure. The IVP cannot properly confirm the integrity of the installation if the files have not been moved to their target directories.

Rationale

IVPs are not meant to test every feature, enhancement, or problem fix included in the software package, but since the installation is the customer's first look at the product, the IVP must demonstrate that the software performs properly.

Exceptions

None

Example

Not applicable

ULTRIX, OSF, Sun

The following sections are specific to ULTRIX, OSF, and Sun layered products.

Description

The Installation Verification Procedure (IVP) confirms that the software installation was complete and correct.

The IVP should robustly confirm the product's functional integration with the system and demonstrate functionality. There should be one comprehensive IVP for the product, not one IVP for each subset. Developers should write the IVP to be as complete a test as possible for the product.

Rationale

IVPs are not meant to test every feature, enhancement, or problem fix included in the software package, but since the installation is the customer's first look at the product, the IVP must demonstrate that the software performs properly.

Exceptions

None

Example

```
# setld -v FEBMC300

Beginning DECbridge 500 V3.0 IVP

Successfully found ./usr/kits/FEB300/defeb300.release_notes
Successfully found ./usr/kits/FEB300/defeb300.sys
Successfully found ./usr/kits/FEB300/mib_ii_decem_1990.txt
Successfully found ./usr/kits/FEB300/fddi_mib_nov_1990.txt
Successfully found ./usr/kits/FEB300/dec_elanxt_ven_mib_jan91.txt

If you have a DECbridge 500 presently running on your network,
you may at this time get the version information from the device.
This will occur via a ndu -s command.
You will need to know the Mac address of of the DECbridge 500
to do this.

Do you want to get the version information from your DECbridge? (Y/N) [N] Y

Enter the Mac address: (08-00-2b-14-13-10) 08-00-2b-13-db-68
(c) Digital Equipment Corporation. 1991. All Rights Reserved.
Name          DEFEB
Firmware Rev  V3.0
Successful completion of DECbridge 500 V3.0 IVP
```

This is an example of an IVP display for the DECbridge 500 product.

For more information

Refer to the installation verification requirements in this guide for more information on designing and implementing an IVP.

rtl

If product shipped an Run Time Library (RTL), then close contact is kept with OpenVMS Engineering.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OpenVMS
PERFORMED BY:	Software Engineer
REQUIREMENTS AID(S):	None

Description

OpenVMS Run Time Library (RTL) is a library of prewritten, commonly used routines that perform a variety of functions. RTL routines can only be used in programs written in languages that produce native code for the VAX hardware.

As each version of OpenVMS may include a variation of a shipped RTL, it is advised that any layered product shipping, and subsequently replacing an RTL, perform a OpenVMS version check to replace or not replace the RTL accordingly. Specific dependencies may exist with certain versions; it is therefore important to understand any changes made to the RTL in each OpenVMS version.

Products shipping an RTL must remain in close contact with OpenVMS Engineering. Prior to making any changes to an existing RTL, OpenVMS Engineering should contact the parent development group. The development group must be aware of any and all changes during the release cycle of a particular OpenVMS version.

RTLs for Product Variants

In cases where the products ship an RTL on a language variant operating system, the development group of the language variant should also stay in close contact with the development group of the language variant of the operating system. For example, for VAX COBOL/Japanese on OpenVMS/Japanese, the Japanese COBOL development group needs to keep close ties with the Japanese OpenVMS development group.

Language RTLs

Currently, with each OpenVMS release the following language RTLs are shipped:

- High-Level Languages

  VAX Ada
  VAX BASIC
  VAX BLISS-32
  VAX C
  VAX COBOL
  VAX DIBOL
  VAX FORTRAN
  VAX MACRO
  VAX Pascal
  VAX PL/I
  VAX RPG
  VAX SCAN

- Interpreted Languages

  VAX DATATRIEVE
  VAX DSM

Modifying RTLs

Layered products groups wishing to include a new RTL or modify an existing one in a future version of OpenVMS should follow the procedure stated below:

1. Decide on the targeted release of OpenVMS. Discussions must begin with OpenVMS Engineering as early as possible to ensure the RTL change receives the proper attention.
2. Receive approval from the OpenVMS Project Manager of the targeted release. The Project Manager will understand the focus of this release and will be able to assess the impact of including or not including the RTL in the next release.
3. Understand what is and will be required of you in regard to meeting the OpenVMS release schedules. Contact either the Project Leader of an existing language (chances are they have gone through these procedures) or the Technical Project Leader of the targeted OpenVMS release.

Before following the above procedure, layered products groups wishing to include a new RTL or modify an existing one should **first** contact OpenVMS Engineering. The assignment of Project Leaders and Managers varies for each release of OpenVMS.

Using and Testing RTLs

Products shipping an RTL must remain in close contact with OpenVMS Engineering. Prior to making any changes to an existing RTL, OpenVMS Engineering should contact the parent development group. The development group must be aware of all changes during the release cycle of a particular OpenVMS version.

As each version of OpenVMS may include a variation of a shipped RTL, it is advised that any layered product shipping, and subsequently replacing, an RTL perform a OpenVMS version check to replace or not replace the RTL accordingly. Specific dependencies may exist with certain versions; it is therefore important to understand any changes made to the RTL in each OpenVMS version.

Rationale

To remain informed about decisions made by the OpenVMS Engineering group about routines used in the layered product, and to avoid unsupported dependencies.

Exceptions

None

Example

Not applicable

For more information

None

run_sep_instl

The Installation Verification Procedure (IVP) can be run as part of the installation or can be executed separately.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	AIX	HP-UX
SCO UNIX		

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

OpenVMS

The following sections are specific to OpenVMS layered products.

Description

The user should be able to invoke the Installation Verification Procedure (IVP) manually and separately from the installation procedure. The residual IVP command file should be placed in the SYS$TEST system directory.

If the IVP is an image or consists of more than one file, you should direct the file(s) to a subdirectory of SYS$TEST. The main command procedure residing in SYS$TEST should execute the files in the subdirectory.

Rationale

Providing a residual IVP permits the user to confirm the integrity of the product at any time.

Exceptions

None

Example

```
$ DIR SYS$COMMON:[SYSTEST]

FOO$IVP.COM
FOO.DIR

$ DIR SYS$COMMON:[SYSTEST.FOO]

FOO$RUN.EXE
FOO$MOD.DAT
FOO$IVPFOR.TST
```

ULTRIX, OSF, Sun

The following sections are specific to ULTRIX, OSF, and Sun layered products.

Description

The user must be able to invoke the IVP in two ways: as part of the installation and separately from the installation procedure.

For Sun, this functionality should be provided in the product's installation script.

For ULTRIX and OSF, the IVP should be placed in the subset control program file (<subset_name>.scp) where the ACT variable is equal to POST_L (that is, where actions take place after loading subsets).

The IVP must be invoked using the setld utility with the -v option.

In many cases the IVP cannot be executed during a Diskless Management Services (DMS) installation. In this situation, the layered product should take the following steps:

1. Include a message in the IVP prompt stating that the IVP cannot be executed during a DMS installation (see example in the Example section).
2. Ensure that the installation guide or release notes are updated to inform the user that the IVP cannot be executed during a DMS installation. Provide the user with instructions for executing the IVP using the command, setld -v [subset_name].

Rationale

Compliance with this requirement enables the user to ensure, either during the installation or any time thereafter, that the product has installed correctly.

Exceptions

None

Example

```
Would you like to run the IVP after the installation? [Y/N]

You may run the IVP separately using the command "setld -v [SUBSET_NAME]",
anytime after the product is installed.
```

This example shows sample IVP prompts.

```
The IVP cannot be executed during the DMS installation.
```

This example shows the statement displayed if the IVP cannot be executed during a DMS installation.

For more information

None

sample_install

The installation guide or an appendix of the installation guide contains a sample installation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

The installation guide or an appendix must contain a sample installation. If there are different installation paths possible, or if there are different kit components such as Run Time Only or Full Development, each sample installation should be included.

Rationale

Allows the user to examine a typical installation and see what type of information is needed for that installation. It also gives the user something against which to check the progress of their own installation.

Exceptions

None

Example

Not applicable

For more information Refer to Appendix C Reference Documents for pointers to Installation Guide templates.

save_iteration

Before changes are made to existing files, a copy of the original file should be made.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX		

PERFORMED BY: Software Engineer or Project Leader

REQUIREMENTS AID(S): None

Description

Before changes are made to an already existing file, the layered product should copy the file to filename.sav[x] (where x is the next copy of the file being saved).

Rationale

If an unplanned installation abort occurs, the .sav[x] file could be retrieved with minimal effort.

Exceptions

State this requirement is not applicable to layered products running on the V1.2 release of the OSF AXP operating system.

Example

If the product modifies a pre-existing system related file, such as /etc/fstab (UNIX), a copy of the original fstab should be made as "fstab.sav1."

For more information None

savesets_instl

If additional product components are to be installed separately from the main product, the installation procedure for each component must verify the existence of the main product and prompt the installer for each component to be installed.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer or Software Engineer

REQUIREMENTS AID(S): None

Description

There are many layered products that provide additional savesets as part of their kit. These additional savesets are usually optional components of the product.

The installation of these additional savesets can be accomplished either during the product installation, or as a separate installation. Which of the two methods is chosen is the decision of the development group.

During Product Installation

The main installation procedure should verify that the system has met all of the requirements of the component to be installed. In addition, the installation procedure should prompt the installer for each component to be installed. For example:

```
* Would you like to install the HCUIS portion of the kit [Y]?
```

The installation guide must state all preinstallation requirements for the main product and all of its components.

As a Separate Installation

If the additional components are to be installed separately from the main product, the installation procedure for each component must verify the existence of the main product as well as any additional system requirements. The installation guide must include discussions on and sample installations of each component. For example:

```
$ @VMSINSTAL VWSDEMO042 MUA0:

  VAX/VMS Software Product Installation Procedure V5.3-2
    .
    .
    .
```

Rationale

To avoid installation failures.

Exceptions

None

Example

Not applicable

For more information None

savesets_organization

Saveset A should contain the KITINSTAL.COM, the Release Notes, any files used to verify the system, and any files called by KITINSTAL.COM. Saveset A should remain small in size.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

The organization of your savesets is an important, and often overlooked, aspect of your installation. In general, the following guidelines should be adhered to for Saveset A and Savesets B through *N*:

Saveset A

This file should contain at least the following files:

- KITINSTAL.COM
- Release Notes
- Other files used to verify the system (or called by KITINSTAL.COM)

These files are associated with determining prerequisite information. If that information has not been validated, the remaining savesets should not be installed and the installation should abort.

Saveset A should remain relatively small in size.

Saveset B through Saveset N

There is generally no set rule for the number of savesets required. For those products providing many options during installation, each option could be designated its own saveset. Only the options chosen by the installer would then be installed.

Note

The BACKUP utility has an overhead of 100+ blocks when bundling a saveset. This needs to be taken into account for peak/net disk block usage.

Rationale

To prevent installation problems, since KITINSTAL.COM does not check whether restoration of Saveset A is complete.

Exceptions

None

Example

Not applicable

For more information None

setld

The product is installable with the setld utility.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OSF/1	ULTRIX
PERFORMED BY:	Software Engineer	
REQUIREMENTS AID(S):	None	

Description

When loading a layered product for the first time, invoke the setld utility by using the load (-l) option as follows:

```
# setld -l /dev/<device_name>
```

The /dev entry in the path name tells setld to look up the device name in the device (/dev) directory. The <device_name> indicates the logical name of the device on which the layered-product subsets are stored.

The following example shows the command with the device name rmt0h:

```
# setld -l /dev/rmt0h
```

Layered product developers should be careful to invoke setld as in the above example to ensure that the product's subsets are installed in the proper directory.

Note

A common mistake in invoking setld is adding the /usr path name to the command, as in setld /usr/dev/<device_name>; this command installs the files in the /usr directory, which means the product will not be able to support the dms and ris environments.

Rationale

Ensures consistent layered product installations.

Exceptions

None

Example

See examples within the description.

For more information

If You Need	Do the Following
More information on setld	Refer to the VAX Notes VIRTUE::SETLD_LPS conference.
More information on using setld with ULTRIX layered products	Refer to *ULTRIX Software Development*, Volume 2, *Guide to Preparing Software for Distribution on ULTRIX Systems.*

setld_kit_comp

The distribution kit contains all the files needed by the setld installation utility and uses the setld utilities to create product kits.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OSF/1	ULTRIX
PERFORMED BY:	Software Engineer	
REQUIREMENTS AID(S):	The setld utilities, REQ_CHECK	

Description

The setld utility must be used to create product kits. Therefore, the distribution kit must contain all the files needed by the setld installation utility.

Rationale

Using the setld utility and associated utilities ensures layered product consistency during product kit creation and installation.

Kits that are compatible with the setld utility will also be compatible with the Remote Installation Service (ris) utility and the Diskless Management Services (dms) utility. With the ris utility, a product can be installed on a server machine and distributed to other systems in a network. With the dms utility, a product can be installed into a diskless environment.

Exceptions

None

Example

Not applicable

For more information

- *Digital OSF/1 Guide to Programming Supplement Tools*, Section 8.0. (Order number: AA-PJUPA-TE)
- *ULTRIX Software Development*, Volume 2, *Guide to Prepearing Software for Distribution on ULTRIX Systems*
- VIRTUE::SETLD_ULTRIX_LPS VAX Notes conference

set_sys_param

System parameters and process quotas are neither set nor modified by the installation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer or Software Engineer

REQUIREMENTS AID(S): None

Description

Any system parameters required for installation must be included in the pre-installation section of the installation guide. The installation procedure should check to ensure the proper SYSGEN parameter values, but it must not alter any SYSGEN parameter values, nor any process quotas. SYSGEN parameters required after the installation for use of the product, should be included in the postinstallation section of the installation guide.

Rationale

Because customer environments are so varied, installations should prevent forcing undesirable changes to the system.

Exceptions

None

Example

Not applicable

For more information None

set_verify

The SET VERIFY command must not be included in any command file shipped by the product.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): REQ_CHECK

Description

See the requirement statement.

Rationale

To promote a user-friendly installation and work environment.

Exceptions

None

Example

Not applicable

For more information None

single_user_mode

An application should not require that the system be shut down to single-user mode for installation of the product.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OSF/1	ULTRIX	Sun
HP-UX	SCO UNIX	AIX

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Applications should not require a system to be shut down to single-user mode to install on that system.

Rationale

In a production environment, customers request that product installations have a minimal effect on normal operations. The need to shut down to single-user mode has an adverse effect on such environments.

Exceptions

Some products must interact with files in such a way that they could harm a system in multiuser mode. One such case is if a product must interact with a shared library on the system. These products may find it necessary to require single-user mode for installation.

Example

Not applicable

For more information None

snpc

The product team has contacted the Software New Products Committee (SNPC) to discuss operational details of product introduction, including licensing, Licensing Management Facility (LMF) implementation, and packaging.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Product Manager or Project Leader

REQUIREMENTS AID(S): None

Description

The Software New Products Committee (SNPC) is an advisory forum created to assist product groups with the complex process of introducing a software product into the international marketplace. The SNPC consists of representatives from worldwide licensing, service, manufacturing, engineering, and operational /administrative groups.

The SNPC can provide consulting and guidance in the following areas:

- Digital software licensing policies
- Digital software service policies
- Assignment of software model numbers for licenses, kits, and services in accordance with company software strategies and in compliance with *DEC STD 012-4 Unified Numbering Code (UNC) - Software Numbering Conventions*
- Technical implementation of Licensing Management Facility (LMF) support, if implemented.
- Inclusion of software model numbers in various worldwide pricing and operations systems
- The software introduction process and organizational introduction requirements

The SNPC can also introduce developers to other organizations and people who can provide help in a variety of areas, including the following:

- Licensing

- Pricing and service strategies
- Legal issues
- Royalty issues
- Export requirements
- Internationalization issues
- Technical requirements.

Rationale

The objective of the SNPC is to ensure product teams are supplied with the information and support necessary to successfully introduce a product package in conformance with Digital business practices.

Exceptions

Products licensed directly by third-party vendors.

Example

Not applicable

For more information

For more information on the SNPC, contact MRKTNG::SNPC.

spd_ssa

All layered products must have an Software Product Description (SPD) and System Support Addendum (SSA).

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Product Manager

REQUIREMENTS AID(S): None

Description

The Software Product Description (SPD) and the System Support Addendum (SSA) are legal documents that describe a software product and detail the technical environment in which it will be supported.

An SPD is a description of the product's base features and is written by the product development group. The information contained in the SPD should be valid everywhere the product is used. Features or business policies that are applicable only to a specific country or region should not be described in the SPD; these are described in the SSA. For example, the SPD should not describe Digital's license and service policies unless they are applicable in every country in which the product is being distributed.

An SSA is a description of the technical environment in which the product is supported. It is written by the product development group. As with the SPD, the information in the SSA should be valid in the international marketplace. It should also describe country-specific information. For example, the SSA should list any local country constraints, additional hardware or software options, and distribution media availability.

Both the SPD and SSA are revised during the life of the product to reflect changes in the product status, such as new releases of the product, changes to hardware or software supported, and changes to available options.

The development group is responsible for all revisions to the SPD. The development group and local engineering groups are both responsible for revisions to the SSA. The development group is responsible for providing a new SSA at release time and for updating it between product releases to indicate additional support for processors or new versions of the operating system. The local

engineering groups are responsible for keeping any country-specific information up-to-date.

Rationale

The SPD and SSA provide the information necessary to buy, sell, and support the product. They also provide the purchaser with a legally binding statement of the specific features and environment that Digital has committed to support in accordance with the corporation's standard terms and conditions.

Exceptions

None

Example

Not applicable

For more information For more information on the SPD/SSA, Workbook is available on VTX SPD.

standard_notices_source

Standard notices must be placed in all software sources produced by Digital.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): *DEC STD 197-0*

Description

All the standard legal notices (copyright, government data rights, and license notices) listed in Section 2.2 of *DEC STD 197-0 Legal Requirements and Guidelines for Digital Publications and Software*, must be coded into all software modules, shell scripts, batch and command files developed by Digital. The standard notices must appear in both the source code and executable code even if there is no plan to release a source kit or listings.

Rationale

Use of appropriate legal notices and labels protects the security and ownership of Digital's intellectual property.

Exceptions

If there are exceptions to this requirement, contact Digital's Law department.

Example

Refer to *DEC STD 197-0 Legal Requirements and Guidelines for Digital Publications and Software* for examples.

For more information

Refer to *DEC STD 197-0 Legal Requirements and Guidelines for Digital Publications and Software* for further information.

std_unix_env

All products tested in the standard UNIX-based customer environment, that is, YP, BIND, DECnet, TCP/IP, DMS (if applicable), and RIS.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OSF/1	Sun	ULTRIX
AIX	SCO UNIX	HP-UX

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

All layered products must support the protocols, utilities, and services of the standard UNIX operating environment. The requirements differ depending on the operating system.

Rationale

It is expected that a layered product will work correctly in a typical customer environment.

Exceptions

None

Example

ULTRIX

Yellow Pages (yp)
Berkeley Internet Name Domain (bind)
Shell Support
Transmission Control Protocol/Internet Protocol (TCP/IP)
DECnet–ULTRIX
Diskless Management Services (DMS) - if applicable
Remote Installation Service (RIS)

OSF/1

Transmission Control Protocol/Internet Protocol (TCP/IP)
Network Information Service (NIS)
Remote Installation Service (RIS)
Berkeley Internet Name Domain (bind)

Sun

Transmission Control Protocol/Internet Protocol (TCP/IP)
Berkeley Internet Name Domain (bind)
Yellow Pages (yp)
Shell Support

Layered products must support Yellow Pages (yp).

Yellow Pages (yp) is a network data base service that maintains a set of data bases and propagates updated data bases among the systems on the network, ensuring consistency.

Layered products must support the Berkeley Internet Name Domain.

The Berkeley Internet Name Domain (bind) is a server-based lookup service that allows client systems to obtain host names and addresses.

Layered products must support one of the indicated shells.

The ULTRIX operating system supports the Bourne family of shells (sh and sh5). ULTRIX V4.0 and later releases support the Korn shell (ksh) in addition to the Bourne shells. Support should preferably be for sh5 or ksh.

The OSF/1 and SunOS operating systems support the Bourne shell sh (which is equivalent to ULTRIX sh5) and the C shell csh.

Layered products should be tested for support of TCP/IP.

If a layered product uses the TCP/IP network protocol, that use should be tested. The following example shows an Installation Verification Procedure (IVP) that illustrates a test for a layered product's support of TCP/IP. In the following example, the use of the single colon (outfld:0) in response to the "Connection Identifier (display)?" prompt represents the use of the TCP/IP transport to display the IVP to a workstation screen.

```
TCP/IP EXAMPLE (example of error found)
       will not display on node outfld

#  setld -v DECQUAL210

Running the DEC QUALITY on ULTRIX V2.1 Installation Verification Procedure

Enter the QUALITY workstation type to be used.
VT125, VT_OUTPUT, VT240, VT3XX, TEKTRONIX
TEK_OUTPUT,  TEK4107, TEK4107_OUTPUT, DECWINDOWS
VS35240, VS3540

Workstation Type? DECWINDOWS
Connection Identifier ( display )? outfld:0
%QUALITY-E-ERROR_026, Specified workstation cannot be opened in routine
popenqual
%DECQUALITY-F-ERROR_NEG_154, Internal QUALITY error: Unable to configure default
workstation type in routine popenqual
```

```
You may also run the IVP on your workstation by doing the following:
setld -v DECQUAL210

End of DEC QUALITY on ULTRIX V2.1 Installation Verification Procedure

DEC QUALITY on ULTRIX Installation Procedure completed
        EXAMPLE OF IVP RUN LOCALLY (works/displays correctly)
```

Layered products should be tested for support of DECnet–ULTRIX.

If a layered product uses the DECnet protocol, that use should be tested. The following example shows an IVP that illustrates a test for a layered product's support of DECnet. In the following example, the use of the double colon (outfld::0) in response to the "Connection Identifier (display)?" prompt represents the use of the DECnet transport to display the IVP to a workstation screen.

```
DECNET EXAMPLE (no error messages displays
                correctly on node outfld)

# sh
# setld -v DECQUAL210

Running the DEC QUALITY on ULTRIX V2.1 Installation Verification Procedure


Enter the QUALITY workstation type to be used.
VT125, VT_OUTPUT, VT240, VT3XX, TEKTRONIX
TEK_OUTPUT,  TEK4107, TEK4107_OUTPUT, DECWINDOWS
VS35240, VS3540

Workstation Type? DECWINDOWS
Connection Identifier ( display )? outfld::0

You may also run the IVP on your workstation by doing the following:
setld -v DECQUAL210

End of DEC QUALITY on ULTRIX V2.1 Installation Verification Procedure

DEC QUALITY on ULTRIX Installation Procedure completed
#
```

Layered products should demonstrate the capability of being installed on the server processor in the standard DMS environment.

Diskless Management Services (DMS) is an ULTRIX utility that allows a diskless client node to transparently use software on a server node. The diskless client is booted from the server and has no system management responsibilities because the software remains located on the server, where all system management takes place. Using DMS, you can install diskless product environments and register diskless clients from a common server machine.

In the DMS environment, the /usr area of a particular root.vax or root.mips directory is shared by all clients registered for that environment. This shared area is read-only. Therefore, any type of write operation by the application to this area will result in failure.

To set up a product to run on a diskless machine as if it were installed locally, create a DMS area on the server machine and then use the setld command to install the product on the server.

If the product cannot be tested in a standard DMS environment, a DMS environment can be emulated on one ULTRIX machine, using the following steps.

Emulating a DMS Environment

1. Bring the system down to single-user mode by executing the following command:

   ```
   /etc/shutdown now
   ```

2. Modify the /etc/fstab file. This file controls the automatic mounting of file systems. The reference to the /usr file must be modified to be read-only, for example,

   ```
   /dev/rz0g:/usr:rw:1:2:ufs::
   ```

   should be modified to be

   ```
   /dev/rz0g:/usr:ro:1:2:ufs::
   ```

3. Unmount the file systems using the following command:

   ```
   unmount -a
   ```

4. Remount the file systems by using the following command:

   ```
   mount -a
   ```

5. Bring the system backup to multiuser mode using the following command:

   ```
   Ctrl/D
   ```

The system is now in emulation mode. In emulation mode, users cannot send mail or print files because these functions require writing to the /usr file, which is now forbidden.

Ending Emulation Mode

When finished using emulation mode, repeat steps 1 through 5 to return the system to the normal operating mode. In step 2, be sure to reset the /etc/fstab file to allow read/write access, as follows:

```
/dev/rz0g:/usr:ro:1:2:ufs::
```

should be modified to be

```
/dev/rz0g:/usr:rw:1:2:ufs::
```

Layered products should successfully demonstrate the capability of being installed using the Remote Installation Service (RIS) utility.

Remote Installation Service (RIS) is an ULTRIX utility that installs software subsets from a server to a client node over the network. To test this capability, install the layered product in the RIS area of the server machine. Then execute the setld command from the client node to install the product subset over the network from the server to the client. If the installation runs successfully, setld displays all the subsets from the RIS area on the server. The installation is completed by selecting the layered product's subsets from this display and storing them on the client node.

For more information

If You Need	Do the Following
More information on building kits for the ULTRIX environment	Refer to the *ULTRIX Software Development*, Volume 2, *Guide to Preparing Software for Distribution on ULTRIX systems*

If You Need	Do the Following
More information on DMS	Refer to the *ULTRIX System Management*, Volume 3, *Guide to Diskless Management Services*
More information on RIS	Refer to the *ULTRIX Reference Pages*
More information on yellow pages (yp)	Refer to the *ULTRIX System Management*, Volume 3, *Guide to the Yellow Pages Service*
More information on the Berkeley Internet Name Domain (bind)	Refer to the *ULTRIX System Management*, Volume 3, *Guide to Bind Service*

subset_check

During installation, checks for required subset dependencies in the .scp file are performed.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OSF/1	Sun	ULTRIX
PERFORMED BY:	Software Engineer		
REQUIREMENTS AID(S):	REQ_CHECK		

Description

Each subset must have a subset control program (.scp) file containing dependency checking and configuration information for the files in a subset. Sun products must do dependency checking for SunOS software categories from within their installation scripts.

Rationale

Use of dependency checking routines provides added assurance of installation accuracy.

Exceptions

None

Example

Not applicable

For more information

- *Digital OSF/1 Guide to Programming Supplement Tools*, (Order Number: AA-PJUPA-TE)
- *ULTRIX Software Development*, Volume 2, *Guide to Preparing Software for Distribution on ULTRIX Systems*
- VIRTUE::SETLD-ULTRIX-LPS VAX Notes conference
- KEPERA::SUN_USERS VAX Notes conference

successful_del_reinstl

The layered product can be deleted and reinstalled successfully.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OSF/1 Sun ULTRIX

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): REQ-CHECK on ULTRIX and OSF/1

Description

A product must have the capability to be deleted and reinstalled. On the ULTRIX and OSF/1 operating systems, this capability must be implemented using the system utility setld. For products installing on the Sun operating system, the product's installation script must handle these tasks.

Rationale

Customers request the ability to remove all files associated with a product.

Exceptions

None

Example

```
setld -d DECQUAL100
Deleting DEC Quality V1.0 (DECQUAL100)
```

This example shows the setld utility being used to delete and reinstall the product DEC Quality V1.0.

For more information

If You Need	Do the Following
Assistance using setld	Refer to the man page for setld or Refer to *Preparing Software for Distribution on ULTRIX Systems*
Hints on installing on the Sun operating system	Refer to the KEPERA::SUN_USERS VAX Notes conference.

sysgen_param_change

The installation procedure does not alter any of the security-related SYSGEN parameters, nor does it suggest that the customer alter them.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): REQ_CHECK

Description

Security policies at the customers' sites may stipulate what these system parameters should be. Therefore you should avoid modifying these parameters governing customers' security policies:

- PQL_MPRCLM
- LGI_RETRY_LIM
- LGI_RETRY_TMO
- LGI_BRK_TERM
- LGI_BRK_DISUSER
- LGI_BRK_LIM
- LGI_BRK_TMO
- LGI_HID_TIM
- LGI_PWD_TMO
- MAXSYSGROUP
- RMS_FILPROT

Additionally, layered product groups must not modify SYSGEN parameters or UAF quotas during installation.

Rationale

To be in keeping with Digital security guidelines.

Exceptions

None

Example

None

For more information See also the *Security Standards for ULTRIX and OpenVMS Layered Products*.

sys_param_change_instr

The postinstallation section of the installation guide contains instructions for modifying the system parameters to run the product or to ensure optimal system performance.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

The postinstallation section of the installation guide should include requirements for system parameters needed to run the product or to ensure optimal system performance. The installer can be referred to the preinstallation section of the installation guide for instructions on modifying these parameters. As in the preinstallation section, system parameters should be modified in SYS$SYSTEM:MODPARAMS.DAT via SYS$UPDATE:AUTOGEN.COM.

Rationale

To ensure optimal system performance after product installation.

Exceptions

None

Example

System Parameters

The following table shows the minimum values to which certain system parameters should be set when users develop DECforms applications on your system.

System Parameter	Minimum Value
PQL_MBYTLM	16384
PQL_MPRCLM	3
MAXBUF	4096

To change system parameters, do the following:

1. Edit the file SYS$SYSTEM:MODPARAMS.DAT.

   To change the value of a parameter listed in the MODPARAMS.DAT file, use an editor to locate the line containing that parameter. Delete the current value associated with that parameter and enter the new value.

   To add a new parameter and its value, add a line to the MODPARAMS.DAT file. Make sure you include both the name of the parameter and its value in the new line. For example, to include a setting for the PQL_MBYTLM and PQL_MPRCLM parameters, add the the following lines to the MODPARAMS.DAT file:

   ```
   PQL_MBYTLM=16384
   PQL_MPRACLM=3
   ```

2. Once you modify the file and exit the editor, run the AUTOGEN procedure to recalculate your system parameters. Enter the following command:

   ```
   $ @SYS$UPDATE:AUTOGEN GETDATA REBOOT
   ```

   AUTOGEN does an automatic system shutdown and reboots once it has finished. Rebooting your system activates the new parameter values. For more information about using AUTOGEN, see the instructions on modifying system parameters in the OpenVMS documentation on system management and operations.

For more information None

sys_startup_instr

The postinstallation section of the installation guide contains instructions for editing the system startup files.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

Instructions for editing the system startup and shutdown files should be included in this section. Editing the system startup and shutdown files provides for automatic startup and shutdown of the layered product.

For international products, be sure to include the startup command procedure name for each product variant.

Also include in this section any specific requirements for command line placement in the system startup/shutdown file. For example, if the product requires the network to be running, the installer should place the product startup command line in SYS$STARTUP:SYSTARTUP_V5.COM after the network startup command line. The product shutdown command line should be placed in SYS$MANAGER:SYSHUTDWN.COM.

This section may also contain any other relevant information on layered product startup. If your product has a DECwindows interface, you may also include instructions for starting DECwindows. Note that from OpenVMS V5.3 onward, there is no need to add @SYS$MANAGER:DECW$STARTUP.COM into SYSTARTUP_V5.COM.

Rationale

Allows for automatic startup and shutdown of the product when the system is rebooting.

Exceptions

None

Example

Editing the System Startup File

To provide for automatic startup of DECforms when your system is rebooted, you must edit the system startup file. You also might want to add the command to run a command procedure that defines output formats for dates and times other than the standard OpenVMS format.

Add the command line that starts DECforms to the system startup file, SYS$STARTUP:SYSTARTUP_V5.COM. Put this command line with the command lines that set up other products. These command lines follow the network startup command line, as follows:

```
$ @SYS$MANAGER:STARTNET.COM
    .
    .
    .
$ @SYS$STARTUP:FORMS$STARTUP.COM
```

For more information None

sys_tuning

The postinstallation section of the installation guide contains information for system tuning and installing the product as a shared image.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer
REQUIREMENTS AID(S): None

Description

This section should include information on system tuning and instructions for installing the product as a shared image:

- Tuning the system—After installing the layered product, the installer may want to adjust the system to enhance performance or to lower the use of certain system resources. Include the appropriate information in this section.
- Installing the product as a shared image—If the product is to be used extensively on the system, advise the installer to reduce system overhead and memory requirements by installing the product as a shared image. Global pages and global sections will need to be increased when installing an image. This section must contain the required values for those parameters. The installer can be referred to the preinstallation section of the installation guide for instructions on modifying the values. Instructions for installing the product as a shared image must also be included.

Resident Images

In addition, there is a new feature in OpenVMS (Alpha AXP) called “resident images.” A resident image is allocated contiguous memory permanently. This allows the system manager to improve system performance on systems with plenty of memory. If the product uses this feature, it should be documented in the installation guide.

Rationale

Allows the system manager to improve system performance.

Exceptions

None

Example

Installing the Form Development Utilities as Shared Images

DECforms provides the following form development utilities:

- Form Development Environment (FDE)
- Panel Editor
- IFDL Translator
- Back Translator
- Extract Utility
- Test Utility
- TDMS Converter
- FMS Converter

If you use these form development utilities extensively on your system, you can install them as shared images to reduce the system overhead and memory requirements.

To install the form development utilities as shared images, do the following:

1. Determine the number of available global pages and global sections on your system. For information on how to verify and modify the number of global pages and global sections, see the preinstallation section on system parameters.

   To install the form development utilities as shared images, you must increase the number of global pages by 4100 and the number of global sections by 23. These are increases over what is required during installation. Install the utilities just after you start your system. The available space in the global page table is less likely to be fragmented.

2. To install the form development utilities as shared images on a currently running system, enter the following command:

   ```
   $ @SYS$STARTUP:FORMS$STARTUP.COM DEVELOP
   ```

   To install the utilities as shared images each time your system is rebooted, add the DEVELOP parameter to the line you added to the system startup procedure, SYS$STARTUP:SYSTARTUP_V5.COM, as follows:

   ```
   $ @SYS$MANAGER:STARTNET.COM
       .
       .
       .
   $ @SYS$STARTUP:FORMS$STARTUP.COM DEVELOP
   ```

To install selected form utilities, use the OpenVMS Install Utility (INSTALL). For example, to install only the Panel Editor as a shared image, enter the following commands:

```
INSTALL> ADD SYS$SYSTEM:FORMS$FSESHR.EXE  /OPEN /SHARED / HEADER
INSTALL> ADD SYS$SYSTEM:FORMS$PEDSHR.EXE  /OPEN /SHARED / HEADER
INSTALL> ADD SYS$SYSTEM:FORMS$MSGPRDSHR.EXE  /OPEN /SHARED / HEADER
INSTALL> ADD SYS$SYSTEM:FORMS$PEDVTEXEC.EXE  /OPEN /SHARED / HEADER
```

You must install FORMS$FSESHR.EXE and FORMS$MSGPRDSHR.EXE with the first form development utility you install. Once you have installed these two images, you need not reinstall them to add other form development utilities.

For more information None

third_party_products

Products that are associated with a third-party contract must comply with DEC STD 038-1 System Evaluation of New Products - Software and DEC STD 197-0 Legal Requirements and Guidelines for Digital Publications and Software.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Product Manager

REQUIREMENTS AID(S): None

Description

Third-party products (that is, products that Digital develops concurrently with external vendors), Digital buyouts, or any other Digital product that is associated with a third-party contract, must comply with *DEC STD 038-1 Systems Evaluation of New Products - Software*.

Digital's legal department must be contacted when any contractual agreement is entered into with a third party. This is to ensure the following:

- The Digital representative is provided legal advice/support when negotiating a third-party contract.
- Compliance with *DEC STD 197-0 Legal Requirements and Guidelines for Digital Publications and Software* (section 2.5 Notices for Third-Party Products)

Rationale

To maintain consistency among all Digital products and to ensure that Digital's quality requirements are incorporated into the product.

Exceptions

If for some reason, Digital requirements cannot be met by the external vendor, the development group should contact their Software Quality Operations and Information Services (SQOIS) representative and the Software Quality Assurance (SQA) organization (GSFSYS::SQA) during the phase "0" time frame.

Example

Not applicable

For more information None

unix_image_header_id

Layered products must uniquely identify what version of an image a customer is using.

OPERATING SYSTEMS (ALL ARCHITECTURES):

ULTRIX	SCO UNIX	OSF/1
Sun	AIX	HP-UX

PERFORMED BY: Software Engineer or Project Leader

REQUIREMENTS AID(S): None

Description

Currently there is no documented way describing how to identify or update an image ID header for Digital's UNIX products. This requirement provides guidelines outlining the format of the image ID header and basic information that must be in the image ID header.

Because of possible syntactical conflicts concerning the UNIX command and it's interaction with files from different revision controls systems, that is, SCCS, RCS, specific syntax guidelines for the image ID header will not be given here.

The image ID header is an initialized character array. For major and point releases the first three pieces of information contained in this character array must be:

- image name
- image revision level
- build date

Note

The three items listed above are managed by the software revision control systems, that is, RCS, SCCS.

If the image was revised and is part of an ECO or MUP, the fourth, fifth, sixth, and seventh pieces of information contained in the Image_ID character array MUST be:

- formal product name of which this image is a part
- the name of the engineer who made the update

- the e-mail address of the engineer who made the update
- the name of the organization the engineer belongs to

Additional information can be added to the image ID header at engineering's discretion.

The command to type to display the image ID header would be:

```
what image_name
```

Note

Some development groups may not rebuild all images associated with their product for every release of the product. For such products, it is recommended that the image ID header not be changed for images not revised for a particular release.

Rationale

The image ID header will expedite the debugging process, the *DEC STD 204-0 Software Correction and Distribution* process, and help Digital Services by providing a finer granularity in determining specific image information for a software product.

Exceptions

None

Example

The following is an example image_id header for a major or point release:

```
#ifndef lint
 char decquality_id[] ="@(#)decimage.c    5.1    4/20/93  ";
#endif                          |          |       |
                          image name    image    build
                                        revision  date
                                        level
```

The following is an example image_id header for a ECO or MUP release:

```
#ifndef lint
char dec_id[] ="@(#)decimage.c 5.1 12/20/92 DECQuality John Doe abc::doe stt=";
#endif                 |      |       |       |         |     |         |
               image name   image    build  product   eng    eng       org
                            revision  date   name      who    e-mail
                            level                      update address
```

For more information Reference the UNIX man page describing the "what" command.

unsup_vms_interfaces

If a layered product requires features that are available through an unsupported internal interface, you can submit a request to the OpenVMS development group to use the internal interface.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

An unsupported OpenVMS interface is one for which the OpenVMS operating system has no documented code.

It is recommended that all OpenVMS layered products use supported and documented interfaces of the OpenVMS operating system. However, if your layered product requires features that are available through an unsupported internal interface, you can request to use the internal interface.

Procedure to Submit Request

Submit OpenVMS Phase 0 Requests directly to OpenVMS product management. Follow these steps to obtain the use of an unsupported internal OpenVMS interface:

1. Ensure that the features needed are not available through supported interfaces by researching the appropriate OpenVMS documentation.
2. Submit an OpenVMS Phase 0 Request to the OpenVMS development group for a supported method of obtaining the features needed.
3. Once these conditions have been met, layered product groups can request to use an unsupported internal OpenVMS interface. The request should be made during Phase 0/1 of the layered product development cycle.
4. To make a request, layered product groups should contact the OpenVMS development group to discuss the proposed use of the interface, supported alternatives, and potential risks.
5. If the OpenVMS development group agrees that the proposed use of the interface is appropriate, provide Software Quality Operations and Information Services (SQOIS) with information regarding the use of the interface.

Receiving permission to use an unsupported internal interface does not imply or constitute support. Rather, it allows the OpenVMS development group to track the use of these interfaces by layered product groups. The OpenVMS development group is also able to provide notification to the appropriate layered product groups whenever one of these interfaces is modified.

Rationale

Support for unsupported OpenVMS interfaces may be denied in the future, hence the recommendation to use all supported OpenVMS interfaces.

There may also be legal implications involved in supplying customers with unsupported interfaces.

Exceptions

None

Example

```
SUBMITTOR:      [person/group making the request]

ABSTRACT:       [short description of the request]

DESCRIPTION:    [detailed description of the request]

SCHEDULE:
Controller specification available on DD-MMM-YY
Controller prototype available to OpenVMS DD-MMM-YY
First customer test of media DD-MMM-YY
First customer ship DD-MMM-YY

BENEFIT:
[short statement of benefit to Digital with substantiating data]

IMPACT OF NOT  GRANTING REQUEST:
[impact to Digital if request is turned down]

JUSTIFICATION:
[any other reason for doing]

RATING:
[desired, required, or critical]

ISSUES:
[statement of risk either to schedule or content]

SUPPORTING DOCUMENTS:
[any documents that add detail to the request]
```

The above example shows the form to use when submitting your request to the OpenVMS development group.

For more information None

user_filesys_loc

The user has been given the option to specify the location where the application is to be installed.

OPERATING SYSTEMS (ALL ARCHITECTURES):

Sun

PERFORMED BY: Project Leader or Software Engineer

REQUIREMENTS AID(S): None

Description

Before installation of a product on the system, the user should be asked to specify the path where the application files should be installed. It is required that applications use environmental variables to access product functionality, rather than use links in generic system directories such as /usr/bin or /usr/lib. This may require significant set up by installers to incorporate the correct search paths for users. Necessary configuration information, or an automated configuration script, should be provided to simplify this task. As an additional option, applications may provide the ability to create links from system directories, but this should not be required. The installation procedure should then ensure that the product functions correctly when it is placed in the installer specified location.

Rationale

In a SunOS environment, users prefer to have the flexibility of placing layered products in user-specified directories. This allows them to use their system resources in the most efficient manner. To support distributed environments, links should not be utilized by a product. Environmental variables allow users to efficiently integrate layered products into diverse user environments.

Exceptions

None

Example

```
DEC Quality for Sun
c Copyright Digital Equipment Corporation

DEC Quality for Sun will install its files in the /usr directory unless
you specify otherwise.

Where would you like DEC Quality for Sun to install
      its files [/usr]:  /prod_kits/DECqual
Would you like links placed in the /usr area? [yes]: no

DEC Quality for Sun will install its files in the /prod_kits/DECqual
directory.  The environmental variable DECQUALHOME points to this location.
Refer to the installation guide for proper configuration of your
system after installation.
```

In this example, the user has specified an alternate directory for the files in DEC Quality for Sun and has indicated that links should not be placed in the /usr area. The user has pointed to the installation guide for further information about set up with the environmental variable DECQUALHOME. Real product examples include OpenWindows and CDA Run Time Services for Sun SPARCstation.

For more information None

user_priv_instr

The postinstallation section of the installation guide contains instructions for modifying any necessary user account privileges and quotas.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

This section should include any privileges and account quotas necessary for using the product. Instructions for setting up user accounts must also be included.

Rationale

To ensure proper functioning of the application on the system.

Exceptions

None

Example

User Account Requirements

To use DECforms, user accounts on your system must have the TMPMBX and NETMBX privileges. The accounts must also have the quotas shown in the following table.

Account Quota	Minimum Value
BYTLM	16384
PRCLM	3

User account privileges and quotas are stored in the file SYSUAF.DAT.

To verify and change user account privileges and quotas, use the AUTHORIZE utility as follows:

1. Set your directory to SYS$SYSTEM and run the AUTHORIZE utility:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF>
```

2. At the AUTHORIZE utility prompt (UAF>), to check a particular account, enter the SHOW command with an account name. For example:

```
UAF> SHOW SMITH
```

3. To change a privilege or quota, enter the MODIFY command in the following format:

```
UAF> MODIFY ACCOUNT_NAME /PRIVILEGES=PRIVILEGE_NAME/QUOTA_NAME=NNNN
```

   The following example adds the NET MBX privilege, modifies the BYTLM quota for the SMITH account, and exits the utility:

```
UAF> MODIFY SMITH /PRIVILEGES=NETMBX /BYTLM=16384
UAF> EXIT
```

   After you exit the utility, the OpenVMS operating system displays messages indicating whether changes were made. Once you have made the changes, users must log out and log in again for the new privileges and quotas to take effect.

For more information on modifying account privileges and quotas, see the *OpenVMS Authorize Utility Manual*.

For more information None

usr_var_conventions

The following directory naming conventions for /usr and /var are recommended: /usr/kits/productcode_productiversion for read-only operations, and /usr/var/kits /productcode_productversion for read/write operations.

OPERATING SYSTEMS (ALL ARCHITECTURES):

ULTRIX

PERFORMED BY: Software Engineer/Project Leader

REQUIREMENTS AID(S): None

Description

The /usr file system should contain the read-only files for a product. The /var file system should contain the read/write files for the product.

Rationale

In the Diskless Management Services (DMS) environment, the /usr filesystem is shared by all DMS clients and is therefore a read-only area. Attempts by the DMS client to write to this area result in failure of the intended operation.

Exceptions

None

Example

Not applicable

For more information

For ULTRIX products, refer to the *Guide to Disk Maintenance.*

For OSF products, refer to the *Guide to Programming Support Tools.*

vaxcluster_env

The product should be tested in a VAXcluster environment, including executing Installation Verification Procedure (IVP) on nodes other than those where the product was installed.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

You must fully test your product in all supported VAXcluster environments as stated in the System Support Addendum (SSA). In the simplest case, this testing is accomplished by installing the product on NODE_A and executing the startup procedure(s), and IVP on NODE_B. More complex products, such as cluster performance software, can require boundary testing by adding nodes, queues, or processes to validate the product in this environment. In general, layered-product groups must assess the individual dependencies in a clustered environment and design scripts accordingly. The minimal amount of testing should include that which is outlined in the simplest case.

Rationale

To ensure that the product functions correctly on all nodes of a cluster after installation in a VAXcluster environment.

Exceptions

None

Example

Not applicable

For more information None

vaxcluster_instr

The postinstallation section of the installation guide contains instructions for licensing, starting, and running the product on a VAXcluster.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer
REQUIREMENTS AID(S): None

Description

The postinstallation section of the installation guide should include any instructions for running the product on a VAXcluster. It should state that a LICENSE LOAD must be performed on every node in the VAXcluster to run the product in that VAXcluster. This is done by using the Digital Command Language (DCL) command LICENSE LOAD with the appropriate license Product Authorization Key (PAK) information.

This section should also include other actions that must be performed by the installer on other nodes of the cluster after the installation. It should include the replacement of DCL tables and the execution of the product startup command procedure.

Rationale

To ensure that all requirements necessary to run the applications are fulfilled.

Exceptions

None

Example

Cluster Considerations

After you install DECforms on a VAXcluster system, you must do the following tasks on each licensed node in the VAXcluster system:

- Run the DECforms startup command procedure.
- Install DCLTABLES.EXE.
- Register the license for DECforms.

- Run the full development kit Installation Verification Procedure (IVP) (if you are installing the full development kit) and the run time kit IVP.

To run the startup command procedure, log into a node in the VAXcluster system and enter the following command:

```
$ @SYS$STARTUP:FORMS$STARTUP.COM
```

To install DCLTABLES.EXE, enter the following command:

```
$ INSTALL REPLACE SYS$SHARE:DCLTABLES.EXE/OPEN/SHARE/HEADER
```

To register the DECforms license on a node, see the preinstallation section on registering a license. To run the full development kit and run time kit IVPs, see the IVP section.

For more information None

vaxcluster_req

The preinstallation section of the installation guide lists any VAXcluster requirements that need to be satisfied prior to the installation.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

If applicable, VAXcluster requirements that must be satisfied before the product installation should be included in preinstallation section of the installation guide.

Rationale

To ensure that all requirements are met before the installation starts, if the product runs on a VAXcluster.

Exceptions

This requirement is waived for Alpha AXP V1.0.

Example

None

For more information None

vaxcluster_support

Recommendations to support the full cluster model have been taken into consideration.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer
REQUIREMENTS AID(S): None

Description

It is strongly recommended that you use the full support cluster model. The following list contains the software characteristics that adhere to this standard:

- A single installation makes the software available to all nodes that access that particular system disk. It is acceptable to execute startup or configuration procedures outside of the installation to fully enable the product. With the advent of very large clusters, it becomes much too cumbersome to require separate installations.
- Software is expected to be available through a single installation per system disk; it follows that in a heterogeneous environment (more than one system disk), you must install the software as many times as there are system disks in the cluster, to make it available cluster-wide.
- The run time execution of a product must work properly in a clustered environment. For example, if a product updates a file common to all accessing nodes, then concurrent write operations must be planned for, and they must exhibit known and acceptable behavior. No run time restrictions can exist for products adhering to the full support cluster model; this includes working in a mixed-version environment.
- Design the installation procedure to be independent of the processor type. Even if a subset of processors is supported, the installation must be generic in this respect.
- The installation procedure must place files in common directories. Layered product files placed in specific directories will not be available to any other nodes except the installation node.
- Placement of files in specific directories must occur outside the installation procedure. If several commands are required, the layered product must ship a command procedure to be executed on each enabled node to minimize the customer's effort. Often during the installation a template file is placed

in a common area. If this occurs, then during the node-specific phase the command procedure accesses the template and transforms it for specific use.

- Configuration of specific devices must also occur outside the installation procedure with a shipped command procedure.

Rationale

To promote consistency across Digital products. Also, customers expect it.

Exceptions

None

Example

Not applicable

For more information None

vmsinstal_checks

The preinstallation section of the installation guide lists the major VMSINSTAL checks.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer

REQUIREMENTS AID(S): None

Description

This section should state that VMSINSTAL checks the following:

- If the user is logged in to a privileged account.
- If any other processes are running on the system.
- If the following minimum account quotas are met:

  ASTLM = 24
  BIOLM = 18
  BYTLM = 18,000
  DIOLM = 18
  ENQLM = 30
  FILLM = 20

Rationale

To ensure a successful product installation.

Exceptions

None

Example

Not applicable

For more information None

vms_tailor_class

The minimum OpenVMS tailoring classes required for the product have been assessed and tested, and they are consistent with those documented in the System Support Addendum (SSA) and the installation guide.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Software Engineer

REQUIREMENTS AID(S): None

Description

Assessing OpenVMS Class Support

All OpenVMS layered products must state support for the base or required saveset, because these classes are non-tailorable and reflect minimal OpenVMS capability; other classes can consist of OpenVMS files needed by some layered products and unnecessary to others. The following list guides you in assessing which classes are required for your product:

- If the product is new, the project leader should review the listing of complete OpenVMS files categorized by class in the OpenVMS file SYS$UPDATE:VMSKITBLD.DAT. The project leader should be familiar enough with the product to identify any file dependencies. Note that if only one file in a class of a hundred files is required by the product, then the entire class must be selected.
- If the product has prerequisite software, the product must support at least those classes needed by prerequisite software. It does not make sense to allow a site to delete files required by any prerequisite software. The product can require additional classes above and beyond prerequisite software support; however, it will never require fewer classes.

Testing can begin once you approximate class support.

If you complete this testing during product iteration, it need not be repeated for maintenance updates of the product. However, when your product offers new features, accesses new OpenVMS components, or is substantially modified, you should repeat tailoring testing to ensure that class support (as stated on the SSA) is current and reflects actual OpenVMS dependencies.

To date, layered-product groups have found making this assessment and completing required testing a straightforward procedure. Layered-product development teams often pose the following question: "How can I use VAX DEC/Test Manager (and the classes it requires) to run regression tests without introducing a bias on the system?" If you wish to use DEC/Test Manager (DTM) on a tailored system to run regression tests, refer to the For More Information section at the end of this requirement description.

OpenVMS Classes

OpenVMS classes are mutually exclusive subsets of OpenVMS files that provide a mechanism by which customers can include only site-specific OpenVMS files on their systems. This frees up space on the system disk and results in performance and maintenance gains. It also tailors the general all-purpose nature of the OpenVMS operating system to specific customer needs. The following table provides a complete list of these classes; every OpenVMS file is included in one, and only one, of these categories.

Class	Description
BASE:	Files required to boot OpenVMS. Includes Cluster Support, OpenVMS Device Drivers, and System Management Files. Also called the OpenVMS Required Saveset.
NET:	Network Support
PROG:	Programming Support
SYSP:	System Programming Support
USER:	Secure User's Environment
UTIL:	Utilities
RMSJ:	RMS Journaling Files
WKST:	OpenVMS Workstation Support
OPTL:	Miscellaneous Files
BLIS:	BLISS Require Files
XMPL:	Example Files
UETP:	User Environment Test Package

The list of files included in each of these classes is documented in the following OpenVMS file: SYS$UPDATE:VMSKITBLD.DAT.

You must assess and test which classes are required by your software. In a System Support Addendum (SSA) entry, you must state which OpenVMS classes your product requires. Without this information, customers would have no idea which OpenVMS files could be excluded on their systems with respect to Digital layered software.

OpenVMS ships a program, SYS$UPDATE:VMSTAILOR.EXE, that allows you to remove or add parts of OpenVMS by function rather than by file. Typically, a system manager would use this image to "tailor off" OpenVMS files. ("Tailoring off" simply means deleting files in groups.) Should a site wish to "tailor on" (add OpenVMS files in groups) any class, the original OpenVMS kit (on magnetic tape, CD-ROM, and so on) must be mounted and the device specification provided to VMSTAILOR.EXE. The Example section of this requirement description shows a sample run of this program.

Note that from a customer's point of view, each of these classes can be further divided into mutually exclusive "subclasses"; however, from the point of view of the layered product, this level of granularity is unnecessary for its statement and assessment of support.

A Typical Test Run

The following steps illustrate a typical test run:

1. Build or access a full OpenVMS system. OpenVMS is, by default, shipped and built with all OpenVMS files included.
2. Ideally, include a backup of the system on another disk for quick rebuilds. Otherwise, include a backup of the system on tape.
3. Using the SYS$UPDATE:VMSTAILOR.EXE program, tailor off those classes not needed by the product. The example in the Example section includes a sample run of the tailoring off process.
4. Install the software, including any prerequisites. Execute the IVP. If this phase fails, make sure that the errors are caused by missing files. Assess which files are missing and to which classes they map. Rebuild the entire system (which takes only a few minutes from a disk backup), making sure to include any new class(es) on which testing shows dependence. Repeat steps 3 and 4 until this phase is completed successfully. Note that you can add classes by using the TAILOR ON feature of VMSTAILOR.EXE.
5. Select a regression test suite that exercises OpenVMS components or OpenVMS file access. Execute this suite, again noting those tests that fail because of missing files. If this phase fails, repeat the rebuild procedure outlined in step 4, being sure to include any new class(es) identified in this phase of testing.

Rationale

To ensure that the product functions correctly on the minimum OpenVMS tailoring classes specified as prerequisite.

Exceptions

None

Example

The following example contains a sample run of VMSTAILOR and shows how to tailor a system for which the stated support is the following:

OpenVMS Required Saveset
Programming Support
Secure User Environment

```
$ RUN SYS$UPDATE:VMSTAILOR

 This is the OpenVMS tailoring utility. It lets you customize OpenVMS for your
 particular system. You do this by identifying classes and subclasses of
 OpenVMS files that you want to add or delete.

 VMSTAILOR has three phases:
```

```
 - First, VMSTAILOR asks if you want to tailor your system by adding files
 ("tailor on") or by deleting files ("tailor off").

 - Second, if you choose the "tailor on" procedure, VMSTAILOR lists each
 class of OpenVMS files and asks if you want to include them on the system disk.
 If you choose the "tailor off" procedure, VMSTAILOR lists each class of OpenVMS
 files and asks if you want to delete them from the system disk.

 - Third, VMSTAILOR carries out your instructions.

 NOTE: To cancel VMSTAILOR during the first and second phases, press
 Ctrl/C, Ctrl/Y, or Ctrl/Z. If you cancel VMSTAILOR during the
 third phase, you may end up with a partially tailored disk.

 Do you want to tailor files "ON" or "OFF"? OFF

TAILOR-OFF
----------
 You will now be prompted with a list of the classes and subclasses of OpenVMS
 files that are optional. The size of each class and subclass is included
 in the list. This will help you decide whether or not you want to remove a
 class or subclass from your system.

 Under some classes there is a set of common files that is required in order
 for any subclasses to work. These files will only be deleted if you remove
 the ENTIRE class. If you choose to keep any subclass, this set of files is
 not deleted.

 NOTE: The files associated with selected classes and subclasses will
       be removed from your system.

Total size of the system disk is 311200 blocks.
Total space used on the system disk is 154654 blocks.
Total space left on the system disk is 156546 blocks.

CLASS - Network Support
          Size of entire class (with subclasses): 1445
          Size of common files required for any subclass: 1223
          Do you wish to select the entire class (default = NO)? YES

CLASS - Programming Support
          Size of entire class (with subclasses): 9009
          Size of common files required for any subclass: 0
          Do you wish to select the entire class (default = NO)?
          Do you wish to select any of its subclasses (default = NO)?

CLASS - RMS Journaling Files
          Size of entire class (with subclasses): 72
          No subclasses in this tailor class.
          Do you wish to select the entire class (default = NO)? YES

CLASS - System Programming Support
          Size of entire class (with subclasses): 1269
          Size of common files required for any subclass: 87
          Do you wish to select the entire class (default = NO)? YES

CLASS - Secure User's Environment
          Size of entire class (with subclasses): 613
          Size of common files required for any subclass: 0
          Do you wish to select the entire class (default = NO)?
          Do you wish to select any of its subclasses (default = NO)?

CLASS - Utilities
          Size of entire class (with subclasses): 8639
          Size of common files required for any subclass: 0
          Do you wish to select the entire class (default = NO)? YES
```

```
CLASS - OpenVMS Workstation Support
          Size of entire class (with subclasses): 4
          Size of common files required for any subclass: 0
          Do you wish to select the entire class (default = NO)? YES

CLASS - BLISS Required Files
          Size of entire class (with subclasses): 3945
          No subclasses in this tailor class.
          Do you wish to select the entire class (default = NO)? YES

CLASS - Miscellaneous Files
          Size of entire class (with subclasses): 1555
          Size of common files required for any subclass: 0
          Do you wish to select the entire class (default = NO)? YES

CLASS - Use Environment Test Package
          Size of entire class (with subclasses): 551
          No subclasses in this tailor class.
          Do you wish to select the entire class (default = NO)? YES

CLASS - Example files
          Size of entire class (with subclasses): 1142
          No subclasses in this tailor class.
          Do you wish to select the entire class (default = NO)? YES

Files have been selected
Do you wish to remove all of the options selected? Y
Removing files, please wait...
         .
         .               {A list of deleted files follow}

$ LO
```

For more information

For information on using DTM on a tailored system to run regression tests, refer to Note #697 in the CLT::DTM VAX Notes conference.

vms_tailor_doc

The preinstallation section of the installation guide lists the required OpenVMS tailoring classes, if there are any.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS

PERFORMED BY: Technical Writer
REQUIREMENTS AID(S): None

Description

The preinstallation section of the installation guide must include a list of the minimum OpenVMS classes necessary for full functionality of the layered product. If the installation guide is not updated with each release of the product, a pointer to this information in the System Support Addendum (SSA) should be included rather than specific class information. If a list is included, full class names must be used instead of mnemonics. All layered products must by default support "OpenVMS Required Saveset" and include this class in this section.

Also, any OpenVMS class information included in the installation guide must be consistent with the OpenVMS class information that appears in the SSA.

Rationale

To ensure all requirements are met for a successful product installation.

Exceptions

None

Example

Required Operating System Components

To install DECforms, you must have certain operating system components. The components you need depend on whether you are installing the DECforms full development kit or the run time kit.

The DECforms full development kit contains the DECforms run time system and development utilities. To install this kit, you must have installed the following OpenVMS classes:

- OpenVMS Required Save Set
- Programming Support (for STARTLET.OLB and IMAGELIB.OLB)

- Utilities (for HELPLIB.OLB)

The DECforms run time kit contains only the DECforms run time system. To install the DECforms run time kit, you must have installed the OpenVMS Required Save Set class.

For more information

Refer to SYS$UPDATE:VMSKITBLD.DAT for the latest list of OpenVMS tailoring classes.

window_manip

The product handles user requests for window manipulation in a mixed platform environment.

OPERATING SYSTEMS (ALL ARCHITECTURES):

	OpenVMS OSF/1 Sun ULTRIX
PERFORMED BY:	Software Engineer
REQUIREMENTS AID(S):	None

Description

The DECwindows Motif application must demonstrate it can handle user requests to move, resize, shrink windows to icons, and expand icons to windows in a mixed platform environment.

All client applications must ensure proper sizing and placement of their initial display, regardless of the size or aspect ratio of the targeted server (for example, a Macintosh server).

Areas of the initial display that do not appear on the screen are acceptable, provided the off-screen area does *not* contain:

- Any components of the title bar, such as icon and product name
- Function buttons
- Scroll bars

Rationale

A user may not have access to full functionality of a product on a small screen unless the product has compensated for the size of the screen. Additionally, all products using DECwindows Motif servers must fully interoperate with all servers supported by Network Application Services (NAS).

Exceptions

None

Example

Not applicable

For more information None

xtapplinitialize

XtApplInitialize is normally called first before any other function in the Xt Toolkit. It initializes various Xt data structures, opens the connection to the X server, and returns a shell widget.

OPERATING SYSTEMS (ALL ARCHITECTURES):

OpenVMS	OSF/1	Sun
ULTRIX	SCO UNIX	AIX
HP-UX	Windows NT	Windows
Mac	MS–DOS	

PERFORMED BY: Software Engineer or Project Leader

REQUIREMENTS AID(S): None

Description

The XtApplInitialize function is the interface that should be used by applications when the X-server must be initialized.

Rationale

The use of the toolkit allows a greater standardization of the developers code with less possiblity of errors introduced through low level programming.

Exceptions

Non-windowing applications or applications not calling the Xt toolkit directly.

Example

Not applicable

For more information

Refer to *ULTRIX Worksystem Software (UWS)*, Volume 3, *Programming*)

For more information on getopts, refer to the man page for getopts on a SunOS system.

Randi J. Rost's book "X and Motif Quick Reference Guide"

X Window System Toolkit" by Asente & Swick

A

Requirement Aids

This appendix provides summary information on the following Requirement aids referenced in this guide:

- Section A.1, REQ_CHECK
- Section A.2, APRS

A.1 REQ_CHECK

INTRODUCTION

REQ_CHECK is a quality assessment tool developed by Software Quality Operations and Information Services (SQOIS). The REQ_CHECK software tests specific quality attributes of layered products. It helps to determine problems early in the development cycle and to reduce costly problems before product release.

The REQ_CHECK tools currently works on the following platforms:

- OpenVMS/VAX
- ULTRIX/VAX
- ULTRIX/RISC
- OSF/1-AXP
- OSF/1(RISC)

- ULTRIX and OSF products installed using setld format. This version also can install a product and the evaluate it.
- OpenVMS VAX product need to be installed using VMSINSTAL option 'S' first.

Technical requirements verified by the REQ_CHECK tool.

If you are using EL-CP790-00, Technical Requirement Guide (TRG) - Reference, the requirements verified by the REQ_CHECK tools are identified by the word REQ_CHECK in the "Requirement Aids" section.

If you are using any of the following checklists:

EL-CP790-01, Technical Requirement Guide (TRG) - PC Checklist

EL-CP790-02, Technical Requirement Guide (TRG) - OpenVMS Checklist

EL-CP790-03, Technical Requirement Guide (TRG) - UNIX Checklist

the requirements verified by the REQ_CHECK tool are identified by the REQ_ CHECK after the one or two sentence requirement statement. The Requirements Overview at the beginning of this guide identifies the requirements that the OpenVMS VAX and UNIX-based REQ_CHECK tools verify when you run the software. Look for requirements that are marked with a "R".

For the latest information on the type of checks performed, check the VAX Notes Conferences listed in the For More Information section.

After running REQ_CHECK, the software generates a report explaining the tests performed and the status of the product. The report has two sections:

Section_1:	The first section contains information on the quality features examined and the product's compliance (or non-compliance) with the requirements or guidelines.
Section_2:	The second section contains logs of the disk space usage, installation text of the product under test, and the system configuration prior to installation. This information is included for the recording purposes.

Documentation

There is no formal documentation on REQ_CHECK. At present, the tool is intended for the software engineers who are familiar with setld and VMSINSTAL; therefore, a basic level of knowledge is assumed.

The next section provides details on how to obtain the tool and where information and comments can be posted.

A.1.1 HOW TO RECEIVE THE LATEST REQ_CHECK TOOL

Requests for the tool should be made to VIRTUE::INFORMATION by stating requester's name, node, the group they represent, and the tool requested. Instructions on how to copy and set up the software will be sent to you.

- OpenVMS VAX needs the following prerequisites to operate:
  - VMS V5.3 or later
  - Run time DECforms 1.3 or later.
- ULTRIX/OSF is written in korn shell. Because of this, it will not need any changes for different machine architectures. It operates on VAX, MIPS, and Alpha AXP without any changes. The is one small program that requires a "C" compiler.

A.1.2 FOR MORE INFORMATION

OpenVMS VAX

Refer to note #418 in the VIRTUE::SQM-TOOLS VAX Notes Conference for detailed instructions on the set up and operation of the REQ_CHECK software. This entry also serves as a medium to communicate your comments on the tool, your suggested enhancements, and any problems or bugs.

ULTRIX/OSF

Refer to note #174 in the VIRTUE::SETLD-ULTRIX-LPS VAX Notes conference for detailed instructions on set up and operation of the REQ_CHECK software. This entry also serves as a medium to communicate your comments on the tool, your suggested enhancements, and any problems or bugs.

A.2 APRS

INTRODUCTION

The Automatic Product Registration System (APRS) is a software application with the following characteristics:

- It is a server-client application.
- The client runs on the OpenVMS operating system.
- It comes with a character cell interface.
- It is used to register unique product attributes of Digital Products in a registration data base.

Where APRS Helps You

The following benefits are only a few of the advantages the APRS software brings to the software development community:

- Provides a central location in which all registration information on products is maintained.
- Gives direct access to the data base.
- Supports proactive queries in planning for new attributes. For example, Is the FOO BAR mnemonic available, or is it registered to another product.
- Avoids delays in the registration of product and software attributes, improving the effectiveness of the registration process.
- Simplifies the registration process by providing user-friendly interfaces with the ability to register registration data from a text file, or copy them from another product.
- Helps in determining product trends.

Location of the APRS Software and Documentation

The APRS software files and user's guide are located in the following directory:

OOBME::PUBLIC$APRS:

The following identifies the APRS software files you need for your working environment:

aprs.com
aprs.exe
aprs_users_guide.ps

Refer to the APRS User's Guide for details on installing and running the APRS software to register product and software attributes.

B

List of Abbreviations and Acronyms

ACA: Application Control Architecture

APRS: Automatic Product Registration Tool

bind: Berkeley Internet Name Domain

BIV: Basic International Version

BOM: Bill of Material

ConDist: Consolidated Distribution

CPAK: Component Product Authorization Key

CPU: Central Processing Unit

CSERL: Central Software Engineering Release Library

CSSE: Customer Service System Engineering

DCE: Distributed Computing Environment

DCL: Digital Command Language

DDSLA: Digital Distributed Software License Architecture

DMS: Diskless Management Services

DTM: DEC/Test Manager

EIID: Electronic Information Integration & Delivery

EMA: Enterprise Management Architecture

EPS: Electronic Product Submission

FRS: First Revenue Ship

GIA: General International Area

I14Y: Interoperability (This abbreviation is derived using the first and last letters of the word and the number of letters between them.)

I18N: Internationalization (This abbreviation is derived using the first and last letters of the word and the number of letters between them.)

ISE: This abbreviation has two possible definitions: International Systems Engineering or Integrated Systems Evaluation.

IVP: Installation Verification Procedure

ksh: Korn shell

LDB: License Data base

LMF: License Management Facility

MBU: Manufacturing Business Unit

NAS: Network Application Services

NIS: Network Information Service

NLS: Native Language Service

ORDS: Online Registration and Distribution System

PAK: Product Authorization Key

PRM: Product Readiness Memo

PTT: Public Telephone Telegraph

QAR: Quality Assurance Report

RIS: Remote Installation Services

SDT: Software Development Tools

sh: Bourne shell

sh5: System V Bourne shell

SIM: Systems Integration Management. This group is now called Software Quality Assurance (SQA). Any references within this manual to SIM should be interpreted as referring to the group now called SQA.

SMPG: Software Manufacturing Products Group

SNPC: Software New Products Committee

SPD: Software Product Description

SPR: Software Performance Report

SQA: Software Quality Assurance (This group was formerly known as SIM.)

SQM: Systems Quality Management (includes SQM-UK, SQM-US, and SQM-Japan). This group is now known as Software Quality Operations and Information Services (SQOIS). Any references within this manual to SQM should be interpreted as referring to the group now called SQOIS.

SQOIS: Software Quality Operations and Information Services

SQTF: Systems Quality Test Facility

SSA: System Support Addendum

SSB: Software Supply Business

TCP/IP: Transmission Control Protocol/Internet Protocol

TRG: Digital Technical Requirements Guide

UCX: VMS/ULTRIX Connection

UID: User Interface Definition

UIL: User Interface Language

URI: Unique Requirement Identifier

yp: Yellow Pages

C

REFERENCED DOCUMENTS

C.1 Digital Documents, EL-Class

Document Number	Document Title
EL-00012-04	*DEC STD 012-4 Unified Numbering Code (UNC) - Software Numbering Conventions*
EL-00038-01	*DEC STD 038-1 Systems Evaluation of New Products - Software*
EL-00041-00	*DEC STD 041-0 Customer Installability: Product Requirements*
EL-00066-03	*DEC STD 066-3 Policy for Designing Products for All Countries Designated as Strategic Markets*
EL-00095-00	*DEC STD 095-0 VAX/VMS Internal Application Release Criteria*
EL-00197-00	*DEC STD 197-0 Legal Requirements and Guidelines for Digital Publications and Software*
EL-00204-00	*DEC STD 204-0 Software Correction and Distribution*
EL-SM498-00	*Producing International Products Reference Set*
EL-CP790-01	*Technical Requirements Guide (TRG) - PC Checklist*
EL-CP790-02	*Technical Requirements Guide (TRG) - OpenVMS Checklist*
EL-CP790-03	*Technical Requirements Guide (TRG) - UNIX Checklist*

C.2 Digital Document, Other Than EL-Class

Document Number	Document Title
AA-PHGWA-TK	*DECwrite User's Guide*
EY-F577E-DP	*Digital Guide to Developing International Products*
AA-PJUPA-TE	*Digital OSF/1 Guide to Programming Supplement Tools*
AA-PQEMA-TE	*PATHWORKS for SCO UNIX API Programmer's Reference*
AA-PE81A-TK	*NAS Guide to Developing Portable Software*
AA-LA00B-TE	*OpenVMS System Manager's Guide*
AA-PBK5A-TE	*OpenVMS DCL Dictionary - Part I*
AA-PBK6A-TE	*OpenVMS DCL Dictionary - Part II*
AA-LA35A-TE	*OpenVMS Backup Utility Manual*
AA-LA42A-TE	*OpenVMS Authorize Utility Manual*
AA-LA58A-TE	*Guide to Creating OpenVMS Modular Procedures*
AA-LA17B-TE	*OpenVMS Systems Messages and Recover Procedures Manual, Part I*
AA-LA18B-TE	*OpenVMS Systems Messages and Recover Procedures Manual, Part II* *OpenVMS Message Utility Manual* *OpenVMS DECwindows Motif Guide to Application Programming*
AA-ME93B-TE	*Guide to Disk Maintenance*
AA-ME92B-TE	*Guide to Backup and Restore - ULTRIX*
AA-MG62B-TE	*ULTRIX Software Development, Volume 2, Guide to Preparing Software for Distribution on ULTRIX Systems*
AA-LY21B-TE	*ULTRIX System Management, Volume 3, Guide to Bind Service*
AA-ME00B-TE	*ULTRIX System Management, Volume 3, Guide to the Yellow Pages Service*
AA-LY26B-TE	*ULTRIX Software Development Guide to Developing International Software*
AA-MG62B-TE	*Preparing Software for Distribution on ULTRIX Systems*
	ULTRIX System Management, Volume 3, Guide to Diskless Management Services
	ULTRIX Reference Pages
AA-PC7NB-TE	*OSF/Motif Programmer's Guide*
AA-OC7PB-TE	*OSF/Motif Style Guide* *Digital Guide to Software Development* *X Window System* *Digital Guide to Producing International Products* *Guide to Programming Support Tools* *Open Desktop Product Engineering Toolkit Guide* *Guide to Programming Support Tools (For OSF/1)* *ULTRIX Worksystem Software (UWS), Volume 3, Programming*

The following documents can be copied from the pointers given.

Security Standards for ULTRIX and OpenVMS Layered Products
{VIRTUE, SQMUK, SQMJPN}::SYS$INFO:SECURITY_GUIDE.*

ULTRIX Installation Guide
VIRTURE::SYS$INFO:INSTAL_GD_ULTRIX.SDML

Software Licensing and License Management Facility Guide
VIRTUE::SYS$INFO:LMF_GUIDE.*

SPD/SSA Production and Review Process Workbook for OpenVMS and ULTRIX Layered Products
VIRTUE::SYS$INFO:SPD_SSA_WORKBOOK.*

Guide to Portable Software
PIPE::AIA$INFO:GD_TO_PORT_SW.PS

Developer's Guide to VMSINSTAL
VIRTUE::SYS$INFO:DEC_VMINSTAL_AUG90.*

APRS User's Guide
AURORA::WORK1$:[CUI_TOOLS.PUBLIC]

OpenVMS Installation Guide
VIRTUE::SYS$INFO:INSTAL_GD_VMS.SDML

Product Variant Evaluation Strategy
Send E-Mail request to SQMUK::SQM

X-Open Portability Guide
See VAX Notes Conference VIRTUE::ULTRIXINFO (Note 54)

Articles are available from the authors listed.

"Performance Engineering Throughout the Product Life Cycle,"
Eyal Zimran and David Butchart, DEC-TR 868.

"Intergrating Performance Engineering into the SQA Process,"
Eyal Zimran, January 1993, Presentation.

"Guidelines for Development of Performance Requirements, Goals and Test Plans,"
Eyal Zimran and Jim Behyer, DEC-TR 869.

C.3 Documents External to Digital

Document Number	Document Title
ISO 639	*Codes for the Representation of Names and Languages*
ISO 3166	*Codes for the Representation of Names of Countries*
	X and Motif Quick Reference Guide, Randi J. Rost
	X Window System Toolkit, Asente & Swick

C.4 Ordering Information

Use VTX SMC to order copies of EL-class Digital documents from Standards and Methods Control. Send distribution questions to JOKUR::SMC or call DTN: 234-4423.

Following are the sources for documents not available from Standards and Methods Control. Copies may also be available from your local Digital library.

- EY-class part numbers can be ordered from:
  Publishing and Circulation Services
  DTN: 234-4325, NEST::ORDER
- AA-class part numbers can be ordered from:
  Software Supply Business (SSB) Order Administration
  DTN 241-3023, WMOIS::SDCISO
- Use VTX SECURITY_POLICY for Security Policies and Standards.
- ISO Standards can be ordered from:
  American National Standards Institute (ANSI)
  11 West 42nd Street
  New York, NY 10036
  Telephone (212) 642-4900, or
  GVPROD::STANDARDS
- For documents with no part numbers provided or with specifc distribution questions on any of the above referenced documentation, please contact SQA.

Index

G

I

K

L

M

N

O

P

R

S

T

U

V

W

X

```
+---------------------------------------------------------------+
|                        READER COMMENTS                        |
|                                                               |
| Your  comments  and suggestions  will help  Standards and Methods |
| Control improve their services and documents.                 |
+---------------------------------------------------------------+
```

Did you request this document? _____ If so, did it arrive within a satisfactory period of time? _____ Please comment.

What are your impressions of this document? Consider format, organization, completeness, readability, and illustrations.

——————————FOLD ON THIS LINE—————————————————————— -

Did you find technical or clerical errors in this document? If so, please specify the page number(s) and the error(s).

Are the instructions for the update package clear? _____
Was an index available? _____ If not, is one needed? _____
Do you have other suggestions for improving this document?

The following information is optional:

Name _____ Mailstop _____

Department _____ Node _____

Send your comments to JOKUR::PROJECTS, or fold, staple, and send this page through interoffice mail to:

```
+------------------------------+
|       READERS' COMMENTS      |
| STANDARDS AND METHODS CONTROL |
|            NRO5/I2           |
+------------------------------+
```