OSF® DCE Version 1.2.2

Release Notes

January 22, 1997

DOC-RN-00028

Open Software Foundation
11 Cambridge Center
Cambridge, MA 02142

The information contained within this document is subject to change without notice.

OSF MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

OSF shall not be liable for errors contained herein, or for any direct or indirect, incidental, special or consequential damages in connection with the furnishing, performance, or use of this material.

Legal Notice
For
OSF® DCE 1.2.2 Release Notes

1. **The internationalized version of DCE 1.2.2 does not include the following directories (See Section 1 of the DCE Test License Terms & Conditions):**

   - **src/security/krb5/lib/crypto/domestic**
   - **src/rpc/kdes**

   Note: The following three (3) files contained in the ‘‘src/rpc/kdes’’ directory are included in the internationalized version of DCE 1.2.2: ‘‘Makefile,’’ ‘‘des_neuter.c,’’ and ‘‘des.h.’’

2. **Licensing and Packaging**

   For information on licensing and packaging, see the DCE 1.2.2 Licensing and Packaging Agreement.

3. **Treatment of Informix C-ISAM code:**

   For those portions of the DCE source code based on Informix C-ISAM source code, Licensees can use such source code only for the purpose of porting the ‘‘Global Directory Service (‘GDS’)’’ application. Such portions are all of the files identified in the following directory:

   - **src/directory/gds/cisam**

4. Free Software Foundation (‘‘FSF’’) source code appears in the below referenced directories. Licensees must follow the terms of the Free Software Foundation Copyright Notice and the GNU General Public License preceding the FSF source code.

   - **directory/gds/nds/gnu_regex.c**
   - **directory/gds/nds/gnu_regex.h**

Contents

LIST OF FIGURES

LIST OF TABLES

Preface

The *OSF® DCE Version 1.2.2 Release Notes* contain information on OSF DCE Version 1.2.2.

Audience

The *OSF DCE Release Notes* are written for developers, system administrators, and documentation writers.

Applicability

This is Revision 1.0 of this document. It applies to OSF DCE Version 1.2.2.

Purpose

The *OSF DCE Release Notes* are written for developers, system administrators, and documentation writers at licensees' sites and contains instructions for loading and building documentation and source directories from the distribution tape. The *OSF DCE Release Notes* also summarize notable changes to the code and to the documentation that have occurred since the previous version of DCE.

Document Usage

The *DCE Release Notes* are organized into 7 chapters:

- Chapter 1. Overview of OSF DCE Version 1.2.2

  This chapter gives general DCE Version 1.2.2 release information, including the enhancements made since DCE Version 1.2.1.

- Chapter 2. State of the DCE Components in DCE 1.2.2

  This chapter describes the state of the DCE 1.2.2 enhancements, organized by component.

- Chapter 3. OSF DCE 1.2.2 Directory Structure

  This chapter contains graphical depictions of the DCE 1.2.2 archives.

- Chapter 4. Building and Installing DCE 1.2.2

  This chapter gives information on loading, building, and installing DCE 1.2.2.

- Chapter 5. Documentation Supplied in DCE 1.2.2

  This chapter describes the documentation in DCE 1.2.2.

- Chapter 6. Building the DCE 1.2.2 Documentation

  This chapter contains information on building and printing the DCE 1.2.2 documentation.

- Chapter 7. Distributed File-Access (DFA) Release Notes

  This chapter contains information describing how to build and install the DCE Distributed File-Access manager (DFA).

Typographic and Keying Conventions

This document uses the following typographic conventions:

Bold	**Bold** words or characters represent system elements that you must use literally, such as commands, flags, and pathnames.
Italic	*Italic* words or characters represent variable values that you must supply.
`Constant width`	Examples and information that the system displays appear in `constant width` typeface.
[ ]	Brackets enclose optional items in format and syntax descriptions.
{ }	Braces enclose a list from which you must choose an item in format and syntax descriptions.

|	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on the keyboard.
...	Horizontal ellipsis points indicate that you can repeat the preceding item one or more times. Vertical ellipsis points indicate that you can repeat the preceding item one or more times.

This document uses the following keying conventions:

<Ctrl-*x***>** or ^*x*	The notation **<Ctrl-***x***>** or ^*x* followed by the name of a key, indicates a control character sequence. For example, **<Ctrl-c>** means that you hold down the control key while pressing **<c>**.
<Return>	The notation **<Return>** refers to the key on your terminal or workstation that is labeled with the word Return or Enter, or with a left arrow.
Entering commands	When instructed to *enter* a command, type the command name and then press **<Return>**. For example, the instruction "Enter the **ls** command" means that you type the **ls** command and then press **<Return>** (enter = type command + press **<Return>**).

Pathnames of Directories and Files in DCE Documentation

For a list of the pathnames for directories and files referred to in this guide, see the *OSF DCE Administration Guide* and the *OSF DCE Testing Guide*.

Chapter 1. Overview of OSF® DCE Version 1.2.2

This chapter consists of an overview of the software supplied in OSF DCE Version 1.2.2 of the Open Software Foundation's (OSF) Distributed Computing Environment (DCE).

1.1 Distributed Computing Environment

These notes accompany Version 1.2.2 of DCE. DCE 1.2.2 is a software offering that includes complete source code, documentation, and tests to allow licensees to build and run DCE programs and to develop products based on DCE technology. Your rights to distribute DCE 1.2.2 may vary depending on the license options you have selected; you should consult your license agreement for details.

DCE 1.2.2 is an OEM software release that has been integrated and tested sufficiently to be used as a basis for further development efforts, such as ports to new platforms, distribution to customers who can make use of the new technology, and enhancement activities for specific licensee business opportunities. DCE licensees should expect to learn enough about the technology to meet their business needs, including providing end-user support—DCE is not a ''turnkey'' technology offering. Significant DCE internals documentation, including a testing guide and internals specifications, is included on the DCE release distribution tapes. OSF also offers courses on various aspects of DCE.

Significant items in DCE 1.2.2 added since DCE 1.2.1 include the following:

- Kerberos V5 support

  The DCE security service includes an implementation of the MIT Kerberos Version 5 (V5) authentication and key distribution service. Prior to DCE 1.2.2 there has been no formal OSF DCE interoperability commitment.

  The Kerberos V5 protocol has become more stable with the release of IETF-RFC 1510 and its movement through the IETF standards process. DCE 1.2.2 enhances the high degree of interoperability that existed in previous releases with the committed support for the IETF-RFC 1510 protocol. The protocol formally allows Kerberos V5 applications running on either DCE or non-DCE platforms to access the DCE

security server as a full-function IETF-RFC 1510 Kerberos server. The DCE security server's implementation has been tested against MIT Kerberos Version 5 releases beta 4 and beta 5.

Many consumers of DCE wish to extend the single login environment. The MIT Kerberos releases include network utilities that transmit user account information. These utilities are integrated with Kerberos to achieve a single login facility in the networked environment. DCE 1.2.2 includes implementations of `rlogin` and `rsh` that use DCE's Kerberos facilities to avoid exposing passwords on a network.

- Public Key support

  DCE 1.2.2 allows public key technology to be used to support login. With this technology, the security server does not need to store the long term key (or password) for a principal so that it will remain undisclosed should any compromise of the security server occur. Administrators can specify that some principals may use the pre-DCE 1.2 mechanisms while others have access to the public key mechanism; it will retain full interoperability with existing DCE releases.

  At login, public key users receive credentials that allow them to use the current (Kerberos-based) DCE authentication mechanism. A new pre-authentication protocol is used. The login client does not have to determine whether a given user is public key-capable prior to requesting credentials.

  As a transition aid, this revision of DCE includes a new component called the Private Key Storage Server (PKSS). It is intended for customers who want to use public key encryption for secure logins (per OSF RFC 68.2), but who may not want to invest in the equipment and infrastructure required to deploy hardware-based public key security enterprise-wide, or who may want to phase in such mechanisms over time. A PKSS server **pkssd** and related client API are supplied. The API includes functions for user clients and management clients. The functionality of the PKSS server and its user and management APIs are described completely in OSF RFC 94.1. PKSS is also integrated into the public key login facility.

  A new certification API is also being provided. This facility handles the mapping of a principal name to a public key, allowing programmers to hide the details of their own Certificate Authority (CA) access methods and trust model. By letting developers ''plug in'' their own policy and storage modules, this facility continues the DCE practice of providing widespread foundation without dictating a single use model. (Note that public key functionality in DCE 1.2.2 depends on the presence of the RSA ''BSAFE'' encryption code. For further information, see ''Contents of the DCE 1.2.2 Release Area'' below.)

- User-to-user authentication

  When the concept of ''server'' is associated with a long running system resource — such as the name server or the security server — it seems natural that the server run on a machine with access to the long term key to the identity of that server. (If for no other reason than that the server is not normally associated with a human user but rather with a pseudo-user corresponding to the system service.) This does not, however, map well onto all application domains. In particular, some applications need a ''peer-to-peer'' model.

  The user-to-user authentication facility provides an alternate Ticket Granting Service (TGS) protocol as defined in IETF-RFC 1510. In particular it is now possible to

direct a protected RPC to a program that only has a login context, and no key table (file) or other access to a long-term key.

- Global groups

  DCE 1.2.2 allows principals from a foreign cell to be added to groups in the local cell. Such tasks as enterprise-wide security administration, cell reconfiguration, and other management tasks are also made much easier. For example, cross-cell cooperation among DCE time services is now possible.

- Single-threaded RPC

  With DCE 1.2.2, datagram RPC clients no longer have a dependency on a threaded environment. This simplifies the task of application developers, eliminating the need for thread-aware programmers, debuggers and support libraries.

- Scalability improvements to security

  Changes to the security server have been made resulting in considerable performance improvements with large cells (those with more than 50,000 principals). The improvements include documenting the configurable checkpoint interval and partitioning internal datasets so that the amount of data written to disk during a checkpoint is proportional to the amount of data modified. In addition, client binding to security servers has been made more efficient.

- DFS server multi-home support

  In DCE 1.2.2 the DFS servers have been enhanced to perform better on hosts connected through multiple interfaces to multiple networks. Prior to DCE 1.2.2, DFS required all clients and servers to be reachable via all network interfaces.

- DFS backup performance enhancements

  A number of performance bottlenecks have been eliminated in the behavior of DFS backups, dumps, and restores:

  — A ‘‘bulk update’’ operation has been added, allowing the master sync site to send sets of updates to each secondary server in a single RPC.

  — The master sync site now updates secondary servers asynchronously, and works on the subsequent batch of changes in parallel with the RPC calls being performed.

  — Database transactions are no longer committed asynchronously, so that all servers now do commit processing in parallel.

  — A new RPC operation has been added so that multiple volumes can be added to the backup database in a single RPC call.

- DFS support for 64-bit filesystems

  DCE 1.2.2 supports very large files and filesystems, while maintaining interoperability with the current widespread 32-bit machines.

- DFS use of protected RPC

  DCE 1.2.2 allows administrators to specify a range of DCE protections that can be used for most client-server communication. All architectural uses of unauthenticated RPCs have been eliminated.

Prior to DCE 1.2.2, DFS used the full range of DCE protection levels on its RPC operations, and the new administrative controls allow administrators to distinguish between same-cell communication from inter-cell communication, so that a DFS Cache Manager will use one set of protection rules for intra-cell use (presumably protected behind a network firewall), while using another set for data-sharing outside the cell. Command line arguments and management clients allow administrators to achieve the right balance of protection and computational overhead.

- Documentation in SGML

  SGML is an industry standard for representing documentation that is intended to be viewed in a variety of formats, encompassing printed matter and on-line ‘‘hypertext’’ viewing. The DCE 1.2.2 documentation is distributed as SGML source, using the DocBook Document Type Definition (DTD).

1.2 Using the DCE Technology

This section contains important information you need to know when working with DCE. It supplements the information found in the license agreement, which we assume you understand. It also describes how to tailor DCE to your environment while keeping synchronized with the DCE work going on at OSF.

The DCE offering includes source in electronic form for the following:

- A reference implementation of DCE, coded for the DCE 1.2.2 reference platform
- Supporting documentation: tutorial, procedural, internals
- Demonstration and example programs
- Functional and system tests to aid in completing product development
- Information about the current state of the technology

Your rights to redistribute executables, object code, and documentation depend on the DCE licensing option you have selected. See your license for details.

DCE is a large, complex, innovative distributed system. The technology developers have worked hard to provide a complete, high-quality implementation that can be used by skilled systems software developers to produce useful applications. Please keep in mind that DCE deployment for end users will require licensee investment in porting, tuning, defect repair, and support. Information about the current state of functionality, testing, and open defects can be found in this document, in the documentation directories, and in the **project** archive under **defect.summary**, **test.results**, and **quality.rpt**.

OSF makes a variety of other materials and services available, apart from the DCE distribution. These include educational materials such as brochures, training classes, and seminars; software support services, including problem resolution and update; development access and consultation at our Cambridge site; and, interoperability testing and validation tools and services. Contact OSF Direct at (617) 621-7300 for more information.

Other organizations make other DCE-related materials and services available, including executable ports to other platforms, training courses, published books, and DCE extensions and applications. OSF can provide contact information for these materials and services as they are made available to us.

1.2.1 Porting and Adding Value to DCE

When adding or removing capabilities, or changing the implementation or specification, licensees should keep in mind the following:

- Porting changes should be made using appropriate techniques, such as conditional compilation and separation into platform-dependent modules. This will aid in determining where problems occur and will facilitate merging updates (see the following section). Although changes to interfaces and protocols by licensees are strongly discouraged, there are cases that require small changes in the interfaces between DCE and the underlying system. Refer to the DCE 1.1 version of the *OSF DCE Porting and Testing Guide* for more details about the areas that may need this type of special attention.

  Note: The optional Private Key Storage Service (PKSS) component has a porting issue that is described in OT CR 13688 and in Section 2.3.10 (''State of Security Code'').

- Changes, additions, or modifications to the DCE source should be made only after careful consideration of their potential effects on application portability, system interoperability, and the management of multivendor installations. In particular, we strongly advise against making changes to the wire protocols as specified by the interface definition (IDL) files, or to the application-visible programming interfaces. A set of unpublished specifications for DCE, currently being submitted for X/Open Common Applications Environment (CAE) standardization, includes the vendor-neutral specification of the interfaces and protocols that should be present in DCE systems.

  Licensees are urged to communicate to OSF any information they may develop about changing the DCE source to improve performance or fix bugs. See ''Sending Feedback to OSF on DCE 1.2.2'' at the end of this chapter.

- Interoperability issues are particularly important, because end users will expect DCE implementations from various suppliers to interoperate. OSF will sponsor interoperability testing opportunities as a means of verifying ports before deployment.

- On the other hand, licensees should feel comfortable making changes to parts of the DCE source such as platform-dependent Makefiles, start-up scripts, administrative documentation, and other system-dependent areas in which changes will not adversely affect DCE portability or interoperability. Internal changes to fix bugs or improve performance are appropriate and desirable, though these changes should be identified clearly in the source files in order to simplify merging future updates. You should always make sure that internal changes do not affect the external semantics of the functions and protocols.

- The DCE license suggests one way to package and deploy the DCE technology. In addition, licensees are allowed to extract parts of the DCE technology for use in embedded systems that need not interoperate with other DCE systems. OSF will not provide any special support for such applications, and license fees still apply.
- Licensees and their customers are encouraged to develop additional DCE applications such as development tools and management applications. In general, deployment of such applications to systems that already have DCE will not incur additional license fees (unless the application is directly derived from DCE sources, rather than simply linking with DCE functions).

1.3 Contents of the DCE 1.2.2 Release Area

DCE 1.2.2 will be released both electronically, and on QIC-150 (6150) and 8mm tape and on CD-ROM. It consists of archive images for:

- The DCE source code and tests.
- The DCE documentation, including DCE specifications.
- The code and documentation building tools (ODE, SGML support files, etc.).
- Release information such as test results, defect summaries, and a quality report.
- Unsupported DCE source code (located in a subdirectory of the DCE source tree), containing bug fixes or features the code for which was completed too late in the release process to be incorporated in the DCE 1.2.2 source tree.

Instructions for extracting these archives on a target machine, as well as instructions for building and installing the sources in the **dce** archive, can be found in Chapter 4 of this document.

In addition, if licensees desire to build the new public key functionality (see ‘‘Public Key Support’’ in ‘‘Distributed Computing Environment’’ above), they must build DCE 1.2.2 with the RSA ‘‘BSAFE’’ encryption code. The RSA BSAFE 2.1 release, which is optional for building DCE 1.2.2 (but necessary for building the 1.2.2 public key functionality), is available *separately* from OSF on all of the DCE 1.2.2 distribution media. (BSAFE can be licensed directly from RSA or from OSF. See the DCE price sheet and licensing agreement for further information. BSAFE is available only to domestic DCE licensees.)

The following table shows the names of the archives in DCE 1.2.2 and their contents:

TABLE 1-1. The Archives in DCE 1.2.2

Archive Number	Contents Name	Description
1	**doc**	Documentation
2	**ode**	ODE Tools
3	**dce**	DCE Source
4	**project**	Additional information

The DCE 1.2.2 unsupported code is contained in a subtree of the **dce** archive.

1.3.1 Structure of DCE 1.2.2 Archives

Throughout this document, it is assumed that the DCE 1.2.2 archives have been unloaded (following the procedures described in Chapter 4) onto your system at a pathname specified as follows:

your-root-dir/**dce/V1.2.2**

—where *your-root-dir* is a partition with adequate disk space in which you have unloaded and installed the archives, as described in Chapter 4 of this document. This pathname is normally abbreviated as

dce1.2.2-root-dir

in the other chapters of this document.

Immediately beneath this pathname will be found several directories corresponding to the separate archives mentioned in the preceding section. For example, the directory

dce1.2.2-root-dir/**dce**

will contain the extracted **dce** archive, which consists of the DCE 1.2.2 source tree; and the directory

dce1.2.2-root-dir/**doc**

will contain the extracted **doc** archive, which consists of the DCE 1.2.2 documentation set (further described in Chapters 5 and 6).

Chapter 3 of this document contains graphical representations of the basic tree structure of all four of the DCE 1.2.2 archives.

1.3.2 Contents of DCE 1.2.2 Archives

The contents of the DCE 1.2.2 archives are as follows:

- Archive: **doc**

  This **tar** image contains documentation source files and the formatted output of the source in PostScript format. The DCE 1.2.2 documentation is distributed as SGML source; the SML source (in which form the DCE documentation has been distributed prior to DCE 1.2.2) is available electronically upon request. Contact OSF Direct at (617) 621-7300 for more information.

  In addition to a **README** file, the following subdirectories are located under **doc**:

  — **src**

  This directory contains four subdirectories:

  release_doc Contains the DCE 1.2.2 ‘‘release documentation’’, which consists of the *OSF DCE 1.2.2 Release Notes* (this book), the *OSF DCE Version 1.2.2 Testing Guide* (a new book in DCE 1.2.2), and the *DFA FVT User’s Guide* (the functional and

operational guide for running the DFA verification test suite).

The old 1.1 *OSF DCE Porting Guide* has not been updated for DCE 1.2.2; for DCE 1.2.2 test documentation, it has been replaced by the *Testing Guide*.

Note: The old *Porting Guide* has been included among the auxiliary files available to licensees in the AFS cell, as a convenience to licensees who may wish to consult it for information about ODE (the OSF Development Environment) and other DCE porting topics. For further information about these documents in DCE 1.2.2, see Chapter 5 of the *Release Notes* (this document).

Also included among the auxiliary files in the AFS cell are (1) the SML sources (and output), from which the **dce_books_sgml** were converted, (2) the **dte** directory containing the tools required to build the SML versions of the documentation, and (3) the **supp_docs** directory tree. The **supp_docs** directory in the AFS cell includes the contents that it has contained since DCE 1.0.3, and also, now, the old *Porting Guide*.

dce_books_sgml

Contains the DCE 1.2.2 documentation set (with the exception of the release documentation) in the form of SGML source. The content of this directory is described in detail in Chapters 5 and 6.

built_books Contains PostScript output for each book in the **dce_books_sgml** directory.

exec Contains the build scripts used at OSF to format the SGML sources. See Chapter 6 for details on formatting the SGML sources.

— **logs**

Build logs for the documentation builds.

— **DCE1.2.2_specs**

This directory contains functional specifications and RFCs from the suppliers and OSF that apply to DCE 1.2.2.

— **aes**

This subdirectory contains:

— *DCE: Remote Procedure Call* specification
Located in **aes/rpc**

— *DCE: Directory Services* specification
Located in **aes/directory**

— *DCE: Time Services* specification
Located in **aes/time**

— *DCE: Authentication and Security Services* specification
Located in **aes/security**

— *DCE: Distributed File Service* specification
Located in **aes/dfs**

— *DCE: Threads Services* (OSF)
Located in **aes/threads**

Note: With the merger of X/Open and OSF, the previous practice of maintaining two sets of specs will evolve into maintenance of a single set of CAE specs. Therefore, the OSF "Application Environment Specification" term is being phased out with the X/Open term "Common Application Environment" specification superceding.

The Distributed File Service specification is a Preliminary Specification reflecting DCE Version 1.1. It was published by X/Open in September, 1996.

The Remote Procedure Call specification is a draft reflecting DCE Version 1.1. It is in the process of being submitted for publication by X/Open as a CAE specification, and will be put through the X/Open fast-track process as part of this effort.

The Authentication and Security Services is a draft reflecting DCE Version 1.1. It is in the process of being submitted for publication by X/Open as a CAE specification, and will be put through the X/Open fast-track process as part of this effort.

The Directory Services specification is a draft reflecting DCE Version 1.1. It is in the process of being submitted for publication by X/Open as a CAE specification, and will be put through the X/Open fast-track process as part of this effort.

The Threads Services specification reflects DCE 1.1. It has not been published by X/Open.

The Time Services specification is a CAE specification.

All six documents, as described here, are supplied in the form of PostScript formatted output files, ready to print.

Contact X/Open at

XoSpecs@xopen.co.uk

or

http://www.xopen.org

or

X/Open Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

for final versions of the unpublished documents as well as publishing dates.

— **BUGREPORT**

This file contains a template for returning DCE defect information to OSF. For more information, see ‘‘Sending Feedback to OSF on DCE 1.2.2’’ at the end of this chapter.

Chapter 3 of this document contains a graphical representation of the basic structure of the **doc** tree.

- Archive: **ode**

  This **tar** image contains source code and documentation for the OSF Development Environment (ODE) tool set.

- Archive: **dce**

  This **tar** image contains: the source tree for the DCE and its integrated components; system tests; functional test suites; and source code for building the tools that are necessary for building the DCE 1.2.2 code to run on the reference platform.

  The DCE technology is divided into components, each of which is built from code contained in a separate subdirectory located immediately under **dce**. Chapter 3 contains a graphical representation of the basic structure of the entire **dce** tree, including all of the separate DCE component source subdirectories.

  The following directories are also included in the **dce tar** image:

  — **src/test**

  This subdirectory contains the source for the system test cases (in the **systest** subdirectory) and the functional test suites for each component (in separate subdirectories for each component), as well as source for the DCE sample program **timop** (in the **sample** subdirectory).

  Further information about the structure of the DCE test tree can be found in the *OSF DCE Testing Guide*.

  — **src/lbe**

  This subdirectory contains files necessary for building DCE.

  — **src/config**

  This subdirectory contains code and scripts for DCE configuration.

  — **src/libdce**

  This subdirectory contains the **Makefile** used to build the DCE library, **libdce.a**, **libdce.so** or **libdce.sl** (depending on the platform), which is a master static or shared library made up of other DCE libraries used by application programmers writing applications (note that on the AIX platform the library will be called

libdce.a, regardless of whether it is built static or shared).

— **src/examples**

This subdirectory contains source code for several sample DCE applications.

— **src/mit_krb5**

This subdirectory contains source code for building parts of the MIT Kerberos Version 5 authentication and key distribution service on the DCE 1.2.2 reference platform.

— **src/nosupport**

This directory contains DCE source code containing bug fixes or features which were completed too late in the release process to be incorporated in the DCE 1.2.2 source tree. In addition, the **nosupport** directory contains code for several features and add-ons which, although not part of DCE as such, are being included with DCE 1.2.2 for the convenience and information of licensees. The code in **src/nosupport** is not supported by OSF.

The

src/nosupport/README

contains descriptions of the contents of each of these directories. The following directories are included:

— **dced_fixes**

This subdirectory contains code to complete the **dced** functionality defined in the **dced** functional specification (for example, automatic starting of servers, etc.).

— **dfs_gg_fix**

This subdirectory contains code that contains the fix to the DFS global group problem (see OT CR 13681 for a description of the problem).

— **global_groups_tests**

This subdirectory contains code for a suite of manual tests of the new global groups feature.

— **scale_tests**

This subdirectory contains the tests used to verify the security scalability work.

— **K5Beta7**

This subdirectory contains documentation and source code to help licensees provide additional integration with the Kerberos authentication and key distribution service.

— **messaging**

This subdirectory contains code for the first snapshot of a prototype for Message-Oriented Services (MOS), which provides a message queueing capability for OSF DCE. The facility has been named ‘‘Message-Oriented Services’’ rather than simply ‘‘Queueing’’ in order to leave open the

possibility of including a message passing capability in the future.

The MOS is a DCE-based message queueing service, allowing clients to enqueue and dequeue messages via DCE RPC. Queues are protected by DCE ACLs, thus offering protection from unauthorized access. And since communication is done through DCE RPC, clients can take advantage of RPC security features.

— **pk_random**

This subdirectory contains code that can be used to develop a random number generator for public key security use. Note that the code as is does not build a generator random enough for truly secure environments; it must be made random by the vendor productizing the code.

— **rpc**

This subdirectory contains code to support the **RPC_SUPPORTED_NETADDRS** environment variable in the RPC runtime. This variable controls restriction of network address advertising and is located in the

/sbin/init.d/dce

script.

For further information, see the **README** in

src/nosupport/rpc

— **threads**

This subdirectory contains threads wrapper library code that provides a simple way for licensees to expose most of the Posix API, while still using the DCE threads library internally.

See Chapter 2 for further important information about the state of the various DCE components.

Chapter 3 of this document contains a graphical representation of the basic structure of the **dce** tree.

- Archive: **project**

This **tar** image contains information such as system test plans, test results, quality report, and an OT defect summary.

The **project** archive contains the following subdirectories and files:

— **defect.summary**

This subdirectory contains information about all known defects, both fixed (verified or closed) and open, for DCE 1.2.2. (It also includes the contents of the OT (DCE bug report) database. Whenever an OT defect report number is specified in this document, you should refer to the **defect.summary** subdirectory for more information on the defect.)

— **TestSummary**

This file contains the latest test information for functional and system tests and is organized by component.

— **test.results**

This directory contains the log files for all the functional and system tests run for each component.

— **quality.rpt**

Quality and testing report for DCE 1.2.2.

— **test.plans**

This directory contains the test plans for the DCE components.

Chapter 3 contains a graphical representation of the basic structure of the **project** tree.

1.4 Operating System Requirements for DCE 1.2.2

The DCE 1.2.2 reference platform is the IBM RISC System/6000 running the AIX 3.2.5 operating system (revision 3.2.5.E4.R9 or a later revision). Note also the following important information:

- Required **MALLOCTYPE** setting

  Before *running* any DCE code on AIX 3.2.5, you must set the **MALLOCTYPE** environment variable to a value of ‘‘3.1’’, which will cause the AIX 3.1 version of **malloc( )** to be used. This version of **malloc( )**, unlike later versions, is guaranteed to return zero-initialized space. DCE depends on this behavior.

- C++ runtime required for Private Key Storage Server

  PKSS is coded on both the client and server side in C++, and requires the C++ runtime to be installed on all client systems (prior to DCE 1.2.2, the C++ runtime was required only on those machines running applications developed with the XIDL C++ facility).

For further information on both of these topics, see ‘‘DCE 1.2.2 Reference Platform’’ in Chapter 4.

1.5 A Note on Pathnames

Most of the files discussed in this document are DCE source files, and are accordingly referred to by pathnames of the following general form

dce1.2.2-root-dir/**dce**/**src**/*rest_of_pathname*

the meaning of which is explained above in ‘‘Structure of DCE 1.2.2 Archives’’.

However, in cases where built or installed files (which usually lie outside the DCE source directory, **src**) are discussed, pathnames of other forms are used. These other forms are briefly described below. Note that the directory structure of all parts of the

DCE tree is a creature of ODE (the OSF Development Environment), and you can find overviews of both the structure as a whole and of ODE itself in Chapter 12 of the DCE 1.1 version of the *OSF DCE Porting and Testing Guide* (included with the DCE 1.2.2 documentation, although not updated for DCE 1.2.2; see Chapter 5 for further information).

1.5.1 Pathnames in the install Tree

The name

dce1.2.2-root-dir/**dce/install/***machine*/**opt/dce1.2.2**

is a path in the DCE install tree, which is established in *dce1.2.2-root-dir* as part of the DCE build. Part of the DCE installation process, done by **dce_config**, involves creating a link (called **/opt/dce1.2.2**) on the host machine to the **opt/dce1.2.2** part of the build tree. It is thus possible to refer to a built (and installed) file in the install tree in either of two ways: by its position in the DCE install tree —for example:

dce1.2.2-root-dir/**dce/install/***machine*/**opt/dce1.2.2/etc/cds_attributes**

or by its position on an installed system —for example:

/opt/dce1.2.2/etc/cds_attributes

(The first is the more rigorous form, since it is an exact description of what is actually built.) Pathnames of both forms occur in this document.

1.5.2 dcelocal Pathnames

The *dcelocal* prefix to a pathname—for example, in

dcelocal/**var/dced/cell_name**

—is an abbreviation that normally stands for the

/opt/dcelocal

directory set up by default by **dce_config** during cell configuration. Note that a *dcelocal* path has nothing to do with ODE or the DCE build; it is part of an installed and configured DCE cell. The difference between the contents of

/opt/dce1.2.2

and of *dcelocal*, which stands for

/opt/dcelocal

is that the latter consists of the necessary subset of DCE executables and other files that must be locally present on every machine in a DCE cell; **/opt/dce1.2.2**, on the other hand, contains the entire built and installable DCE, and is often a remote filesystem that need only be mounted.

For further information about the directory structure of an installed DCE, see Chapter 5 (‘‘Location of Installed DCE Files’’) of the *OSF DCE Administration Guide—Introduction*.

1.5.3 Pathnames of Installed Tests

The **dcetest_config** installation script installs the DCE tests at a pathname specified by the user, and then creates a softlink called

/dcetest/dcelocal

to that location. The built tests can also be found at

dce1.2.2-root-dir/**dce/install/***target_platform*/**dcetest/dce1.2.2/test**

(where *target_platform* denotes the specific platform on which you are building; for the DCE 1.2.2 reference platform this would be **rios**) after a DCE build, or, in the case of the DCE Threads functional tests (which are left in the **obj** tree) at:

dce1.2.2-root-dir/**obj/***target_platform*/**test/threads**

Thus it is possible to refer to tests in either of two ways, depending on whether its built or its installed location is used. Pathnames of both forms appear in this document.

For further information about **dcetest_config**, refer to Chapter 11 of the *OSF DCE Testing Guide*.

1.6 Testing in DCE 1.2.2

For DCE 1.2.2, various types of testing were planned to confirm that new functionality was functionally complete, worked as specified, and caused no regression in reliability of previous functionality.

During the test planning phase, OSF, along with each provider cooperatively selected the group of tests that delivered the best test coverage within the allocated amount of test time. The test plans can be found at:

dce1.2.2-root-dir/**project/test.plans**

Each provider’s test plan lists the new tests which were developed to test DCE 1.2.2 functionality. For information on the results of the testing conducted of DCE 1.2.2, see the DCE 1.2.2 quality and testing report, located at:

dce1.2.2-root-dir/**project/quality.rpt**

Individual test reports can be found in the

dce1.2.2-root-dir/**project/TestSummary**

file.

The DCE defects database is located in the

dce1.2.2-root-dir/**project/defect.summary**

directory.

For information on running the tests, see the *OSF DCE Testing Guide*.

1.7 Sending Feedback to OSF on DCE 1.2.2

To facilitate communication between OSF and its DCE 1.2.2 licensees, we have set up the **dce-talk** mailing list. Licensees may also find the discussions in the **comp.soft-sys.dce** newsgroup useful.

Use **dce-defect** when reporting any defects in functionality that you find with DCE 1.2.2 code. The **dce-defect** address is an alias to the OSF Response Center Administrator. Your report will be acknowledged and researched. When reporting defects to **dce-defect**, please use the form in the **BUGREPORT** file in the top level of the documentation directory. The complete pathname of this template is:

dce1.2.2-root-dir/**doc/BUGREPORT**

You may also use the **dce-talk** mailing list to discuss the offering. This address is a mailing list to all DCE licensees who wish to subscribe to it, and not just 1.2.2 licensees. This discussion group was designed to increase the general knowledge of DCE, and to collect input regarding the functionality and use of the offering. As such, the opinions given in **dce-talk** are strictly those of the individual who expresses them.

To use the mailing lists, simply send mail to:

dce-talk@osf.org

To add your organization's address to **dce-talk**, send a request to:

dce-talk-request@osf.org

When making a request to be added to a mailing list, please include the following information:

your name
your phone number
your fax number
your company name
your company address
the name of the person who receives DCE 1.2.2 archives

We prefer to add names of licensees' internal mailing lists (for example, **dce@licensee.com**) rather than adding a large number of individuals. Note that these mailing lists do not take the place of our support services as regards detailed consulting, problem status tracking, and so forth. Licensees who are OSF members may also wish to subscribe to the **sig-dce** mailing list, which is used to distribute information about the DCE SIG.

Licensees who have questions about OSF's offerings, such as courses, license policies, or questions about software delivery, can call OSF at (617) 621-7300.

Chapter 2. State of the DCE Components in DCE 1.2.2

The following sections contain important information about the state of the DCE version 1.2.2 code and tests.

2.1 Pathnames

It is assumed that the DCE 1.2.2 archives are located at the following pathname on your system:

your-root-dir/**dce/V1.2.2**

—where *your-root-dir* is a partition with adequate disk space in which you have unloaded and installed the archives, as described in Chapter 4 of this document. (Refer to the section ‘‘Contents of the DCE 1.2.2 Release Area’’ in Chapter 1 for more information.) This pathname is abbreviated as

dce1.2.2-root-dir

in the other chapters of this document.

Immediately beneath this pathname will be found several directories corresponding to the separate archives extracted from the DCE 1.2.2 distribution media, as described in Chapter 4. For example, the directory

dce1.2.2-root-dir/**dce**

will contain the extracted **dce** archive, which consists of the DCE 1.2.2 source tree; and the directory

dce1.2.2-root-dir/**doc**

will contain the extracted **doc** archive, which consists of the DCE 1.2.2 documentation set (further described in Chapters 5 and 6).

2.2 Information on the DCE Build

The following sections contain important new information concerning various aspects of building DCE.

2.2.1 Building an Exportable Version of DCE

Note: It is the licensee's responsibility to determine if the version of DCE that has been built is indeed exportable, based on current U.S. export regulations. Effort has been made in the DCE source to facilitate generating an exportable version if such is desired, but it is not possible to guarantee under all circumstances, without testing, that what is built will be exportable.

The exportability of a given binary version of DCE is controlled by U.S. export regulations and depends on the methods of data encryption used, as well as what they are used for. In general, use of encryption for data privacy is export-controlled, but use of encryption for ''other purposes'' (such as authentication, authorization, integrity, password change, and so on) is not export-controlled. In particular, generally exportable binaries must have their encryption routines' entry point symbols either removed or obscured. This cannot be provided for at the source code level; it is a decision that must be made by the individual vendor or ISV.

By this criterion, DCE can be built in any of three different forms:

- Domestic executables from domestic source (cryptographic software is fully present and is used both for data privacy and for other purposes).
- Exportable executables from exportable (''international'') source (no cryptographic code is present).
- Exportable executables from domestic source (cryptographic code is not used for data privacy, but is used for other purposes).

Whether one builds DCE in the first or the second form is governed by the contents of the

dce1.2.2-root-dir/**dce/src/security/krb5/lib/crypto**

and

dce1.2.2-root-dir/**dce/src/rpc/kdes**

directories. The first, in its fullest state, contains three subdirectories, named **common**, **domestic**, and **international**. The second contains a DES implementation suitable for use in the kernel.

The domestic source version of DCE will contain all three **crypto** subdirectories. Building the domestic version of DCE will consist in simply executing a build with these directories (and their contents) present. Constructing an exportable source version of DCE consists in replacing the contents of the **crypto/domestic** with those of the **crypto/international** directory; building DCE with the contents of these two directories

will yield an exportable set of executables in which the DES functionality has been replaced with ‘‘null’’ encryption. This version of DCE will not interoperate with *any* domestic DCE at all, but it is internally consistent and runnable.

The default behavior of the DCE build as a whole is to produce a non-exportable version of DCE (the first method listed above), since the symbol **INTL_BUILD** must be defined in order to disable confidentiality (data privacy) in both GSSAPI and RPC, and by default it is not defined.

The exportable source version of DCE 1.2.2 was produced and built at OSF in the following way:

1. A separate ‘‘international’’ source tree was created, consisting of the normal contents of the DCE 1.2.2 source tree, with the contents of the **crypto/domestic** replaced (as described above) by the contents of the **crypto/international** directory.
2. A **INTL_BUILD** symbol was created and defined in the

   dce1.2.2-root-dir/**dce/src/Makeconf**

   The value of **INTL_BUILD** determines whether **NOENCRYPTION** is defined or not. If **INTL_BUILD** is defined, then **NOENCRYPTION** is also defined.

Note that having **INTL_BUILD** defined in a build should be regarded only as a guide to what it is expected will be produced by the build. OSF assumes no responsibility for validating the exportability of versions of DCE produced by any of the methods described here.

To test whether the GSSAPI you have built is exportable or not, you must run the GSSAPI test program. Instructions on how to do this can be found in Section 8.2.5 in Chapter 8 of the *OSF DCE Testing Guide*. When the test is executed, it will display a warning message if the GSSAPI being tested is non-exportable.

2.2.1.1 Exportability of Secure Remote Utilities

For U.S. vendors and ISVs, OSF DCE provides partial (source-code level) support for building an internationally exportable version of OSF DCE from the domestic sources. However, this does not necessarily extend to full (binary-code level) support for building exportable binaries. Vendors and ISVs are responsible for determining how to build such a product for their platforms, and for verifying that the resulting product is indeed exportable.

At a minimum, you must do the following to ensure that the secure remote utilities (**rsh**/**rshd**, **rlogin**/**rlogind**) limit use of encryption:

- Define **DES_HIDDEN** for the build of any component that would support encryption and/or decryption in the domestic build.

No matter how the above suggestion is accomplished, it should be understood that it constitutes limited source-code level support only, not full binary-code level support for building the domestic DCE sources for export. Each vendor and ISV remains responsible for making sure that the above steps and any additional work necessary to hide and

remove encryption result in a product that complies with the applicable export laws.

2.2.1.2 Exportability of Public Key Preauthentication

For U.S. vendors and ISVs, OSF DCE provides partial (source-code level) support for building an internationally exportable version of OSF DCE from the domestic sources. However, this does not necessarily extend to full (binary-code level) support for building exportable binaries. Vendors and ISVs are responsible for determining to determine how to build such a product for their platforms, and for verifying that the resulting product is indeed exportable.

At a minimum, you must do the following to ensure that Public Key Login limits its use of encryption:

- Define **DES_HIDDEN** for the build of any component that would support encryption and/or decryption in the domestic build.

No matter how the above suggestion is accomplished, it should be understood that it constitutes limited source code level support only, not full binary code level support for building the domestic DCE sources for export. Each vendor and ISV remains responsible for making sure that the above steps and any additional work necessary to hide and remove encryption result in a product that complies with the applicable export laws.

In addition to export restrictions, there are also license restrictions on the exposure of the RSA BSAFE interfaces that must be handled by each vendor or ISV. Each vendor and ISV remains responsible for ensuring that any product produced from the OSF DCE source complies with the terms of the RSA license agreement.

To produce an exportable product, a vendor should do the following:

1. Make the necessary **libdce Makefile** change, so that **libbsafe2.a** is linked into shared **libdce** in an exportable manner. Please note that to ensure export inaccessibility and potential licensing issues, we do not recommend linking the BSAFE library into the **libdce** archive.
2. Do

   nm libbsafe2.a

   to extract all of the symbols in the BSAFE library, and then take whatever steps are necessary to hide or scramble all of the BSAFE symbols in order to fulfill the export requirements and potential licensing issues.

2.2.2 DCE 1.2.2 Build Dependency on BSAFE

Licensees who wish to use the new DCE 1.2.2 public key functionality (see ‘‘Public Key Support’’ in ‘‘Distributed Computing Environment’’ above) must build DCE 1.2.2 with the RSA ‘‘BSAFE’’ encryption code. The RSA BSAFE 2.1 release, which is optional for

building DCE 1.2.2 (but necessary for building the 1.2.2 public key functionality), can be licensed either from OSF or directly from RSA. From OSF BSAFE is available on all of the DCE 1.2.2 distribution media. BSAFE is available only to domestic DCE licensees.

For instructions on how to build BSAFE on the DCE 1.2.2 reference platform, see ''Building BSAFE'' in Chapter 4.

BSAFE can be licensed directly from RSA or from OSF. See the DCE price sheet and licensing agreement for further information.

2.3 State of DCE 1.2.2

All of the DCE 1.2.2 code has undergone functional testing, and many components have undergone system testing. For information on the system tests performed, see the ''State of DCE System Testing'' section below.

2.3.1 Texts of Currently-Known DCE Defect Reports

A collection of bug reports (CRs) on all currently-known DCE defects, including those fixed since DCE 1.2.1, is located at:

dce1.2.2-root-dir/**project/defect.summary**

The contents of this directory consist of pairs of groups of files (seven files in each group), consisting of the ''long'' and ''short'' listing of every recorded defect for each DCE component. The defect state of each DCE 1.2.2 component is thus represented by the contents of 14 files, containing the long- and short-form texts for all **open**, **fix**ed, **verified**, **closed**, **dup**licate, **defer**red, and **cancel**ed bugs recorded for that component. For example, the names of the defect files for the Audit component are as follows:

Long form text	*Short form text*
aud.cancel	**short.aud.cancel**
aud.closed	**short.aud.closed**
aud.defer	**short.aud.defer**
aud.dup	**short.aud.dup**
aud.fix	**short.aud.fix**
aud.open	**short.aud.open**
aud.verified	**short.aud.verified**

The DCE component names and their meanings are as follows:

aud	DCE Audit
bld	DCE Build and Installation procedures

cds	DCE Cell Directory Service
cfg	DCE Configuration routines
code	DCE-wide source code category
dcecp	DCE Command Program (**dcecp**)
dced	DCE Daemon (**dced**)
dcedoc	General DCE-wide documentation category
dfam	DCE Distributed File-Access Manager (DFA)
dfs	DCE Distributed File Service
doc	DCE documentation
dskl	DCE Diskless component (not supported)
dts	DCE Distributed Time Service
gds	DCE Global Directory Service
rpc	DCE Remote Procedure Call
sec	DCE Security Service
systest	DCE system testing
test	DCE functional testing
thr	DCE Threads
unknown	Unknown component
utils	DCE utilities

A summary of DCE 1.2.2 test results can be found in the

dce1.2.2-root-dir/**project/TestSummary**

file. A report on functional test results for DCE 1.2.2 can be found in subdirectories under:

dce1.2.2-root-dir/**project/test.results**

A quality report for DCE 1.2.2 can be found at:

dce1.2.2-root-dir/**project/quality.rpt**

Further information on DCE testing procedures can be found in the *OSF DCE Testing Guide*.

2.3.2 Components Unchanged in DCE 1.2.2

No new functionality was added to the following DCE components and facilities in DCE 1.2.2:

- **dced**

- General ACL and Backing Store Facility
- DCE Threads
- IDL Compiler
- Cell Directory Service (CDS)
- Global Directory Service (GDS)
- DCE DTS
- Generic Security Service Application Program Interface (GSSAPI)

2.3.3 DCE Distributed File-Access Manager Component

Note: To successfully run the DFAM 48 hours CHO, you must set the Agent-Gateway waiting interval time from 20 to 200. To do this, you do not need to fix the DFAM source code to change the waiting interval from 20 to 200. This is because the waiting interval time is one of the DFAM Agent parameters you can set before starting DFAM, and DFAM Agent reads the parameter file when initiated.

Here is how you can set 200 to the waiting interval parameter:

1. Generate a file that contains the following line:

   TCPShutdownTimeout 200

2. If the file name of the above one-line file is **paramfile**, use the following command to start DFAM Agent:

   dfamagt -ConfigFile paramfile

DCE 1.2.2 work on the DCE Distributed File-Access Manager (hereafter abbreviated as ‘‘DFA’’), which was new in DCE 1.2.1, was restricted to bug fixes. The following CRs were closed: 13586, 13585, 13584, 13583, 13582, 13570, 13569, 13568, 13567. Refer to the collection of bug reports (CRs) on all currently-known DCE defects, including those fixed since DCE 1.2.1, located at:

dce1.2.2-root-dir/**project/defect.summary**

for further information. Other release information about DFA, as well as porting and other miscellaneous information, can be found in Chapter 7 of this document.

2.3.4 State of the DCE Installation and Configuration Routines

A collection of bug reports (CRs) on all currently-known DCE defects, including those fixed since DCE 1.2.1, is located at:

dce1.2.2-root-dir/**project/defect.summary**

2.3.5 State of dcecp

A collection of bug reports (CRs) on all currently-known DCE defects, including those fixed since DCE 1.2.1, is located at:

dce1.2.2-root-dir/**project/defect.summary**

- The **dcecp host stop**, **dcecp host show**, and **dcecp host start** commands do not work in DCE 1.2.2.

  This is because **dcecp** does not have the ability to track the **srvrexec** object of **secd**, **cdsd**, and the other DCE daemons. Even if the **srvrconf** objects are correctly set up, when they are run (using **host start** or **server start**), what actually happens is that the daemons start, fork a child, and then die, with the forked children remaining up to run as the daemons. But, as far as **dced** (or **dcecp**) is concerned, the servers that were started (although they did start) subsequently died, and as a result **dced** (or **dcecp**) assumes they are not running anymore.

  This problem is documented in DCE defect report 12916, which can be found in the DCE defects database, located in the

  dce1.2.2-root-dir/**project/defect.summary**

  directory.

  Note that, although the code for 1.2.2 does not support these **dcecp** commands, you can use **dce_config** to start or stop daemons.

  Note that aliases do not work automatically across cells, even within a hierarchy. This situation, which was reported in the DCE 1.2.1 *Release Notes* and is still true for DCE 1.2.2, is documented in DCE defect report 12908, which can be found in the DCE defects database, located in the

  dce1.2.2-root-dir/**project/defect.summary**

  directory.

- The **dcecp cellalias set** command, which is currently described in the *OSF DCE Administration Guide -- Core Components*, is not supported in DCE 1.2.2. This operation was intended to support cell renaming, which is not supported in version 1.2.2 of DCE, and is not planned to be supported in any future version of DCE.

  This situation, which was reported in the DCE 1.2.1 *Release Notes* and is still true for DCE 1.2.2, is documented in DCE defect reports 12917, 12908, and 12864, which can be found in the DCE defects database, located in the

  dce1.2.2-root-dir/**project/defect.summary**

  directory.

2.3.5.1 State of dcecp Functional Testing

A summary of DCE 1.2.2 **dcecp** test results can be found in the

dce1.2.2-root-dir/**project/TestSummary**

file. A report on **dcecp** functional test results for DCE 1.2.2 can be found in:

dce1.2.2-root-dir/**project/test.results**

The procedures for running the **dcecp** functional tests are documented in Chapter 2 of the *OSF DCE Testing Guide*.

2.3.6 State of dced

No new functionality was added to **dced** for DCE 1.2.2.

The format of the databases that **dced** uses to keep its configuration information was changed as a bug fix to DCE 1.1, incorporated into DCE 1.2.1. Refer to OT CR 13662 for a description of the problem that was fixed.

Upon migration from a DCE 1.1 to DCE 1.2.2 code base, **dced** will spend an increased amount of time in initialization, during which it will convert any DCE 1.1-formatted databases into the 1.2.2 format. The additional amount of time taken will vary with the size of the database. This behavior may have an effect on any initialization scripts which expect **dced** to complete initialization within some definite period of time. Note that performance is affected only during migration.

The fix affects licensees who did not pick up the DCE 1.2.1 warranty patch, in which the **dced** database format was revised.

A collection of bug reports (CRs) on all currently-known DCE defects, including those fixed since DCE 1.2.1, is located at:

dce1.2.2-root-dir/**project/defect.summary**

A summary of DCE 1.2.2 test results can be found in the

dce1.2.2-root-dir/**project/TestSummary**

file. A report on functional test results for DCE 1.2.2 can be found in:

dce1.2.2-root-dir/**project/test.results/**

The procedures for running the **dced** functional tests are documented in Chapter 2 of the *OSF DCE Testing Guide*.

2.3.7 State of the DCE RPC Code

The following enhancement was made to the RPC component for DCE 1.2.2:

- Single-threaded client datagram support

  For the datagram protocol, the multi-thread model on the client side was changed to a single thread model.

A collection of bug reports (CRs) on all currently-known DCE defects, including those fixed since DCE 1.2.1, is located at:

dce1.2.2-root-dir/**project/defect.summary**

2.3.7.1 State of DCE RPC Testing

A summary of DCE 1.2.2 RPC test results can be found in the

dce1.2.2-root-dir/**project/TestSummary**

file. A report on RPC functional test results for DCE 1.2.2 can be found in:

dce1.2.2-root-dir/**project/test.results**

The RPC functional tests are in the

dce1.2.2-root-dir/**dce/src/test/rpc**

directory; this includes the RPC runtime internationalized functional tests. There is a **README** file in the directory explaining how to set up the environment and how to execute the tests. The following are general requirements for executing the tests:

The OSF character and code set registry should be installed as:

/usr/lib/nls/csr/code_set_registry.db

This is a binary file, which is produced by **csrc** (code set registry compiler). The input file to **csrc** is platform dependent, and will be found in the following directory:

dce1.2.2-root-dir/**dce/src/test/functional/rpc/runtime/i18n_api/ts/cs_rgy**/*target_platform*

where *target_platform* is the name of the platform which you have built and are testing DCE on (for the AIX reference platform, *target_platform* will be **RIOS**).

These **csrc** input files are not the OSF official version. The OSF official template for the codeset registry is found in:

dce1.2.2-root-dir/**dce/src/rpc/csrc/csr**/*target_platform*/**code_set_registry.txt**

For functional testing only, the

dce1.2.2-root-dir/**dce/src/test/functional/rpc/runtime/i18n_api/ts/cs_rgy**/*target_platform*/**code_set_registry.txt**

file is ready to use.

Japanese EUC and SJIS locales are required. This is because input data are Japanese. However, if the contents of **i18n_input_data** are changed to other data, for example, French, the other locale (French locale) will be required.

The procedures for running the other RPC functional tests are documented in Chapter 4 of the *OSF DCE Testing Guide*.

2.3.8 State of Cell Directory Service (CDS) Code

No new functionality was added to CDS for DCE 1.2.2.

The following CDS OT CRs were fixed or closed in DCE 1.2.2:

CR number	File(s) affected
13581	**directory/cds/library/dnscmpsimple.c**
13208	**directory/cds/server/db_diags.c** **directory/cds/server/back_ground.c** **directory/cds/server/ta_cle.c**
13085	**directory/cds/server/back_skulk.c** **directory/cds/server/ta_create.c** **directory/cds/server/ta_lookup.c**
12932	**directory/cds/server/ta_cle.c**
12914	**directory/cds/control/get_command.c**
12987	(duplicate of 12879)
12879	**directory/cds/includes/dbdef.h** **directory/cds/server/db_btree.c** **directory/cds/server/db_entry.c**
12819	**directory/cds/control/dnscp_crekid.c** **directory/cds/control/emit_ref_hdr.c** **directory/cds/control/visit_cds.c** **directory/cds/gda/gda_update_parent.c** **directory/cds/server/unix_stubs.c**
12812	(duplicate of 12802)
12802	**directory/cds/server/dns_service_rpc.c**
12767	**directory/cds/server/sets_lib.c**
12465	**directory/cds/includes/cds.sams**
12453	(duplicate of 12879)
12389	**directory/cds/library/x500_name_utils.c**
9700	**directory/cds/server/back_ground.c** **directory/cds/server/db_common.c** **directory/cds/server/dump_navigator.c** **directory/cds/server/ta_cle.c**

directory/cds/server/unix_cds_net.c

The following CDS OT CRs were cancelled for reasons stated in the CR:

13004, 11405, 11508, 11272, 10777, 9514, 9065, 8081

A collection of bug reports (CRs) on all currently-known DCE defects, including those fixed since DCE 1.2.1, is located at:

dce1.2.2-root-dir/**project/defect.summary**

Cell renaming is not supported in DCE 1.2.2, and aliases do not work automatically across cells, even within a hierarchy.

The code for DCE 1.2.2 does not support the **dcecp cellalias set** command, which is currently described in Chapter 21 of the *OSF DCE Administration Guide*. This operation was intended to support cell renaming, which is not supported in version 1.2.2 of DCE, and is not planned to be supported in any future version of DCE.

This situation is documented in DCE defect reports 12917, 12908, and 12864, which can be found in the DCE defects database, located in the

dce1.2.2-root-dir/**project/defect.summary**

directory.

2.3.8.1 State of DCE CDS Testing

A summary of DCE 1.2.2 CDS test results can be found in the

dce1.2.2-root-dir/**project/TestSummary**

file. A report on CDS functional test results for DCE 1.2.2 can be found in:

dce1.2.2-root-dir/**project/test.results**

The procedures for running the CDS functional tests are documented in Chapter 5 of the *OSF DCE Testing Guide*.

2.3.9 State of Global Directory Service (GDS) Code

No new functionality was added to GDS for DCE 1.2.2.

In the course of developing the DCE certification facility for DCE 1.2.2, deficiencies were discovered in GDS that effectively prevent it from being able to usefully store public key certificates. While the certification API facility includes workarounds for these deficiencies, licensees are advised to read OT CR reports 13660 and 13661 and make sure they understand the implications before deploying GDS as the directory for an interoperable public key infrastructure.

The instructions for configuring and starting GDS in Chapter 6 of the *OSF DCE GDS Administration Guide and Reference* are incomplete. Refer to OT CR 13654 for a set of

correct instructions.

A collection of bug reports (CRs) on all currently-known DCE defects, including those fixed since DCE 1.2.1, is located at:

dce1.2.2-root-dir/**project/defect.summary**

2.3.9.1 State of DCE GDS Testing

A summary of DCE 1.2.2 GDS test results can be found in the

dce1.2.2-root-dir/**project/TestSummary**

file. A report on GDS functional test results for DCE 1.2.2 can be found in:

dce1.2.2-root-dir/**project/test.results**

The procedures for running the GDS functional tests are documented in Chapter 6 of the *OSF DCE Testing Guide*.

2.3.10 State of Security Code

Note: DCE 1.2.2 contains an optional component, the Private Key Storage Service (PKSS). The implementation of this component requires a **setuid-root** program on each client system that supports the service. There is a potential security issue with this program on platforms that provide both shared libraries and user-configurable library search-paths, which must be addressed by platform-specific code. OT CR 13688 describes this problem, as well as the solution that DCE 1.2.2 provides for the reference platform. While this solution is AIX-specific, a similar solution is expected to apply to other platforms; the reference code should be used as a template for the platform-specific code that will be required.

Note: MUST APPLY OT

Configuration failures have been reported for DCE 1.2.2 international code on AIX. The **dce_login** during **dce_config** was failing because of memory corruption in the PK preauth code. Licensees should refer to OT CR 13682 for the fix to apply.

Note: Very late in the test cycle a critical problem was found in the implementation of the database code that is used by PKSS. Under some circumstances it is possible for the database to be corrupted. The problem is documented in OT CR 13693. Since PKSS is the only component using this specific database code PKSS is the only component affected by the problem.

: Late in the project, a problem was discovered that prevented invoking **dce_config**. Several unsuccessful attempts were made to reproduce the problem. FVTs and CHO testing continued to pass. OT CR 13699 contains a description of the problem. As

more data becomes available from continued problem resolution, it will be placed within this OT.

The following enhancements and additions to the Security component were made for DCE 1.2.2:

- Public key login

  The login facility now supports the public key protocol. The public key preauthentication protocol is used by DCE security clients and servers to obtain ticket granting tickets (TGTs) for users. This protocol, which was not available in previous releases of DCE, provides the highest level of security during preauthentication login.

  Note that public key functionality in DCE 1.2.2 depends on the presence of the RSA ‘‘BSAFE’’ encryption functionality, with which DCE 1.2.2 must be built if public key login is desired. See ‘‘DCE 1.2.2 Build Dependency on BSAFE’’ above for further information.

  Users can enter public key password information by either one of the following methods:

  — Using the **dcecp** commands **account create** and **account modify**, which prompt the user for public key password information (*oldpassphrase* and *newpassphrase*). This method is more secure because the *passphrase*s are not displayed.

  — Using the command line interface to specify the *oldpassphrase* and *newpassphrase*. This method is less secure because the *passphrase*s are visible.

  Following is an example of using the **dcecp** commands to interactively enter password information:

```
dcecp> account create gumby -group none -org none -password gumby \
> -mypwd -dce-
dcecp>

dcecp> account create gumby -group none -org none -password gumby
Enter Your Password:
dcecp>

dcecp> account create gumby -group none -org none
Enter Account Password:
Again:
Enter Your Password:
dcecp>

dcecp> account modify gumby -mypwd -dce- -password pokey
dcecp>

dcecp> account modify gumby -mypwd -dce- -password ""
Enter Account Password:
Again:
dcecp>
```

```
dcecp> account create gumby -group none -org none -password gumby \
> -mypwd -dce- -pksig {{gen 256} {newpass pokey}} \
> -pkkey {{gen 512} {newpass pokey}}
Warning: Generating pksignature public key; this may take a few minutes.
Warning: Generating pkkeycipher public key; this may take a few minutes.
dcecp>

dcecp> account create gumby -group none -org none -password gumby \
> -mypwd -dce- -pksig {gen 256} -pkkeycipher {gen 512}
Enter passphrase for signature key:
Again:
Enter passphrase for key encryption key:
Again:
Warning: Generating pksignature public key; this may take a few minutes.
Warning: Generating pkkeycipher public key; this may take a few minutes.
dcecp>

dcecp> account modify gumby -pksig {{oldpass ""} {newpass ""}}
Enter old passphrase for signature key:
Again:
Enter new passphrase for signature key:
Again:

dcecp> account modify gumby -pkkey {{gen 256} {newpass ""}}
Enter new passphrase for key encryption key:
Again:
Warning: Generating pkkeycipher public key; this may take a few minutes.
dcecp>
```

Note that this information is not documented in the DCE 1.2.2 versions of the *OSF DCE Administration Guide — Core Components* volume or the *OSF DCE Command Reference*.

- User-to-user authentication

  Allows applications that do not have access to a principal's long term key to receive protected RPCs.

  Note: In order to make full use of the user-to-user protocol in DCE 1.2.2 (for example, via authenticated RPC instead of only through Kerberos), the fix to a memory leak-associated problem described in OT CR 13686 must be applied to DCE 1.2.2 by licensees. For further details see the text of the OT.

- Global groups

  Principals from foreign cells can be added to a group in a local cell. The **nosupport** directory contains a required fix in **dfs_gg_fix**. See OT CR 13681.

  See OT CR 13696 for diffs for the suggested global groups threadsafe fix.

- Scalability

  Security performance for large databases has been improved.

- Kerberos 5 support

  Support for DCE KDC and V5 clients has been added. This allows Kerberos V5 applications running on either DCE or non-DCE platforms to access the DCE Security Service. Included is support for the following utilities:

  — **rlogin**

  — **rlogind**

  — **rsh**

  — **rshd**

  The authentication portion of the DCE Security Service is primarily based on Version 5 of the Kerberos network authentication system, which is described in Internet Engineering Task Force (IETF) RFC 1510. For the most part, this has allowed the DCE Security Server to operate as a Kerberos Key Distribution Center (KDC) for Kerberos V5 clients. In prior releases of DCE (Versions 1.2.1 and earlier), this interoperability was not officially supported or documented. In DCE 1.2.2, this interoperability and the remote utilities are described in a new chapter of the *OSF DCE Administration Guide —Core Components* volume.

- X.509 Certificates

  The DCE certification API is a new component in DCE 1.2.2. It implements a trusted public-key retrieval service, based on either X.509 certificates or trusted registry storage of public keys. The certification API is described in OSF RFC 80.1.

  If your DCE build includes the reference implementation of GDS (in particular the XDS/CDS switch layer), and you wish to use CDS as a certificate store, you must apply the patch documented in OT CR 13665. This patch fixes the following two problems in the XDS API to CDS:

  — Access to a CDS root directory is prohibited.

  — Octet-string attribute values are copied using **strncpy** (instead of **memcpy**), so internal 0 octets will truncate the attribute.

  Without fixes for these problems, the certification API CDS tests (**pc1**- **pc7**) will not run successfully.

  Since the certification API was shipped to OSF, the following problems have been discovered, and fixes for them have been placed in the OT system.

  — Support for multiple certificate signature algorithms is broken. The code as shipped supports only **MD2withRSA**. The fix for this (enabling built-in support for **MD5withRSA** and allowing user/vendor registration of additional algorithms) is documented in OT CR 13664.

  — The certification API test build does not properly install the tests. There is also a problem in that certain test failures may not appear in the test log when running the tests under TET. The fix for both of these problems is documented in OT CR 13667.

  Note: See Section 2.3.9 (‘‘State of Global Directory Service (GDS) Code’’) and OT CRs 13660 and 13661 for information about problems and workarounds for storing certificates using GDS.

Note that public key functionality in DCE 1.2.2 depends on the presence of the RSA ‘‘BSAFE’’ encryption functionality, with which DCE 1.2.2 must be built if use of public keys is desired. See ‘‘DCE 1.2.2 Build Dependency on BSAFE’’ earlier in this chapter for further information.

- Modifying ACLs on the Master Security Server

  To allow more flexibility in remote administration, ACLs on certain **dced** objects can be modified by using the **dced_acl_patcher** script on the master Security Server. This script was supported but not documented in prior versions of DCE.

A collection of bug reports (CRs) on all currently-known DCE defects, including those fixed since DCE 1.2.1, is located at:

dce1.2.2-root-dir/**project/defect.summary**

The following items give important information about security in DCE 1.2.2 additional to that provided in the DCE documentation set.

- **secd** no longer implements the server side of the **sec_acl_test_access_on_behalf( )** functionality, as there is no proper use for it and it was present by mistake in the 1.0.*x* versions of DCE.
- Transitive trust is not fully supported in DCE 1.2.2: transited intermediate cell principals in a delegation chain are not automatically registered in the security namespace. The problem is documented in DCE defect report 12908, which can be found in the DCE defects database, located in the

  dce1.2.2-root-dir/**project/defect.summary**

  directory.
- **sec_salvage_db** now uses the DCE message API. (Note also that the reference page for **sec_salvage_db**, which was moved in DCE 1.1 to an Appendix in the *Release Notes*, is back in the *OSF DCE Command Reference* for DCE 1.2.2.)

2.3.10.1 Random Number Generation for Public Key in Secure Environments

A truly secure environment based on public keys requires a random number generator capable of producing a ‘‘highly random’’ number. This highly random number is then the seed value which is fed to the process that creates a public key pair.

Two random number generators are included with DCE 1.2.2:

- **crypto_randomize( )**, a routine defined in the **sec_bsafe.c** code.

  This program is not likely to produce numbers that are random enough for truly secure environments; it uses a simple algorithm to generate a number that is not highly random. Vendors are responsible for supplying better random numbers to this routine.
- **/dev/random**, which is built from unsupported code located in:

  dce1.2.2-root-dir/**dce/src/nosupport/random_num_gen**

This program is provided as example code and is to be used only as a guide reference implementation. The code can be regarded as a starting point for vendors who want to develop a kernel driver.

Note that the two programs can be used together to produce a highly random number. After porting **/dev/random**, a vendor can use it as a reference implementation to produce highly random numbesr which can then be fed into the **crypto_randomize( )** routine.

2.3.10.2 Security Functional Testing

A summary of DCE 1.2.2 security test results can be found in the

dce1.2.2-root-dir/**project/TestSummary**

file. A report on security functional test results for DCE 1.2.2 can be found in:

dce1.2.2-root-dir/**project/test.results**

The procedures for running the security functional tests are documented in Chapter 8 of the *OSF DCE Testing Guide*.

2.3.11 State of the DCE Auditing Code

DCE 1.2.2 work on the Audit component was confined to bug fixes. No new functionality was added. The format of the audit trail record was restructured to improve its appearance (refer to DCE defect report 13195 for further details).

A collection of bug reports (CRs) on all currently-known DCE defects, including those fixed since DCE 1.2.1, is located at:

dce1.2.2-root-dir/**project/defect.summary**

2.3.11.1 State of DCE Audit Testing

A summary of DCE 1.2.2 audit test results can be found in the

dce1.2.2-root-dir/**project/TestSummary**

file. A report on audit system test results for DCE 1.2.2 can be found in:

dce1.2.2-root-dir/**project/test.results**

The procedures for running the audit system tests are documented in Chapter 12 of the *OSF DCE Testing Guide*.

2.3.12 State of the DCE DFS Code

The following new functionality was added to DFS for DCE 1.2.2:

- Server multi-home support

  In DCE 1.2.2 the DFS servers have been enhanced to perform better on hosts connected through multiple interfaces to multiple networks. Prior to DCE 1.2.2, DFS required all clients and servers to be reachable via all network interfaces.

- Backup performance enhancements

  A number of performance bottlenecks have been eliminated in the behavior of DFS backups, dumps, and restores:

  — A ‘‘bulk update’’ operation has been added, allowing the master sync site to send sets of updates to each secondary server in a single RPC.

  — The master sync site now updates secondary servers asynchronously, and works on the subsequent batch of changes in parallel with the RPC calls being performed.

  — Database transactions are no longer committed asynchronously, so that all servers now do commit processing in parallel.

  — A new RPC operation has been added so that multiple volumes can be added to the backup datase in a single RPC call.

- Support for 64-bit filesystems

  DCE 1.2.2 supports very large files and filesystems, while maintaining interoperability with the current widespread 32-bit machines.

- Support for use of protected RPC

  DCE 1.2.2 allows administrators to specify a range of DCE protections that can be used for most client-server communication. All architectural uses of unauthenticated RPCs have been eliminated. Prior to DCE 1.2.2, DFS used a range of DCE protection levels on its RPC operations, ranging from unauthenticated for peer **repserver** processes to packet-integrity or higher for the management clients.

  The new administrative controls allow administrators to distinguish between same-cell communication from inter-cell communication, so that a DFS Cache Manager will use one set of protection rules for intra-cell use (presumably protected behind a network firewall), while using another set for data-sharing outside the cell. Command line arguments and management clients allow administrators to achieve the right balance of protection and computational overhead.

A collection of bug reports (CRs) on all currently-known DCE defects, including those fixed since DCE 1.2.1, is located at:

dce1.2.2-root-dir/**project/defect.summary**

2.3.12.1 DFS Testing

A summary of DCE 1.2.2 DFS test results can be found in the

dce1.2.2-root-dir/**project/TestSummary**

file. A report on DFS functional, system, and CHO test results for DCE 1.2.2 can be found in:

dce1.2.2-root-dir/**project/test.results**

The procedures for running the other DFS functional tests are documented in Chapter 10 of the *OSF DCE Testing Guide*.

2.3.13 State of DCE System Testing

A summary of DCE 1.2.2 system test results can be found in:

dce1.2.2-root-dir/**project/test.results**

The procedures for running the DCE system tests are documented in Chapters 11-13 of the *OSF DCE Testing Guide*.

Chapter 3. OSF DCE 1.2.2 Directory Structure

It is assumed that the DCE 1.2.2 archives are found at the following pathname on your system:

your-root-dir/**dce/V1.2.2**

—where *your-root-dir* is the location on your disk where you installed the archives, as described in Chapter 4 of this document. (Refer to the section ‘‘Contents of the DCE 1.2.2 Release Area’’ in Chapter 1 of this document for more information.) This entire pathname is abbreviated as

dce1.2.2-root-dir

in the other chapters of this document.

Immediately beneath this pathname will be found several directories corresponding to the separate archives extracted from the DCE 1.2.2 distribution media, as described in Chapter 4. For example, the directory

dce1.2.2-root-dir/**dce**

will contain the extracted **dce** archive, which consists of the DCE 1.2.2 source tree; and the directory

dce1.2.2-root-dir/**doc**

will contain the extracted **doc** archive, which consists of the DCE 1.2.2 documentation set (further described in Chapters 5 and 6). For the sake of additional brevity and clarity, this **doc** archive pathname is abbreviated as

dce1.2.2-doc-dir

in Chapters 5 and 6. For example, the pathname

dce1.2.2-doc-dir/**src/dce_books_sgml/app_gd_style**

is equivalent to:

your-root-dir/**dce/V1.2.1/doc/src/dce_books_sgml/app_gd_style**

For pathnames outside the **doc** tree, the

dce1.2.2-root-dir

notation is used. For example, the pathname

dce1.2.2-root-dir/**ode/doc**

is equivalent to:

your-root-dir/**dce/V1.2.2/ode/doc**

3.1 Directory Structure Diagrams

The remainder of this appendix consists of graphical depictions of the directory structure for the following archives:

- *dce1.2.2-root-dir*/**dce**
- *dce1.2.2-root-dir*/**doc**
- *dce1.2.2-root-dir*/**project**
- *dce1.2.2-root-dir*/**ode**

See Chapter 4 for instructions on loading the DCE 1.2.2 archives.

A graphical depiction of the directory structure for the Distributed File-Access (DFA) component can be found in Chapter 7.

3.1.1 The dce Archive Tree

The extracted **dce** Archive tree, which contains source code for the DCE components and test programs, as well as other utilities, has the following structure:

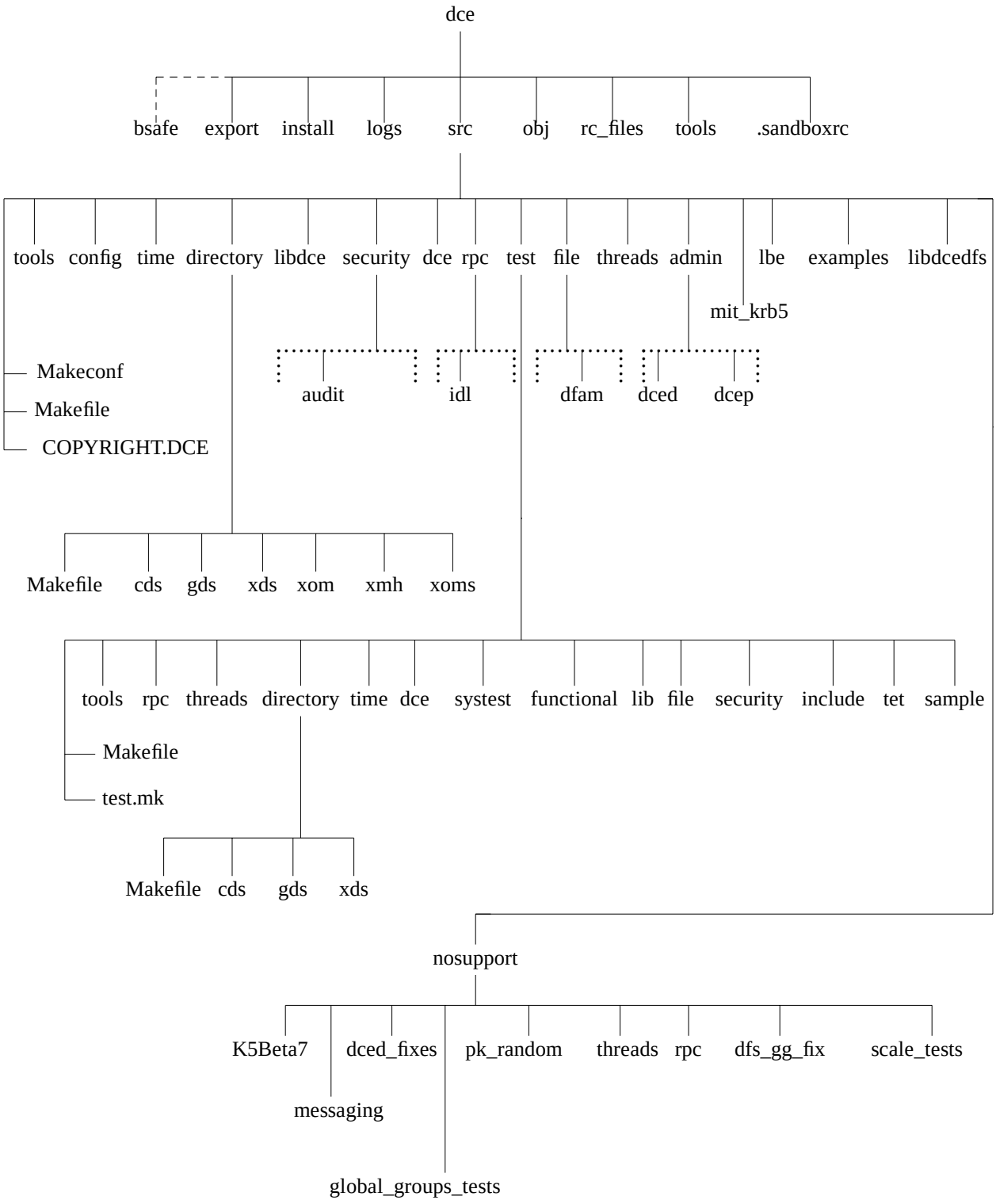


Figure 3-1. dce Tree Structure

Note that the dashed lines showing the position of **bsafe** are meant to indicate that this directory is present only in builds that include the **bsafe** source.

Dotted lines mean that only a portion of the indicated subdirectory structure is depicted in the diagram. See Chapter 7 for a detailed diagram of the contents of the DFA source

subdirectory at:

dce1.2.2-root-dir/**dce/src/file/dfam**

3.1.2 The doc Archive Tree

The extracted **doc** Archive tree, which contains the source and output files for the DCE documentation, has the following structure:

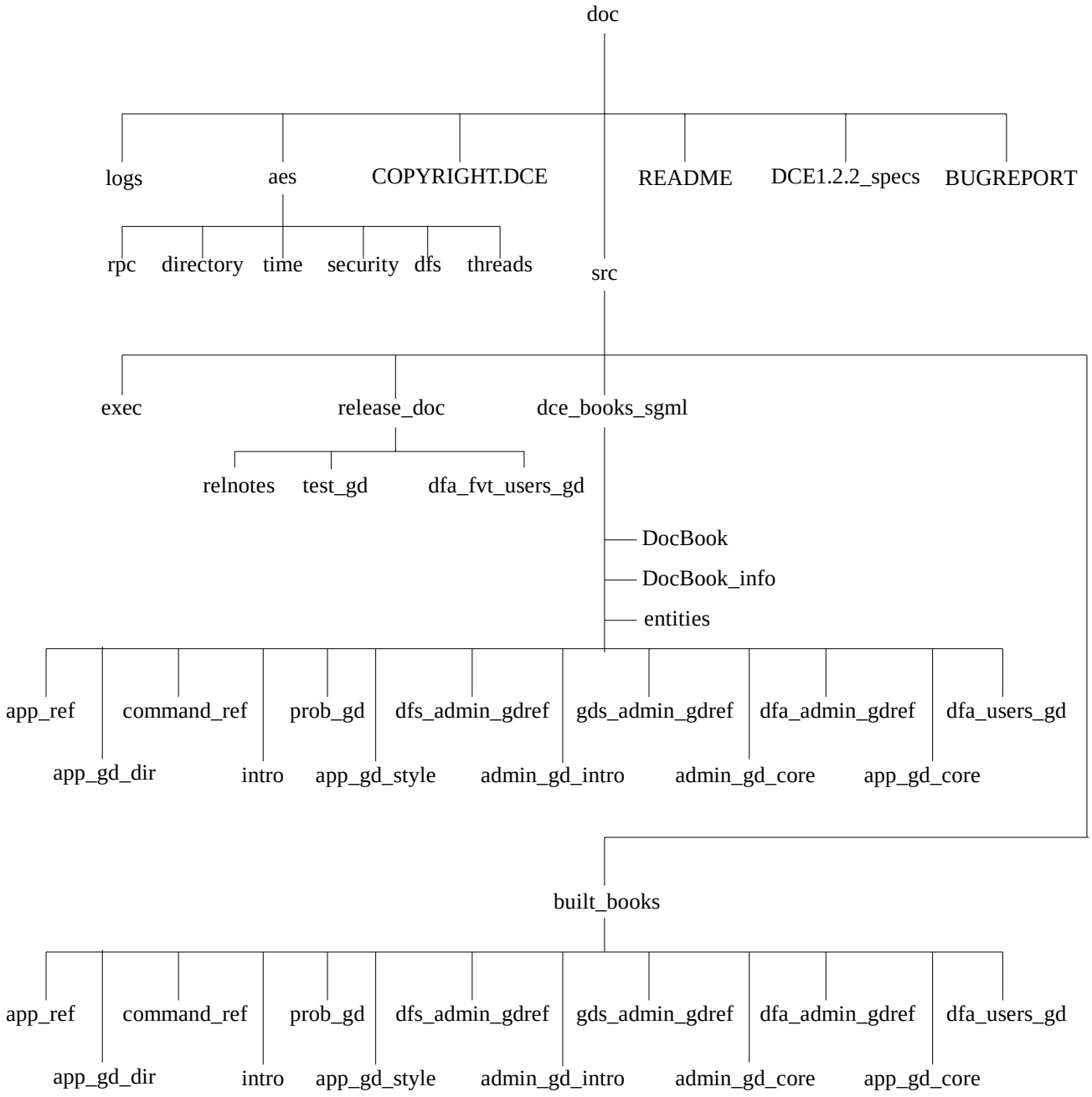


Figure 3-2. doc Tree Structure

3.1.3 The project Archive Tree

The extracted **project** Archive tree has the following structure:

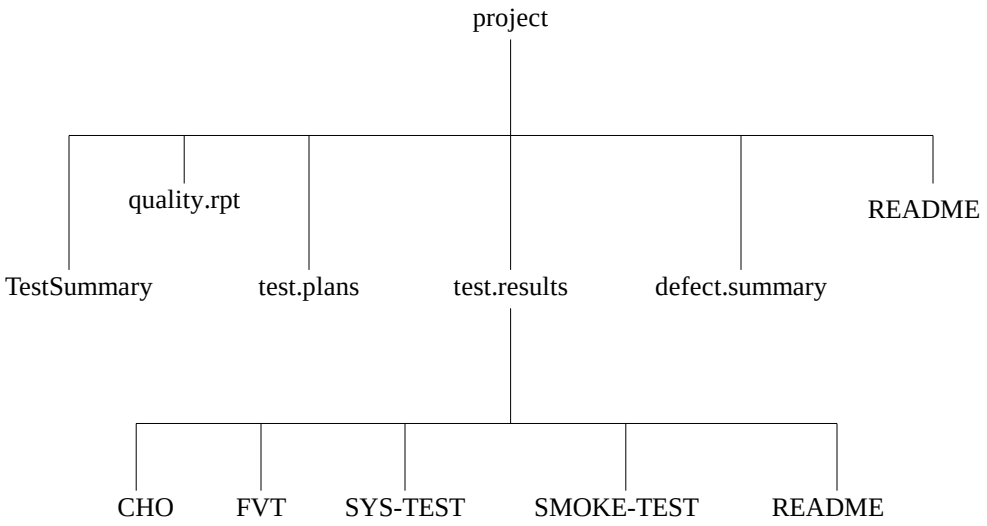


Figure 3-3. project Tree Structure

3.1.4 The ode Archive Tree

The extracted **ode** Archive tree has the following structure:

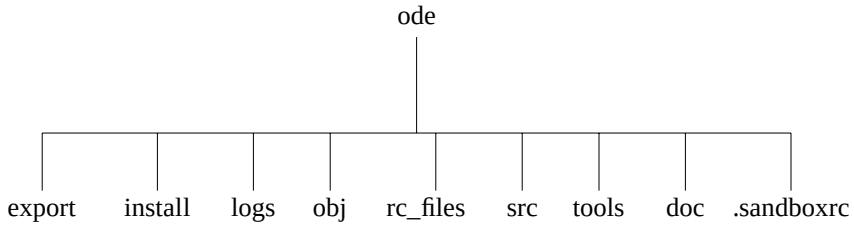


Figure 3-4. ode Tree Structure

Chapter 4. Building and Installing DCE 1.2.2

This chapter consists of sections that describe how to extract the archives from tape and CD-ROM, and how to build and install DCE 1.2.2 on the DCE 1.2.2 reference platform, i.e.: an IBM RISC System/6000 running AIX version 3.2.5 (revision 3.2.5.E4.R9 or a later revision; see ‘‘DCE 1.2.2 Reference Platform’’ below).

Note: Information for other platforms, which were reference platforms for DCE 1.1, is not included in this document. The original DCE 1.1 *Release Notes* should be consulted for information on the procedures for those platforms, along with Release Notes from the various vendors.

Throughout this chapter, it is assumed that the DCE 1.2.2 archives are being unloaded into

your-root-dir/**dce/V1.2.2**

where *your-root-dir* is the location on your disk where you plan to install and/or build the archive. This pathname is also abbreviated as

dce1.2.2-root-dir

4.1 DCE 1.2.2 Reference Platform

The DCE 1.2.2 reference platform is the IBM RISC System/6000 running the AIX 3.2.5 operating system (revision 3.2.5.E4.R9 or a later revision). However, before you execute the DCE code on AIX 3.2.5, you must set the **MALLOCTYPE** environment variable to a value of ‘‘3.1’’.

For example, in a Korn shell, you should enter:

```
export MALLOCTYPE=3.1
```

In a Bourne shell, you should enter the following:

```
MALLOCTYPE=3.1;export
```

In a C shell, you should enter the following:

```
setenv MALLOCTYPE 3.1
```

before you start DCE.

This step is necessary because **malloc()**'s behavior changed in AIX 3.2 as compared to that of AIX 3.1. The AIX 3.1 version of **malloc()** returns a fresh, zero-initialized space, and DCE depends on this behavior. The AIX 3.2 **malloc()**, on the other hand, is not guaranteed to return a zero-initialized space.

Setting **MALLOCTYPE** will cause all applications to use the ''old'' (vintage AIX 3.1) **malloc()**. Setting **MALLOCTYPE** to any other value will cause the new (AIX 3.2) **malloc()** to be used.

Bug report 12924 in the DCE defects database, located in the

dce1.2.2-root-dir/**project/defect.summary**

directory, contains additional information about this condition.

4.2 Overview of the DCE 1.2.2 Archives

The DCE 1.2.2 distribution contains several archives, which are listed with their approximate sizes in the following table. The sizes mentioned in this section are expressed as quantities which have been rounded upward. You may notice slight differences between the sizes given here and the sizes as measured on your system.

DCE 1.2.2 consists of the following archives: **ode**, **dce**, **doc**, and **project**.

There are three methods of obtaining DCE 1.2.2: QIC-150 (6150) tape, 8mm tape, and CD-ROM. The archives are listed below in the order they appear on the different media (top level directories only are shown):

QIC-150 (6150)	**dce**, **doc**, **project**, **ode**
8mm	**dce**, **doc**, **project**, **ode**
CD-ROM	**dce**, **doc**, **project**, **ode**

4.2.1 tar Archives on the Media

Note: Archives with a **.Z** extension are compressed.

Table 4-1 gives the space requirements for the archives.

TABLE 4-1. tar Archives on the Distribution Tapes

Archive Name	Contents	Approximate Size in Megabytes (expanded)	Approximate Size in Megabytes (compressed)
dce	DCE and Test Source	330.7	61.8
doc	Documentation	206.9	62.6
project	Release information	69.0	19.2
ode	ODE Tools	5.38	1.87
Total		611.98	145.47

A total of approximately **612** megabytes of free disk space is thus required to extract all four **tar** archives. The archives are in compressed format on the QIC-150 tape.

Briefly, the archives have the following contents (fuller descriptions can be found in the section ‘‘Contents of the DCE 1.2.2 Tapes’’ in Chapter 1 of this document):

- The **dce** archive contains the entire DCE source and test tree, buildable on the DCE 1.2.2 reference platform.
- The **doc** archive contains the DCE documentation sources, the PostScript and ASCII output, OSF DCE RFCs, DCE specifications, DCE documentation specifications, and AES documentation.
- The **project** archive contains miscellaneous release information, including a defect summary, quality report, and test results.
- The **ode** archive contains source code for the OSF Development Environment. For documentation on the OSF Development Environment tools in the **ode** archive, see the contents of the

  your-root-dir/**dce/V1.2.2/ode/doc**

  directory.

4.3 Extracting the Archives

The following subsections describe how to extract the contents of DCE 1.2.2 from its three distribution media: QIC-150 (6150) tape, 8mm tape, and CD-ROM.

4.3.1 Extracting the Archives from Tape

This section contains instructions for mounting the tapes, extracting the files from the distribution media, checking the available disk space, and creating the OSF/DCE directory tree.

4.3.1.1 Notes and Warnings about tar and tctl

This section contains information about the **tar** and **tctl** commands. If you are familiar with these, skip this section and proceed to Section 4.4.1.2 (‘‘Steps for Extraction’’).

- The **tar** (tape archiver) command reads from the current position of the tape. You should specify a nonrewinding tape device (called *nonrewinding-tape-device* in the instructions) to the command. This will ensure that each time the **tar** command closes the tape device, the tape will be left correctly positioned for reading the next image. If you do not specify a nonrewinding device, the tape will rewind each time the device is closed, and the first **tar** image will be read repeatedly at each successive reinvocation of the command. See your system’s **mtio(4)** reference page or other tape drive documentation for more information.
- If you get a message from **tar** indicating that the blocksize is equal to 0 (zero), then the **tar** command is not skipping the correct number of tape marks between archives and you must reinvoke **tar** to extract the next archive.
- The instructions given later in this chapter for extracting from the tapes describe how to extract all archives using the **tar** command. If you wish to not extract a particular archive, you can skip over it to get to the archive you do wish to extract.

  To skip an archive, enter the **tctl** or **dd** command to advance the tape to the next archive. See your system manuals for complete information on these commands.

  To access any archive directly, you can do either of the following:

  — Use the **tctl** command to position the tape. For example, the command

  tctl -f /dev/*nonrewinding-tape-device* **fsf 2**

  will move forward two **tar** archives. If the tape was positioned at the first archive before the **tctl** command was executed, the next **tar** command (after **tctl** is invoked) will read the third archive on the tape. Then, to return to the beginning of the tape, enter:

  tctl -f /dev/*rewinding-tape-device* **rewind**

  — Alternatively, you can invoke the **dd** command as many times as desired to move successively to the next archive on the distribution tape until the desired archive is reached:

  dd if=/dev/*nonrewinding-tape-device* **of=/dev/null**

4.3.1.2 Steps for Extracting from QIC-150 Tape

Follow these steps to extract from the QIC-150 (6150) tape:

1. Make sure there is enough disk storage space available to store the archives you plan to extract from the tape. Use the **df** command (or whatever command is appropriate for your operating system) to determine the amount of available space. Compare this size with the total size in Table 4-1.

2. If it does not already exist, create the

   your-root-dir/**dce/V1.2.2**

   and

   your-root-dir/**dce_archives**

   directories and **cd** to *your-root-dir*/**dce_archives**. This is the directory in which the DCE 1.2.2 **tar** archives are installed.

   mkdir -p *your-root-dir*/**dce_archives**
   mkdir -p *your-root-dir*/**dce/V1.2.2**
   cd *your-root-dir*/**dce/dce_archives**

3. Mount the tape using the UNIX **mt** or the AIX **tctl** command.

4. Extract the compressed archives from the tape:

   tar -xvf /dev/*tape-device*

5. Uncompress the archive:

   uncompress *.Z

6. You can generate a list of each archive's contents by entering the following **tar** command for each archive, substituting a different *output-file* name for each archive:

   tar -tvf /dev/*nonrewinding-tape-device* **>** *output-file*

   (After doing this, you should rewind the tape before proceeding with the extraction.)

   The following instructions for extracting the archives from the distribution tape will also create a list of the files in each archive.

7. Next, **cd** to *your-root-dir*/**dce/V1.2.2**:

   cd *your-root-dir*/**dce/V1.2.2**

8. To extract the **dce** archive, which contains source code for DCE, enter the following command:

   tar -xvf *your-root-dir*/**dce_archives/dce.tar > dce.tarlog 2>&1**

9. To extract the **doc** archive, which contains the documentation, enter the following command (in a Bourne or Korn shell):

   tar -xvf *your-root-dir*/**dce_archives/doc.tar > doc.tarlog 2>&1**

10. To extract the **project** archive, which contains release information for DCE, enter the following command:

    tar -xvf *your-root-dir*/**dce_archives/project.tar > project.tarlog 2>&1**

11. To extract the **ode** archive, which contains code for the OSF Development Environment, enter the following command:

    tar -xvf *your-root-dir*/**dce_archives/ode.tar > ode.tarlog 2>&1**

4.3.1.3 Steps for Extracting from 8mm Tape

Follow these steps to extract from the 8mm tape:

1. Make sure there is enough disk storage space available to store the archives you plan to extract from the tape. Use the **df** command (or whatever command is appropriate for your operating system) to determine the amount of available space. Compare this size with the total size in Table 4-1.
2. If it does not already exist, create the

   your-root-dir/**dce/V1.2.2**

   directory and **cd** to it. This is the directory in which the DCE 1.2.2 **tar** archives are installed.

   mkdir -p *your-root-dir*/**dce/V1.2.2**
   cd *your-root-dir*/**dce/V1.2.2**

3. Mount the tape using the UNIX **mt** or the AIX **tctl** command.
4. In order to conserve paper, we have not supplied a hardcopy list of the files in each distribution archive. You can generate a list of each archive's contents by entering the following **tar** command for each archive, substituting a different *output-file* name for each archive:

   tar -tvf /dev/*nonrewinding-tape-device* **>** *output-file*

   (After doing this, you should rewind the tape before proceeding with the extraction.)

   The following instructions for extracting the archives from the distribution tape will also create a list of the files in each archive.
5. To extract the **dce** archive, which contains source code for DCE, enter the following command:

   tar -xvf /dev/*nonrewinding-tape-device* **> dce.tarlog 2>&1**

6. To extract the **doc** archive, which contains the documentation, enter the following command (in a Bourne or Korn shell):

tar -xvf /dev/*nonrewinding-tape-device* **> doc.tarlog 2>&1**

7. To extract the **ode** archive, which contains code for the OSF Development Environment, enter the following command:

   tar -xvf /dev/*nonrewinding-tape-device* **> ode.tarlog 2>&1**

8. To extract the **project** archive, which contains release information for DCE, enter the following command:

   tar -xvf /dev/*nonrewinding-tape-device* **> project.tarlog 2>&1**

4.3.2 Extracting the Archives from CD-ROM

In order to successfully extract the DCE 1.2.2 archives, the system you use must have the following:

- Standard UNIX development tools, including **awk**, **lex**, and **yacc**.
- An ANSI C (X3.159-1989) compliant compiler.
- A non-rewinding tape device.
- Access to a tape drive or CD-ROM drive capable of reading one of the following distribution media:
  - QIC-150 (6150) cartridge tape
  - 8mm cartridge tape
  - CD-ROM
- A program capable of extracting files from UNIX **tar** archives on a non-rewinding tape device (**tar** on most systems).
- Sufficient disk space to copy the files from the tapes and sufficient disk space for building (see Tables 4-1, 4-2, and 4-3).

The CD-ROM containing DCE 1.2.2 is in the Rock Ridge format. The files are stored in a Unix directory hierarchy.

4.3.2.1 Mounting the CD-ROM

Before you begin, make sure that you have a local mount point on your system. You are free to use any local mount point; in the following instructions, **/cdrom** is used in the examples. To create the local mount point, do the following:

mkdir -p *cdrom_mount_point*

where *cdrom_mount_point* is the name you have chosen for your local mount point. For example:

mkdir -p /cdrom

Mount the CD-ROM using the UNIX **mount** command. On AIX, the command is:

mount -v 'cdrfs' -r'' /dev/*devicename cdrom_mount_point*

For example, if the CD-ROM is on the device **/dev/cd0**, the command will be:

mount -v 'cdrfs' -r'' /dev/cd0 /cdrom

4.3.2.2 Steps to Extract the Contents of the CD-ROM

Because the CD-ROM is a read-only device, you cannot perform operations such as write or build in the mounted area *cdrom_mount_point*. However, you should be able to use any UNIX commands (such as **ls**, **cd**, and **lpr**) to view and print its contents.

As described in Section 4.3, the CD-ROM contains the following top level directories: **doc**, **ode**, **dce**, and **project**.

Perform the following steps to extract the contents of the CD-ROM:

1. First, make sure you are in your working area, which we will call:

   your-root-dir/**dce/V1.2.2**

   Enter the following commands:

   mkdir -p *your-root-dir*/**dce/V1.2.2**
   cd *your-root-dir*/**dce/V1.2.2**

2. Copy the subtrees.

   To copy the **doc** tree, which contains the documentation, enter the following command (in a Bourne or Korn shell):

   (cd *cdrom_mount_point***; tar -cf - doc) | tar -xvf > doc.tarlog 2>&1**

   For example:

   (cd /cdrom; tar -cf - doc) | tar -xvf > doc.tarlog 2>&1

   To copy the **ode** tree, which contains code for the OSF Development Environment, enter the following command:

   (cd *cdrom_mount_point***; tar -cf - ode) | tar -xvf > ode.tarlog 2>&1**

For example:

(cd /cdrom; tar -cf - ode) | tar -xvf > ode.tarlog 2>&1

To copy the **dce** tree, which contains source code for DCE, enter the following command:

(cd *cdrom_mount_point***; tar -cf - dce) | tar -xvf > dce.tarlog 2>&1**

For example:

(cd /cdrom; tar -cf - dce) | tar -xvf > dce.tarlog 2>&1

To copy the **project** tree, which contains release information for DCE, enter the following command:

(cd *cdrom_mount_point***; tar -cf - project) | tar -xvf > project.tarlog 2>&1**

For example:

(cd /cdrom; tar -cf - project) | tar -xvf > project.tarlog 2>&1

4.4 General Prerequisites for Building and Testing

The object build sizes mentioned in this section are expressed as quantities found to be accurate for DCE 1.2.2 and rounded upward. You may notice slight differences between the sizes given here and the sizes as measured on your system.

To load the DCE 1.2.2 source archive (**dce**), you will need approximately 150 megabytes of disk space. To build all of the DCE 1.2.2 programs and tools, you will need approximately 616 MB of disk space. To build ODE in addition to DCE, you will need an additional 39 MB of disk space.

The following tables show the directories that are created and populated after a complete DCE (and ODE) build. The contents of the **ode** archive are necessary to build ODE, which is then used to build DCE, for which the contents of the **dce** archive are necessary.

TABLE 4-2. Directory Sizes (in MB) for Extracting and Building the ODE Archive

Directory	AIX
src	7
export	1
obj	8
install	2
Total	18

TABLE 4-3. Directory Sizes (in MB) for Extracting and Building the DCE Archive

Directory	AIX
src	80
export	5
obj	309
install	19
tools	4
Totals	417

Loading and building the contents of other archives requires additional free disk space.

Note that the sizes given in the above tables were obtained with the **du** command.

Note that in order to build and run DCE 1.2.2 successfully, every machine involved in the build must be running both the **lockd** and **statd** daemons. This is because file locking is performed during the GDS build by **gdsmkiss**, using **lockd**. It is not sufficient to have **lockd** running on the local machine: if the file being locked is located on an NFS-mounted filesystem, the local **lockd** will attempt to contact the remote **lockd** and **statd** during the lock operation, and if the remote machine is not running either daemon, the local process will hang in kernel mode.

4.4.1 Prerequisites for AIX 3.2.5.E4.R9

This section contains important information about building and configuring DCE 1.2.2 on AIX 3.2.5.E4.R9.

1. The following settings are necessary in order to build a **libdce** on AIX 3.2.5.E4.R9:

   ulimit -c unlimited
   ulimit -t unlimited
   ulimit -d unlimited
   ulimit -s unlimited
   ulimit -m unlimited

   Note that these commands are given in **ksh** syntax. Note also that merging options, as in (for example):

   ulimit -tdsm unlimited

   will not work.

2. You can enable the generation of core files by setting the **core** parameter to an appropriately large value (for example, equal to the value of **fsize**):

   chuser core=2097151 *username*

   You must execute this command as root. You can find the current value of **core** by doing (also as root):

   lsuser *username*

3. Make sure the compiler, **xlc**, has the version ‘‘xlc 1.3 ..’’ Use

   what /usr/bin/xlc

   to find out if it does.

The following error message will be observed when building ODE and DCE on AIX:

```
1506-193: (E) Function call argument cannot be assigned to corresponding parameter.
```

This message is a warning which is incorrectly reported as an error, and it can safely be ignored.

4.4.1.1 Note for Building the DCE 1.2.2 Tests on AIX

At least two versions of **ncform** have been observed to be present on the AIX platform. If the version of **ncform** present on your machine defines **yyunput** to be **void**, you will observe errors as

your-root-dir/**dce/V1.2.2/dce/src/test/security/api/testsh/tsh.l**

is being compiled. The errors are as follows:

```
"lex.yy.c", line 907.6: 1506-132 (S) Function yyunput cannot be redeclared.
"lex.yy.c", line 1364.6: 1506-030 (S) Identifier yyunput cannot be redeclared.
```

If this happens, you must edit **tsh.l** and define **yyunput** to be **void**, then restart the test build.

4.5 Building DCE and the Tools

This section contains step-by-step procedures for building the **ode** and **dce** binaries from the DCE 1.2.2 archives. For information on building the **doc** archive, see Chapter 6, ‘‘Building the DCE Documentation’’. Note that it is not necessary to build the DCE 1.2.2 documentation, since it is supplied in the archive in formatted PostScript and formatted ASCII forms along with its source.

For information on how to build parts of DCE *without* one or more components, see Chapter 12 of the DCE 1.1 *OSF DCE Porting and Testing Guide*.

4.5.1 Building an Exportable Version of DCE

The symbol **INTL_BUILD** in the

dce1.2.2-root-dir/**dce/src/Makeconf**

determines whether or not an exportable version of DCE is built. If **INTL_BUILD** is defined, an ‘‘international’’ (exportable) version of DCE is built. For details, refer to the section ‘‘Building an Exportable Version of DCE’’ in Chapter 2.

Note: It is the licensee’s responsibility to determine if the version of DCE that they have built is indeed exportable, based on current U.S. export regulations. Effort has been made in the DCE source to facilitate generating an exportable version if such is desired, but it is not possible to guarantee under all circumstances, without testing, that what is built will be exportable.

4.5.2 Building BSAFE

If you wish to use the new DCE 1.2.2 public key functionality (see ‘‘Public Key Support’’ in ‘‘Distributed Computing Environment’’ in Chapter 2) you must build DCE 1.2.2 with the RSA ‘‘BSAFE’’ encryption facility. If you do not wish to build the public

key functionality into DCE 1.2.2, you should skip the remainder of this section and go directly to ‘‘Building the ode Archive’’ below.

BSAFE can be licensed directly from RSA or from OSF. See the DCE price sheet and licensing agreement for further information.

In the following section, it is assumed that you have unloaded the BSAFE sources into

bsafe-root-dir

where *bsafe-root-dir* is the location on your system where you plan to build BSAFE.

To build BSAFE, after you have extracted the sources from the distribution media, do the following:

1. Make a copy of the **sun4** (or **solaris**) directory, located under **bsafe**, and rename it to **rios** (i.e., the platform name appropriate for your platform).
2. **cd** to the renamed directory and make whatever platform-specific edits you find necessary to the Makefile there.

   You may find that the only necessary Makefile change for your platform is to set the compiler variable as appropriate, for example:

   CC=xlc

   for the AIX platform.

After you have done the above, you should be able to successfully build the BSAFE code. The result will be the creation of the library **libbsafe2.a** in the directory.

4.5.2.1 Including the Personal Security Module (PSM) in the DCE Build

After you have built BSAFE, do the following:

1. Create the

   dce1.2.2-root-dir/**dce/external**

   directory (outside of **dce/src**) if you have not done so yet.
2. Create the following two directories:

   dce/external/*target_platform***/usr/include**
   dce/external/*target_platform***/usr/lib**

   where *target_platform* is a name representing the platform on which you are building DCE. (For example, *target_platform* might be **HP800** on an HP-UX platform, or **RIOS** on the AIX reference platform.)
3. Create a link from your BSAFE include directory to

   dce/external/*target_platform***/usr/include/bsafe**
4. Create a link from your BSAFE library (**libbsafe2.a**) to

dce/external/*target_platform***/usr/lib/libbsafe2.a**

In the

dce1.2.2-root-dir/**dce/src/dce/Buildconf**

file, there is a **BSAFE_ROOT** variable, which is defined as

${sandbox_base}/external/${target_platform}

You should define **sandbox_base** and **target_platform** as appropriate for your platform and system.

The existence of a ‘‘functioning’’ or ‘‘non-functioning’’ Personal Security Module (PSM) in DCE 1.2.2 is what determines whether the public key functionality is or is not present in the built binaries. The presence of a non-functioning PSM layer will simply result in calls to public key functionality becoming no-ops.

Inclusion or non-inclusion of the PSM in DCE 1.2.2 is done through the same framework by which ‘‘domestic’’ and ‘‘exportable’’ versions of DCE are distinguished (see ‘‘Building an Exportable Version of DCE’’ above). Exportable DCE can be built only with a non-functioning PSM (and thus with no public key functionality). Domestic DCE can be set up to be built either with *or* without a functioning PSM. The remainder of this section describes the details of specifying the kind of domestic build desired.

In

dce1.2.2-root-dir/**dce/src/libdce/Makefile**

the **BSAFELIB** variable is defined as:

```
${BSAFE_ROOT}/usr/lib/libbsafe2.a
```

In

dce1.2.2-root-dir/**dce/src/security/psm/Makefile**

the include directory is defined as follows:

```
.if defined (USE_DES)
BSAFE_DIR = domestic
INCFLAGS =  -Idomestic -I../h -I${BSAFE_ROOT}/usr/include
.else
BSAFE_DIR = international
INCFLAGS =  -Iinternational -I../h
.endif
```

Located under

dce1.2.2-root-dir/**dce/src/security/psm**

there is a **domestic** and **international** directory. The **domestic** directory contains source to build a fully functional PSM. The **international** directory contains source to build a non-functioning PSM.

The DCE 1.2.2 ‘‘domestic’’ source comprises both the **domestic** and **international** directories. The DCE 1.2.2 ‘‘international’’ (i.e., exportable) source is made up of the contents of the **international** directory only. By default,

dce1.2.2-root-dir/**dce/src/Makeconf**

has **USE_DES=1** which will cause the contents of the **domestic** directory (and hence a functioning PSM) to be built. In order to change this and instead build a non-functioning PSM, you should set **USE_DES=0** in:

dce1.2.2-root-dir/**dce/src/Makeconf**

4.5.3 Building the ode Archive

This section contains a step-by-step procedure for building the **ode** tools which are needed to build DCE 1.2.2. The instructions in this section are based on Chapter 3 of the *ODE System Administration Guide*, ''Building and Installing the ODE Tools''.

Note that the following instructions are not sufficient to build the full ODE source control environment. If you wish to build the ODE source control environment, including RCS, or if you want further details about ODE, you should consult the *ODE System Administration Guide* at:

dce1.2.2-root-dir/**ode/doc/sag.ps**

and the *ODE User's Guide* at:

dce1.2.2-root-dir/**ode/doc/dug.ps**

and its appendix, located at:

dce1.2.2-root-dir/**ode/doc/dug.appendixA.ps**

4.5.3.1 ODE 2.3

Version 2.3 of ODE is included with DCE 1.2.2. This ODE includes a parallel version of **make**; the resulting parallel DCE build is much faster than with the single-process version. This newer **make** also includes other performance enhancements and many bug fixes.

Also, the **.LINKS** target was added in ODE 2.3. This allows targets that are symbolic links to be so indicated, so that they can be tested with **lstat( )** instead of **stat( )**. Because of this, all other stats of files are now done with **stat( )** instead of **lstat( )**. This means that the date of the file a link points to is used instead of the date of the link. This is useful when a sandbox has many links as a result of **mklinks** having been run.

Note that the **shared** and **local** files are no longer used; ODE 2.3 uses a **Buildconf** and **Buildconf.exp** file. For more information, see Section 10.1.5.1 of Chapter 12 of the DCE 1.1 *OSF DCE Porting and Testing Guide*.

4.5.3.2 The ODE Archive Sandbox

The ODE archive is in the form of an ODE ‘‘sandbox’’, or build area. The name of this ODE sandbox is **ode**. The sandbox has the following structure:

Figure 4-1. ODE Sandbox Structure

The ODE distributed with DCE 1.2.2 includes a fix to resolve memory usage problems. Make sure to build the ODE tools provided with DCE 1.2.2, even if you already have Version 2.3 of the tools.

The **obj**, **export**, and **tools** subdirectories are created when ODE is built.

The following instructions will walk you through the steps necessary to build the ODE tools from the ODE archive.

1. **cd** to the

   dce1.2.2-root-dir/**ode/src**

   directory.

   You will need the following environment variables set in order to use ODE: **HOME** and **USER**.

   HOME Should be set to the pathname of your home directory.

   USER Should be set to the username of whoever is doing the build.

2. Set the environment variable **NO_RCS**.

   If you wish to build only the ODE tools necessary for building DCE, then you do not need to build RCS. You can turn the RCS build off by asserting **NO_RCS**. For example:

   sh or **ksh**: **NO_RCS=NO_RCS; export NO_RCS**

in **csh**: **setenv NO_RCS**

If you do wish to go on to build the ODE source control tools (note that it is not necessary to do this in order to build DCE 1.2.2), refer to the *ODE System Administrator's Guide*.

3. **setup.sh** sets the **PATH** variable internally. The first element in the path is set to **/usr/local/bin**. This will cause a problem if you have a **/usr/local/bin/make** on your system. If your machine has a **/usr/local/bin/make** you will need to either remove it or edit **setup.sh** so that **/usr/local/bin** is not in the **PATH** variable.

4. The final step is to bootstrap the basic build tools by running **setup.sh** in:

   dce1.2.2-root-dir/**ode/src/ode/setup**

   This step will build the basic tools which will be used to build ODE. The tools built are: **build**, **make**, **md**, **makepath**, **genpath**, and **release**. These are placed in the

   dce1.2.2-root-dir/**ode/tools/***context*/**bin**

   directory, where *context* is **rios_aix**, a name made up of a combination of the hardware platform name (**rios**) and the name of its operating system (**aix**).

   Before executing **setup.sh**, make sure that you are in the

   dce1.2.2-root-dir/**ode/src**

   directory. Check whether the

   dce1.2.2-root-dir/**ode/logs/rios_aix**

   directory exists; if it does not, create it now.

   Execute the setup script in the following form (in a Bourne or Korn shell):

   ```
   sh -x ode/setup/setup.sh rios_aix  >  ../logs/rios_aix/setup.log 2>&1
   ```

   Note: You have now built all the ODE executables required for building DCE. If you intend to use ODE only for building DCE, you should go on to the next section. If you wish to build the full set of ODE tools, refer to the **ode** documentation, found in the **ode** archive (see Section 4.3.1 above), for instructions on how to do this.

   Verify that the build was error free by searching for the following strings in the

   dce1.2.2-root-dir/**ode/logs/***context*/**setup.log**

   output logfile: `Undefined`, `Undeclared`, `Errors`, `don't know how to make`, `Signal 10`, and `Signal 11`.

   Verify that the executables exist by looking in:

   dce1.2.2-root-dir/**ode/tools/rios_aix/bin**

4.5.4 Building the dce Archive

This section contains a step-by-step procedure for building the DCE source archive.

For disk requirements to build DCE, see Table 4-3.

4.5.4.1 Building and Installing DCE Distributed File-Access (DFA)

Instructions for building and installing the DCE Distributed File-Access manager (abbreviated DFA) client-side code can be found in Chapter 7. Instructions for building the DFA server will be found in the following section.

The **dce** archive is in the form of an ODE ‘‘sandbox’’, or build area. The name of this ODE sandbox is **dce**. The sandbox has the following structure:

Figure 4-2. DCE Sandbox Structure

dce
|
|
|---------- **.sandboxrc**
|
|---------- **install/rios/{opt,dcetest}/dce1.2.2**
|
|---------- **export/rios**
|
|---------- **obj/rios**
|
|---------- **tools/rios/bin**
|
|---------- **rc_files**
|
|---------- **logs/rios**
|
|---------- **src**

4.5.4.2 Building DCE 1.2.2

To begin these steps, first **cd** to the top level of the **dce** sandbox at:

dce1.2.2-root-dir/**dce**

Do the following:

1. Edit the **.sandboxrc** file, modifying the second line with the value of your

dce1.2.2-root-dir

as the path. For example, if your *dce1.2.2-root-dir* is **/u2**, the contents of your **.sandboxrc** will be:

```
default dce
base * /u2/dce/V1.2.2
sb dce
```

2. Use the **cd** command to move to the

   dce1.2.2-root-dir/**dce/src**

   directory.

3. Make sure you have the ODE 2.3 tools in the command search path, and that the path to that version of the tools precedes all others. For example, if the ODE tools are in the

   dce1.2.2-root-dir/**ode/tools/rios_aix/bin**

   directory, then make sure that this path is first in your **PATH**. In the Bourne or Korn shells, do the following:

   PATH=*dce1.2.2-root-dir*/**ode/tools/rios_aix/bin:$PATH**
   export PATH

   In the C shell, do the following:

   set path=*dce1.2.2-root-dir*/**ode/tools/rios_aix/bin: $path**

4. Make sure you have the environment variables **HOME**, **USER**, **DEFTOOLBASE**, and **TARGET_SYS** set to the following values:

   HOME=*your-home-directory*
   USER=*username-of-whoever-is-doing-the-build*
   DEFTOOLBASE=*dce1.2.2-root-dir*/**dce/tools/rios/bin/**
   TARGET_SYS=*target_platform*

   On the DCE 1.2.2 reference platform (AIX), **TARGET_SYS** should be set to **RIOS**.

5. Bootstrap the basic DCE tools.

   In this step the basic DCE tools are first built. The tools built are: **compile_et**, **idl**, **idl.cat**, **mavcod**, **mavros**, **sams**, **prs**, and **zic**. These are placed in:

   dce1.2.2-root-dir/**dce/tools/rios/bin**

   Bootstrap the tools by entering the **build** command in the following form (in a Bourne or Korn shell):

   cd *dce1.2.2-root-dir*/**dce/src**
   build -k -rc *dce1.2.2-root-dir*/**dce/.sandboxrc setup_all > ../logs/rios/setup.log 2>&1**

For example:

```
build -k -rc /u2/dce/V1.2.2/dce/.sandboxrc setup_all  >  ../logs/rios/setup.log 2>&1
```

Inspect the **setup.log** file for any failures. Verify that the build was error free by searching for the following strings: `Undefined`, `Undeclared`, `Errors`, `don't know how to make`, `Signal 10`, and `Signal 11`.

6. Build DCE.

   To begin the build:

   cd *dce1.2.2-root-dir*/**dce/src**
   build -k -rc *your-root-dir*/**dce/V1.2.2/dce/.sandboxrc build_all > ../logs/rios/build.log 2>&1**

   For example:

   ```
   build -k -rc /u2/dce/V1.2.2/dce/.sandboxrc build_all > ../logs/rios/build.log 2>&1
   ```

   Inspect the **build.log** file for any failures. Verify that the build was error free by searching for the following strings: `Undefined`, `Undeclared`, `Errors`, `don't know how to make`, `Signal 10`, and `Signal 11`. If all has gone well, proceed to the next step.

7. Create the **install** collection for DCE.

   This step creates the **install** area for DCE. The **install** area contains the executables and files which you will use to install and configure DCE on a system.

   This step must be run as root, but before switching user to root, **echo** your **PATH** to verify its current contents, because it may change after switching. After using **su** to change to root, **echo** your **PATH** again. If the contents of **PATH** are not the same as before, modify **PATH** accordingly. Then do the following:

   build -k -rc *your-root-dir*/**dce/V1.2.2/dce/.sandboxrc **
   TOSTAGE=*path_to_your_install_area* **install_all > **
   ../logs/rios/install.log 2>&1

   For example:

   ```
   build -k -rc /u2/dce/V1.2.2/dce/.sandboxrc TOSTAGE=/u2/my_builds/1.2.2 \
    install_all > ../logs/rios/install.log 2>&1
   ```

   Note: Do not remove any objects from

   dce1.2.2-root-dir/**dce/obj/rios/***

   if you are planning to build the tests; some of these files are needed for the test build.

   When you have completed this step, relinquish the **root** identity.

   Inspect the **install.log** file for any failures. Verify that the build was error free by searching for the following strings: `Undefined`, `Undeclared`, `Errors`, `don't know how to make`, `Signal 10`, and `Signal 11`. If all has gone well, proceed

to the next step.

8. Build the DFA server.

   To build the DFA server, change directory to the **dfam** directory in the **src** tree:

   cd *dce1.2.2-root-dir*/**dce/src/file/dfam**

   Then enter the following:

   build -k -rc *dce1.2.2-root-dir*/**dce/.sandboxrc build_all > ../../../logs/rios/dfa_server.log 2>&1**

   For example:

   build -k -rc /u2/dce/V1.2.2/dce/.sandboxrc build_all > ../../../logs/rios/dfa_server.log 2>&1

   Inspect the **test.log** file for any failures. If all has gone well, proceed to the next step.

9. Build the test suite.

   Note: Before attempting to build the tests, see Section 4.5.1.1, ‘‘Note for Building the DCE 1.2.2 Tests on AIX’’.

   Before building the tests, in the file

   dce1.2.2-root-dir/**dce/src/test/functional/security/lib/Makefile**

   replace the following lines:

```
.if ${TARGET_MACHINE} == "RIOS"
# If OBJECTDIR is a relative path
VPATH           = ${SOURCEBASE}/${OBJECTDIR}/admin/expect_dce
.else
# If OBJECTDIR is an absolute path
VPATH           = ${OBJECTDIR}/admin/expect_dce
.endif
```

   with the following line:

```
VPATH           = ${OBJECTDIR}/admin/expect_dce
```

   To build the test suite, change directory to the **test** directory in the **src** tree:

   cd *dce1.2.2-root-dir*/**dce/src/test**

   Then enter the following:

   build -k -rc *dce1.2.2-root-dir*/**dce/.sandboxrc build_all > ../../logs/rios/test.log 2>&1**

   For example:

   build -k -rc /u2/dce/V1.2.2/dce/.sandboxrc build_all > ../../logs/rios/test.log 2>&1

Inspect the **test.log** file for any failures. If all has gone well, proceed to the next step.

10. Create the test **install** collection for the test suite. This step creates the **install** area for the tests.

    This step must be run as root, but before switching user to root, **echo** your **PATH** to verify its current contents, because they may change after switching. After using **su** to change to root, **echo** your **PATH** again. If the contents of **PATH** are not the same as before, modify **PATH** accordingly. Then do the following:

    build -k -rc *dce1.2.2-root-dir*/**dce/.sandboxrc **
    TOSTAGE=*path_to_your_install_area* **install_all > **
    ../../logs/rios/test_install.log 2>&1

    For example:

    **build -k -rc /u2/dce/V1.2.2/dce/.sandboxrc TOSTAGE=/u2/my_builds/1.2.2 **
    install_all > ../../logs/rios/test_install.log 2>&1

    When you have completed the test install step, relinquish the root identity.

    Inspect the **test_install.log** file for any failures. If all has gone well, you have now successfully built DCE 1.2.2. Have fun.

4.5.4.3 Build Options for Improving DCE Performance

When building DCE 1.2.2 for the purpose of performance testing, you should do at least the following two things:

- Turn off DCE Threads debugging, by putting **-DNDEBUG** in the **CFLAGS** line in:

  dce1.2.2-root-dir/**dce/src/threads/Makefile**

- Turn off DCE RPC runtime debugging, by commenting out the **DEBUG_DEFS** line in:

  dce1.2.2-root-dir/**dce/src/rpc/runtime/Makefile**

4.6 Certification API Porting Information

The DCE Certification API is a new component in DCE 1.2.2. This component uses a C++ class library to parse ASN.1-encoded data (for example, X.509 certificates). This class library includes a platform-specific header file (**pkc_codesets.h**) that defines mappings between the local C language character set of the platform and three ISO-registered character sets. These mappings are employed when a user of the class library wishes character string data to be presented in ‘‘local’’ format.

pkc_codesets.h should be placed in a platform-specific subdirectory of

dce1.2.2-root-dir/**dce/src/security/pkc/h**

prior to building DCE. A version of this file suitable for use on the reference platform is supplied in

dce1.2.2-root-dir/**dce/src/security/pkc/h/RIOS/pkc_codesets.h**

A program is supplied as an aid in constructing a platform-specific version of **pkc_codesets.h**. **gen_codesets** is not built as part of a normal ODE build of DCE. Running the program will produce a platform-specific **pkc_codesets.h** which should be moved to

dce1.2.2-root-dir/**dce/src/security/pkc/h/***target_platform*/**pkc_codesets.h**

prior to building DCE.

4.6.1 Building and Running gen_codesets

The **gen_codesets** program is built from four files:

- **gen_codesets.cxx**

  Portable code, supplied in:

  dce1.2.2-root-dir/**dce/src/security/pkc/utils/**

- **ia5_mapping.h**

  Platform-specific code, an example of which can be found in:

  dce1.2.2-root-dir/**dce/src/security/pkc/h/RIOS/**

- **iso646_mapping.h**

  Platform-specific code, an example of which can be found in:

  dce1.2.2-root-dir/**dce/src/security/pkc/h/RIOS/**

- **t61_mapping.h**

  Platform-specific code, an example of which can be found in:

  dce1.2.2-root-dir/**dce/src/security/pkc/h/RIOS/**

Copy the four files listed above to a working directory.

Each of the three **XXX_mapping.h** header files defines a mapping from an ISO codeset to the local codeset of the platform. **ia5_mapping.h** defines the mapping from ISO International Alphabet 5, **iso646_mapping.h** defines the mapping from ISO-646, and **t61_mapping.h** defines the mapping from the CCITT T.61 codeset. Each header file contains a C array definition of the following form:

```
char XXX_to_local_array[256] = {
'\0',     // 0/0
'\0',     // 0/1
'\0',     // 0/2
```

```
...
};
```

Each entry of the array should contain the local character that corresponds to the character from the ISO codeset. The header files should be edited so that the arrays contain the closest matching local character for each ISO-specified character. Thus, for an ASCII platform, the array element labelled ‘‘4/1’’ within **ia5_mapping.h** should contain the character constant ’A’.

Character codes that do not appear within the ISO-registered codeset should be mapped to the C ’\0’ character. ISO characters that do not correspond to local characters should be mapped either to a similar local character, or to the C translated into the local character set). Aside from the ’\0’ code, no code should appear in more than one position of each array.

Note that the class library supplied with DCE 1.2.2 supports only octet-octet codeset mapping. Non-spacing characters from the T.61 codeset must be mapped into valid C characters, and alternate character set selection is not supported when using the local codeset.

When you have edited the three headers (**ia5_mapping.h**, **iso646_mapping.h** and **t61_mapping.h**), you should compile and run **gen_codesets.cxx** as follows:

1. To compile **gen_codesets**:

   % *your_CPlusPlusCompiler* **-o gen_codesets gen_codesets.cxx**

   where *your_CPlusPlusCompiler* is the appropriate C++ compiler for your platform (‘‘xlC++’’ on the DCE 1.2.2 reference platform). If the compiler does not automatically invoke the linker, then you must issue an appropriate **ld** command.

2. To run **gen_codesets**:

   % gen_codesets pkc_codesets.h

   The resulting file ‘‘**pkc_codesets.h**’’ should then be copied to:

   dce1.2.2-root-dir/**dce/src/security/pkc/h/***target_platform*/**pkc_codesets.h**

4.7 Building DCE 1.2.2 for Systems without C++ Runtime

Currently, the DCE 1.2.2 runtime requires the presence of a C++ runtime. This is because the PKSS client side is written in C++, and the PSM contains an array of entrypoint vectors (one for each keystore), one of which points to PKSS entrypoints. While C++ runtimes are not uncommon, this section documents a way of lifting the C++ restriction, at the cost of eliminating PKSS support.

To build DCE 1.2.2 for a system without C++ runtime, do the following.

a. In the file

dce1.2.2-root-dir/**dce/src/security/psm/domestic/sec_pvtkey_registered_keystores.c**

at (approximately) line 74, replace the following line:

```
#ifdef HPUX
```

with the following:

```
#if defined(HPUX) || defined(NO_PKSS)
```

b. In the file

dce1.2.2-root-dir/**dce/src/security/psm/Makefile**

at (approximately) line 315, after the following line:

```
CFLAGS  = -DOSF_DCE ${DCEPATHS} -D_BSD -DBSD ${${TARGET_MACHINE}_CFLAGS}
```

insert the following line:

```
CFLAGS += -DNO_PKSS
```

Chapter 5. Documentation Supplied in DCE 1.2.2

Throughout the remainder of this chapter, it is assumed that the DCE 1.2.2 archives are found at the following pathname on your system:

your-root-dir/**dce/V1.2.2**

—where *your-root-dir* is the location on your disk where you installed the archives, as described in Chapter 4 of this document. This pathname is abbreviated as

dce1.2.2-root-dir

in the other chapters of this document.

Immediately beneath this pathname will be found several directories corresponding to the separate archives extracted from the DCE 1.2.2 distribution media, as described in Chapter 4. For example, the directory

dce1.2.2-root-dir/**dce**

will contain the extracted **dce** archive, which consists of the DCE 1.2.2 source tree; and the directory

dce1.2.2-root-dir/**doc**

will contain the extracted **doc** archive, which consists of the DCE 1.2.2 documentation set (further described in Chapters 5 and 6). For the sake of additional brevity and clarity, this **doc** archive pathname will be abbreviated as

dce1.2.2-doc-dir

throughout the remainder of this chapter. For example, the pathname

dce1.2.2-doc-dir/**src/dce_books_sgml/app_gd_style**

is equivalent to:

your-root-dir/**dce/V1.2.2/doc/src/dce_books_sgml/app_gd_style**

For pathnames outside of the **doc** tree, the

dce1.2.2-root-dir

notation will be used.

5.1 Contents of the DCE 1.2.2 Documentation Set

The following books and documents make up the DCE 1.2.2 documentation set:

- *OSF DCE Version 1.2.2 Release Notes* -
(*dce1.2.2-doc-dir*/**src/release_doc/relnotes** directory).

  This book (which you are now reading) is written for programmers, writers, and system administrators at licensees' sites, and contains instructions for loading and building documentation and source directories from the DCE distribution tapes.

- *OSF DCE Testing Guide* -
(*dce1.2.2-doc-dir*/**src/release_doc/test_gd** directory).

  This book contains information on how to install and execute the DCE functional and system tests, and how to interpret test results.

  Note: The *OSF DCE Testing Guide* is an abridged version of the former *OSF DCE Porting and Testing Guide*, updated for DCE 1.2.2, in which only information about DCE testing has been retained from the original book.

- *OSF DCE Problem Determination Guide* -
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/prob_gd** directory).

  This book (which was new to the DCE documentation set in DCE 1.1) consists of a listing of all DCE error messages and status codes, along with explanations of the messages and (where appropriate) recovery actions.

- *Introduction to OSF DCE* -
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/intro** directory).

  This book is a comprehensive, descriptive overview of the DCE architecture and its underlying concepts. It also serves as an introduction to the other DCE manuals and contains a master glossary for the entire DCE documentation set.

- *OSF DCE Application Development Guide: Introduction and Style Guide* -
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/app_gd_style** directory).

  The *Style Guide* was new to the DCE documentation set in DCE 1.1. It attempts to discuss all of the major issues that are likely to confront most programmers at some stage in DCE application design and development.

- *OSF DCE Application Development Guide: Core Components* -
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/app_gd_core** directory).

  This volume contains information on how to use the Application Programming Interfaces (APIs) provided for the major DCE components.

- *OSF DCE Application Development Guide: Directory Services* -
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/app_gd_dir** directory).

  This volume describes the DCE Naming and Access to CDS through XDS, the use of the X/Open Directory Service Interface, and the Object Classification Tables and other information on XDS, X.500 Directory, and GDS objects.

- *OSF DCE Application Development Reference* -
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/app_ref** directory).

  This manual contains the reference pages for programmatic interfaces to the various DCE services.

- *OSF DCE Command Reference* -
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/command_ref** directory).

  This book is a reorganization of the reference pages of which the (now-defunct) *OSF DCE Administration Reference* originally consisted, and includes all non-DFS and non-GDS commands. Included also are commands originally in the (now-defunct) *OSF DCE User's Guide and Reference*.

- *OSF DCE Administration Guide* -

  This book is a task-oriented guide to the various tasks that an administrator must perform both at a local (node) and network level to establish and maintain a distributed computing environment. Both text and examples are used. There are two volumes, which have the following titles:

  - *OSF DCE Administration Guide — Introduction*
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/admin_gd_intro** directory).
  - *OSF DCE Administration Guide — Core Components*
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/admin_gd_core** directory).

- *OSF DCE DFS Administration Guide and Reference* -
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/dfs_admin_gdref** directory).

  This manual contains reference pages and guide material for the DCE DFS administrative commands, files, and daemons.

- *OSF DCE GDS Administration Guide and Reference* -
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/gds_admin_gref** directory).

  This manual contains reference pages and guide material for the DCE GDS administrative commands, files, and daemons.

- *OSF DCE File-Access Administration Guide and Reference* -
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/dfa_admin_gdref** directory.)

  This manual describes administration tasks for the Distributed File Access software.

- *OSF DCE File-Access User's Guide* -
(*dce1.2.2-doc-dir*/**src/dce_books_sgml/dfa_users_gd** directory.)

  This manual describes user tasks for the Distributed File Access software.

- *DFA FVT User's Guide* -
(*dce1.2.2-doc-dir*/**src/release_doc/dfa_fvt_users_gd** directory.)

  This manual describes how to verify that the Distributed File Access software has been ported and built correctly.

For information about the documentation available for the OSF Development Environment (ODE) in Archive 2, see Chapter 12 of the DCE 1.1 version of the *OSF DCE Porting and Testing Guide*. Other ODE documentation is located in the

dce1.2.2-root-dir/**ode/src/ode/doc**

directory.

5.2 Supplemental DCE Documentation Available Electronically

OSF is making available, in the Systems Engineering AFS cell, the following auxiliary documentation:

- The SML book sources, plus PostScript output generated from them
- The OSF Documentation Tools Environment (DTE), which is used to format SML source
- Supplementary documentation, including
  - The old *OSF DCE Porting and Testing Guide*
  - The old *OSF DCE DFS Application Development Guide and Reference*
  - The old DCE functional and design specifications

This material resides in the DCE AFS cell under

/afs/syseng.osf.org/dce-archive/dce1.2.2

For access information, licensees should contact

dce-support-admin@opengroup.org
phone +617 621 8990

5.2.1 SML Files

The DCE 1.2.2 documentation updates were made to the SML sources. These sources were then converted to SGML. All technical content that was updated or added for DCE1.2.2 resides in both the SML and SGML sources. However, a final editorial pass was made on the output from the SGML source, and these edits were only incorporated into the SGML sources. Therefore, minor (editorial) enhancements exist in the SGML source that are *not* in the SML.

5.2.2 Other Supplementary Books

The *OSF DCE Porting and Testing Guide* has been replaced by the *OSF DCE Testing Guide*, which contains only DCE functional and system testing information (updated for DCE 1.2.2); all porting and ODE information has been removed.

Note that the porting information in the *OSF DCE Porting and Testing Guide* has *not been updated* since DCE 1.1, and is not guaranteed to be accurate for OSF DCE 1.2.2.

The *OSF DCE Technical Supplement* is a collection of functional and architectural specifications written by DCE providers. These specifications have not been updated since DCE 1.0.3, and are not guaranteed to be accurate for DCE 1.2.2. The *Technical Supplement* is for *licensees' internal use only* and therefore cannot be redistributed in part or whole by any licensee. The *OSF DCE Technical Supplement* is made up of PostScript files, except for two security specifications which are ASCII files.

Note that the contents of this book are identical to the version that was included on the DCE 1.0.3 and DCE 1.1 tape. It has not been updated to include any DCE 1.1 or DCE 1.2.2 information, and it may contain some information that is not accurate for DCE 1.2.2.

Specifications (RFCs) that were relevant for DCE 1.2.2 are located on the shipped tape (or CD-ROM) in:

dce1.2.2-doc-dir/**DCE1.2.2_specs**

Some of this documentation may include information that has been superseded as a result of design changes that postdate the writing of the specifications.

The *OSF DCE DFS Application Development Guide and Reference*, which has not been supported since DCE 1.1, consists of the DFS portions of the *OSF DCE Application Development Guide* and the *OSF DCE Application Development Reference*, which were moved out of those books at DCE 1.1.

5.3 Summary of Documentation Structure Changes Since DCE 1.2.1

The documentation set for DCE 1.2.2 embodies the following major changes since DCE 1.2.1:

- Conversion into SGML source form. See Chapter 6.
- Elimination of the SML source versions, and some other supplementary documentation, from the distribution. See Section 5.2.
- Addition of documentation of the Public Key Certification API to the *OSF DCE Application Guide—Core Components* volume and to the *OSF DCE Application Reference*.
- Addition to the *OSF DCE Application Development Guide—Core Components* volume of a simple source code example demonstrating the use of the **dced** application programming interface.
- Additions to the *OSF DCE DFS Administration Guide and Reference*.
- Addition of DFS information to the *OSF DCE Problem Determination Guide*.

Details of all but the first two of these items will be found below.

Information relating to internal details of the SGML conversion and other non-content-related matters will be found in Chapter 6. Remarks on the present visual (formatted) state resulting from the conversion will be found for each book below, in the separate subsections. A general statement for the entire document set on this aspect of the conversion will be found immediately below.

5.4 General State of the Documentation at Release 1.2.2

All SGML-converted books were formatted using the ArborText *ADEPT* Publisher version 5.0.2 software.

In most cases, the converted formatted books come close to the quality of the SML formatted books distributed with DCE 1.2.1 and earlier versions. However, within the books many tables and some figures require additional work to bring them up to publishing quality.

During the conversion from SML font directives to SGML semantic tagging, many terms (such as variables, typedefs, attributes, and so on) received imprecise or incorrect tagging. However, in most cases the typographical rendition remains correct.

All books were proofread. All typographic and grammatical errors found were corrected. Many stylistic and consistency edits resulting from the proofreading remain to be incorporated.

5.5 State of Each Document in DCE 1.2.2

The following sections contain detailed information about the changes made to each book in the DCE 1.2.2 documentation set since DCE 1.2.1.

A list of all currently-known DCE 1.2.2 restrictions and defects, including documentation defects, is located in the **project** archive at:

dce1.2.2-root-dir/**project/defect.summary**

5.5.1 Introduction to OSF DCE

The formatted appearance of the *Introduction to OSF DCE* is of fairly high quality. Some tables require additional work to bring them up to publishing quality.

No material change to the contents of this book has occurred since DCE 1.2.1. Short summary descriptions of the new DCE 1.2.2 facilities (public key login, public key certificates) were added.

5.5.2 OSF DCE Administration Guide — Introduction

The formatted appearance of the *OSF DCE Administration Guide—Introduction* is of fairly high quality. Some tables require additional work to bring them up to publishing quality.

No major changes were made to the contents of this book since DCE 1.2.1. Some bug-fixing work, described in OT CRs 13555, 13556, and 13493, was done.

5.5.3 OSF DCE Administration Guide — Core Components

The formatted appearance of the *OSF DCE Administration Guide—Core Components* volume is of fairly high quality. Some tables require additional work to bring them up to publishing quality.

See Chapter 2, ‘‘State of Security Code’’, for information about procedures for specifying public key passwords which does not appear in this book.

Additions were made to this book to describe the following new DCE 1.2.2 features:

- Multi-cell groups
- User-to-user authentication
- Public key login **dcecp** operations (from DCE 1.2.1 work)
- **dcecp** ERA operations (from DCE 1.2.1 work)
- Kerberos 5 interoperability

Also, changes were made in this book to reflect bug fixes to **dcecp**.

5.5.4 OSF DCE Command Reference

The formatted appearance of the *OSF DCE Command Reference* is of fairly high quality. Some tables require additional work to bring them up to publishing quality.

See Chapter 2, ‘‘State of Security Code’’, for information about procedures for specifying public key passwords which does not appear in the *Command Reference*.

The following additions were made to this book during DCE 1.2.2:

Reference pages were added documenting Kerberos V5 interoperability features:

- **k5dcelogin(8sec)**
- **k5login(8sec)**
- **rlogin(8sec)**

- **rlogind(8sec)**
- **rsh(8sec)**
- **rshd(8sec)**

Additional improvements were made to the **dcecp** (**man3dce**) reference pages during DCE 1.2.2. This work was in part a continuation from DCE 1.2.1, and is described in the following OT CRs:

- 13377
- 13441
- 13480
- 13493
- 13499
- 13501
- 13505
- 13506
- 13507
- 13508
- 13511
- 13512
- 13513
- 13514
- 13515
- 13517
- 13519
- 13520
- 13521
- 13525
- 13524
- 13625
- 13526
- 13527
- 13528
- 13530
- 13539

- 13543
- 13546
- 13547
- 13548
- 13549
- 13550
- 13551
- 13588
- 13592

The **sec_salvage_db(8sec)** (absent in recent versions) was reincorporated into the book in DCE 1.2.2.

5.5.5 OSF DCE Application Development Guide

Remarks on the contents of each of the three volumes of this book appear in the following three subsections.

5.5.5.1 Volume One: Introduction and Style Guide

The formatted appearance of the *OSF DCE Application Development Guide— Introduction and Style Guide* volume is of fairly high quality. Some tables require additional work to bring them up to publishing quality.

No material change to the contents of this book has occurred since DCE 1.2.1.

5.5.5.2 Volume Two: Core Components

The formatted appearance of the *OSF DCE Application Development Guide— Core Components* volume is of fairly high quality. Some tables require additional work to bring them up to publishing quality.

A **dced** programming example has been added to Chapter 2 (Section 2.9, ‘‘Sample dced Application’’) (CR 9838).

Chapter 37, ‘‘The DCE Certification Service’’, has been added. This chapter documents the DCE certification service, which provides for the secure retrieval of public keys, stored (through the DCE directory service) under the names of the principals with which the keys are associated. It is a name-to-public key translation service intended to be used both by DCE components and DCE applications. The keys are stored in and retrieved

from data structures called ‘‘certificates’’.

Other additions to this book made for DCE 1.2.2 include material to document the new functionality for: public key login (see OT CR 13609), single-threaded clients, private key storage server (CR 13609), and user-to-user authentication (CR 13562).

5.5.5.3 Volume Three: Directory Services

The formatted appearance of the *OSF DCE Application Development Guide— Directory Services* volume is of fairly high quality. Some tables require additional work to bring them up to publishing quality.

No material change to the contents of this book has occurred since DCE 1.2.1.

5.5.6 OSF DCE Application Development Reference

The formatted appearance of the *OSF DCE Application Development Reference* is of fairly high quality. Some tables require additional work to bring them up to publishing quality.

The OSF DCE Application Reference contains the following new reference pages for DCE 1.2.2, which constitute part of the documentation for the DCE Certification facility. Note that the SML versions of these reference pages (available electronically from OSF, as described above) do *not* have change markers. The SGML versions of the pages do, however, have the change bars.

- High-level API manpages (16):
  - **cert_intro(3sec)**
  - **pkc_add_trusted_key(3sec)**
  - **pkc_append_to_trustlist(3sec)**
  - **pkc_free.ps pkc_free_keyinfo(3sec)**
  - **pkc_free_trustbase(3sec)**
  - **pkc_free_trustlist(3sec)**
  - **pkc_get_key_certifier_count(3sec)**
  - **pkc_get_key_certifier_info(3sec)**
  - **pkc_get_key_count(3sec)**
  - **pkc_get_key_data(3sec)**
  - **pkc_get_key_trust_info(3sec)**
  - **pkc_get_registered_policies(3sec)**

- **pkc_init_trustbase(3sec)**
- **pkc_init_trustlist(3sec)**
- **pkc_retrieve_keyinfo(3sec)**
- **pkc_retrieve_keylist(3sec)**

- Trust list (low-level C++ API) manpages (12):
  - **cert_intro_trustlist(3sec)**
  - **pkc_add_trusted_key(3sec)**
  - **pkc_check_cert_against_trustlist(3sec)**
  - **pkc_copy_trustlist(3sec)**
  - **pkc_delete_trustlist(3sec)**
  - **pkc_display_trustlist(3sec)**
  - **pkc_lookup_element_in_trustlist(3sec)**
  - **pkc_lookup_key_in_trustlist(3sec)**
  - **pkc_lookup_keys_in_trustlist(3sec)**
  - **pkc_prune_trustlist(3sec)**
  - **pkc_revoke_certificate(3sec)**
  - **pkc_revoke_certificates(3sec)**
- Trust list C++ class manpages (16):
  - **pkc_ca_key_usage.class(3sec)**
  - **pkc_constraints.class(3sec)**
  - **pkc_generic_key_usage.class(3sec)**
  - **pkc_key_policies.class(3sec)**
  - **pkc_key_policy.class(3sec)**
  - **pkc_key_usage.class(3sec)**
  - **pkc_name_subord_constraint.class(3sec)**
  - **pkc_name_subord_constraints.class(3sec)**
  - **pkc_name_subtree_constraint.class(3sec)**
  - **pkc_name_subtree_constraints.class(3sec)**
  - **pkc_pending_revocation.class(3sec)**
  - **pkc_revocation.class(3sec)**
  - **pkc_revocation_list.class(3sec)**
  - **pkc_trust_list.class(3sec)**
  - **pkc_trust_list_element.class(3sec)**

  - **pkc_trusted_key.class(3sec)**
- Cryptographic module API manpages (7)
  - **cert_intro_crypto(3sec)**
  - **pkc_crypto_generate_keypair(3sec)**
  - **pkc_crypto_get_registered_algorithms(3sec)**
  - **pkc_crypto_lookup_algorithm(3sec)**
  - **pkc_crypto_register_signature_alg(3sec)**
  - **pkc_crypto_sign(3sec)**
  - **pkc_crypto_verify_signature(3sec)**
- Policy module API manpages (13):
  - **cert_intro_policy(3sec)**
  - **pkc_plcy_delete_keyinfo(3sec)**
  - **pkc_plcy_delete_trustbase(3sec)**
  - **pkc_plcy_establish_trustbase(3sec)**
  - **pkc_plcy_get_key_certifier_count(3sec)**
  - **pkc_plcy_get_key_certifier_info(3sec)**
  - **pkc_plcy_get_key_count(3sec)**
  - **pkc_plcy_get_key_data(3sec)**
  - **pkc_plcy_get_key_trust(3sec)**
  - **pkc_plcy_get_registered_policies(3sec)**
  - **pkc_plcy_lookup_policy(3sec)**
  - **pkc_plcy_register_policy(3sec)**
  - **pkc_plcy_retrieve_keyinfo(3sec)**

5.5.7 DCE DFS Administration Guide and Reference

The formatted appearance of the *OSF DCE DFS Administration Guide and Reference* is of fairly high quality. Some tables require additional work to bring them up to publishing quality. Section heads on the reference pages need to be initial-capped, and names removed from ‘‘Purpose’’ sections of reference pages.

The following changes were made to the DFS administrative documentation for DCE 1.2.2:

- Documentation for multihomed server functionality was added. Multihomed servers are File Servers or Fileset Location Servers that have up to four entries in the fileset location database; each entry has a different IP address. Individual Cache Managers can then give a rank to each server entry in the preference list.

The following files were modified for this change:

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/gd/overview.gpsml**

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/gd/ftavail.gpsml**

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/gd/issues.gpsml**

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/gd/cachemgr.gpsml**

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/ref/man8dfs/cm_setpreferences.8dfs**

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/ref/man8dfs/cm_getpreferences.8dfs**

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/cm_statservers.8dfs**

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/ref/man8dfs/cm_lscellinfo.8dfs**

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/ref/man8dfs/cm_whereis.8dfs**

The following files were created for this change:

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/gd/figures/pref1.ps**

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/gd/figures/pref2.ps**

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/gd/figures/pref3.ps**

dce1.2.2-doc-dir/**src/dce_books/dfs_admin_gdref/gd/figures/override1.ps**

This change is documented in DCE OT CR (defect report) 13566, which can be found in the DCE defects database, located in the

dce1.2.2-root-dir/**project/defect.summary**

directory.

- Documentation for security enhancements was added. These enhancements allow users to set an initial and minimum DCE protection level to be used for communications between the Cache Manager and File Servers. They also allow system administrators to set a maximum and minimum DCE protection level for File Servers to accept when communicating with Cache Managers. System administrators also can now set advisory protection level bounds on a per fileset basis. These bounds currently serve to bias the Cache Manager's initial protection level. All of these new features have separate settings for communications within the same cell and communications with foreign cells.

  The following files were modified for this change:

dce1.2.2-doc-dir/**src/dcebooks/dfs_admin_gdref/ref/man8dfs/cm_getprotectlevels.8dfs**

dce1.2.2-doc-dir/**src/dcebooks/dfs_admin_gdref/ref/man8dfs/cm_setprotectlevels.8dfs**

dce1.2.2-doc-dir/**src/dcebooks/dfs_admin_gdref/ref/man8dfs/fts_setprotectlevels.8dfs**

dce1.2.2-doc-dir/**src/dcebooks/dfs_admin_gdref/ref/man8dfs/dfsd.8dfs**

dce1.2.2-doc-dir/**src/dcebooks/dfs_admin_gdref/ref/man8dfs/fts_lsfldb.8dfs**

dce1.2.2-doc-dir/**src/dcebooks/dfs_admin_gdref/ref/man8dfs/fts_lsft.8dfs**

dce1.2.2-doc-dir/**src/dcebooks/dfs_admin_gdref/ref/man8dfs/fxd.8dfs**

dce1.2.2-doc-dir/**src/dcebooks/dfs_admin_gdref/gd/ftmgmt.gpsml**

dce1.2.2-doc-dir/**src/dcebooks/dfs_admin_gdref/gd/ftavail.gpsml**

dce1.2.2-doc-dir/**src/dcebooks/dfs_admin_gdref/gd/overview.gpsml**

dce1.2.2-doc-dir/**src/dcebooks/dfs_admin_gdref/gd/cachemgr.gpsml**

dce1.2.2-doc-dir/**src/dcebooks/dfs_admin_gdref/gd/issues.gpsml**

This change is documented in DCE defect report 13605, which can be found in the DCE defects database, located in the

dce1.2.2-root-dir/**project/defect.summary**

directory.

5.5.8 OSF GDS Administration Guide and Reference

The formatted appearance of the *OSF DCE GDS Administration Guide and Reference* is of less quality than that of the other books. While all information is present and legible, many tables need substantial work to bring them up to the level of tables in other DCE books. Many of the figures need additional scaling, although the figures are currently present in the formatted text and within the printing image area.

No material change to the contents of this book has occurred since DCE 1.2.1.

The following items contain more detailed information about the formatted state of this book.

- Many screens need blank lines added to give consistent depth.
- Several **SimpleLists** need **<Literal>** tagging carried through the whole list (not just the first item).
- The reference pages section, which follows the appendixes, is called ‘‘Chapter 12.’’ This was true, also, in the SML version of the book, but is inappropriate.

5.5.9 OSF DCE File-Access Administration Guide and Reference

The formatted appearance of the *OSF DCE DFA Administration Guide and Reference* is of fairly high quality. Some tables require additional work to bring them up to publishing quality. Several PostScript figures in this book caused build failures and were commented out.

No material change to the contents of this book has occurred since DCE 1.2.1.

5.5.10 OSF DCE File-Access User's Guide

The formatted appearance of the *OSF DCE DFA User's Guide* is of fairly high quality. Some tables require additional work to bring them up to publishing quality. Several PostScript figures in this book caused build failures and were commented out.

No material change to the contents of this book has occurred since DCE 1.2.1.

5.5.11 DFA Functional Verification Test (FVT) USer's Guide

No material change to the contents of this book has occurred since DCE 1.2.1.

Note that this book was *not* converted to SGML. For additional information about this book, see Chapter 6.

5.5.12 OSF DCE Problem Determination Guide

The formatted appearance of the *OSF DCE Problem Determination Guide* is of fairly high quality.

The following major additions have been made to this book for DCE 1.2.2:

- Problem determination information was added for the DCE DFS component.
- Messages added or otherwise updated for various other already-present DCE components.

5.5.13 OSF DCE Testing Guide

For the most part, the *OSF DCE Testing Guide* consists of DCE testing documentation taken over unchanged from the DCE 1.1 version (the latest) of the *OSF DCE Porting and*

Testing Guide, which has not been updated for DCE 1.2.2. However, the following testing information, reflecting changes in DCE testing for DCE 1.2.2, has been added:

- C++ IDL extensions testing
- Public Key Storage Server (PKSS) testing
- Certification API testing
- Global Groups manual testing
- Kerberos 5 testing
- Public Key testing
- DFS Delegation testing
- DFS Multihome server testing
- DFS File Exporter Authorization testing

Each added or modified section is marked with horizontal change bars.

Chapter 6. Source Form of the DCE 1.2.2 Documentation

It is *not necessary to build the DCE 1.2.2 documentation,* as formatted output for all the documentation accompanies the source files. For instructions on printing the books, see the section ‘‘Printing the DCE 1.2.2 Documentation’’ later in this chapter.

This chapter primarily describes and explains mechanical details of the DCE books’ source and output form, and their implications. For information about the actual contents of the DCE 1.2.2 documentation, as well as information about DCE documentation not included with the distribution but available electronically from OSF, see Chapter 5.

6.1 Location of the doc Tree

Throughout the remainder of this chapter, it is assumed that the DCE 1.2.2 archives are found at the following pathname on your system:

your-root-dir/**dce/V1.2.2**

—where *your-root-dir* is the location on your disk where you installed the archives, as described in Chapter 4 of this document. (Refer to the section ‘‘Contents of the DCE 1.2.2 Release Area’’ in Chapter 1 of this document for more information.) This pathname is abbreviated as

dce1.2.2-root-dir

in the other chapters of this document.

Immediately beneath this pathname will be found several directories corresponding to the separate archives extracted from the DCE 1.2.2 distribution media, as described in Chapter 4. For example, the directory

dce1.2.2-root-dir/**dce**

will contain the extracted **dce** archive, which consists of the DCE 1.2.2 source tree; and the directory

dce1.2.2-root-dir/**doc**

will contain the extracted **doc** archive, which consists of the DCE 1.2.2 documentation set (further described in Chapter 5). For the sake of additional brevity and clarity, this **doc** archive pathname will be abbreviated as

dce1.2.2-doc-dir

throughout the remainder of this chapter. For example, the pathname

dce1.2.2-doc-dir/**src/dce_books_sgml/app_gd_style**

is equivalent to:

your-root-dir/**dce/V1.2.2/doc/src/dce_books_sgml/app_gd_style**

For pathnames outside of the **doc** tree, the

dce1.2.2-root-dir

notation will be used.

An illustration of the structure of the **doc** tree can be found in Chapter 3 of this document.

6.2 SGML Conversion of DCE 1.2.2 Documentation Source

For DCE 1.2.2 the entire documentation set (with the exception of the *Release Notes*—the document you are reading—and the *OSF DCE Testing Guide*) was converted from its previous source form, SML (‘‘semantic markup language’’—essentially enhanced troff **mm**/**man** macro markup) to SGML (‘‘Standard Generalized Markup Language’’—ISO 8879:1986). The DocBook version 2.4 Document Type Definition (DTD) was used.

A complete set of DocBook 2.4 DTD source files is supplied with DCE 1.2.2, together with a set of sources for the OSF FOSI which determines (in conjunction with the ArborText *ADEPT* SGML software used at OSF) the formatted style of the DCE books (see ‘‘DocBook DTD V2.4’’ below). The books themselves are supplied both in SGML source form and as PostScript formatted output, as in previous versions of DCE.

Note that *no formatting tools* (other than the DTD and FOSI sources mentioned above) are supplied with DCE 1.2.2. To reformat the books yourself you will have to recompile the DTD and FOSIs using ArborText *ADEPT* or some other SGML system. The FOSIs are a completely *ADEPT*-dependent mechanism for specifying formatting style (no standard method for such specification yet exists), so licensees not using *ADEPT* should read the *OSF DocBook Processing Guide for DCE* (see ‘‘OSF FOSI’’ below) to learn which tags were used for DCE 1.2.2 and how they were stylistically interpreted by the FOSI.

6.2.1 Books Supplied in the DCE 1.2.2 Documentation Set

The DCE books now distributed in SGML are the following:

- *OSF DCE Administration Guide—Core Components*
- *OSF DCE Administration Guide—Introduction*
- *OSF DCE Application Development Guide—Introduction and Style Guide*
- *OSF DCE Application Development Guide—Core Components*
- *OSF DCE Application Development Guide—Directory Services*
- *OSF DCE GDS Administration Guide and Reference*
- *OSF DCE DFS Administration Guide and Reference*
- *OSF DCE Problem Determination Guide*
- *OSF DCE Introduction to DCE*
- *OSF DCE Command Reference*
- *OSF DCE Application Development Reference*
- *OSF DCE/File-Access Administration Guide and Reference*
- *OSF DCE/File-Access User's Guide*

The following books are distributed (in DCE 1.2.2) in PostScript formatted output files generated from SML source which is also included in the distribution:

- *OSF DCE Testing Guide*
- *OSF DCE Release Notes*

None of the other documentation material (*OSF DCE Technical Supplement*, unsupported books such as the still-available DCE 1.1 *Porting Guide*, and so on) have changed form (these books, although not included on the DCE 1.2.2 distribution media, are available electronically from OSF; see Chapter 5).

The DCE 1.2.2 documentation, as before, is made available on the distribution media in both source and PostScript output form.

Tools for formatting the DCE books from source are no longer distributed. This is true both for the SGML-converted books, and for the two books still distributed in SML. The Documentation Tools Environment (DTE) is no longer included on the distribution media, although it can be obtained electronically from OSF; for further information, see Section 5.2 (''Supplemental DCE Documentation Available Electronically'').

Note: Although the SML versions of the DCE documentation are not included on the DCE 1.2.2 distribution media, they (and the DTE tools used to build them) are still available electronically from OSF. See Section 5.2.

6.3 DocBook DTD V2.4

The DocBook version 2.4 document type definition (DTD) was used for the converted SGML DCE 1.2.2 documentation (version 2.4.1 was not available at the time work began). The files used to compile the DTD (using the *ADEPT* version 5.0.2 Document Architect software—the ‘‘**bigdocarch**’’ version was used—running on an HP-UX version 10.10 platform) were obtained by OSF from the Davenport Group archive at the following location:

ftp://ftp.ora.com/pub/davenport/README.html

The files were later slightly modified at OSF (the modifications are described below). The entire set of files, organized under a **DocBook** directory, is included with the DCE 1.2.2 distribution at the following location in the **doc** tree:

dce1.2.2-doc-dir/**src/dce_books_sgml/DocBook**

See ‘‘DTD Installation’’ below for a complete list and description of the DTD and FOSI source files provided.

Note that the FOSI used to format the DCE books is also located in this directory; see ‘‘OSF FOSI’’ below.

6.3.1 Compiling the DTD and FOSIs

The DocBook 2.4 DTD and OSF FOSI in the **DocBook** directory are present in source form only; in order to use them with your *ADEPT* (or other) SGML software, you must re-compile them on your system. With the *ADEPT* Document Architect, the following is probably what you will have to do:

1. Import, compile and install the DTD.

   For the version 5.0.2 Document Architect, the SGML declaration (contained in the **docbook.dcl** file) had to be manually inserted at the top of the **DocBook.dtd** file. After this was done, the DTD proper had to be surrounded with

   ```
   <!DOCTYPE Book
   [
   ```

   and

   ```
   ]>
   ```

   coding. The DTD was then able to successfully be compiled.

2. Compile and install both (‘‘publish’’ and ‘‘draft’’ style) FOSIs.

   When you import and compile a new DTD, Document Architect creates a *doctype*-**da** directory to hold the imported files and the results of its own activities during the session. Within the the *doctype*-**da** directory will be found a

subdirectory named simply *doctype* that will contain the DTD, FOSI, and other files (both source and compiled output, if and when the latter is produced). *You must make sure that Document Architect copies (imports) all necessary files* into its working directory. In some cases this will require manual copying on your part, as Document Architect by no means handles this phase of the operation in the most desirable way. This warning particularly applies to the FOSI files.

Owing to an inconsistency in the naming of the FOSI source files, there was a difference in the way the version 5.0.2 Document Architect handled the FOSI compilations. The **DocBook.fos** file contains the default (''publish'' style) FOSI, and it is compiled by default when the DTD is compiled (evidently because it has the same name as the doctype). However, the ''draft'' style FOSI is contained in a file named **draft.fos**, and it (together with the **docbook.lcl** file) had to be manually copied into the Document Architect working directory (**DocBook-da**) before it too could be selected for compilation and installation.

6.4 Characteristics of the DCE 1.2.2 SGML Source

Conversion was accomplished for each book through a two-step procedure consisting of automated processing followed by extensive touch-up and refinement by hand at OSF (for details of the current state of each converted book, see Chapter 5). The SGML tools used at OSF to parse and format the converted books were the ArborText *ADEPT* series version 5.0.2 Editor/Publisher (the ''**bigadeptpub**'' version was necessary because of the size of the DCE books) and Document Architect (''**bigdocarch**'' was used) running on an HP-UX version 10.10 platform. The organization and content of the DCE 1.2.2 SGML source exhibits some *ADEPT*-specific characteristics, but these should not interfere with licensees' ability to use other SGML software with the books. Specific characteristics of the DCE 1.2.2 SGML source include the following:

- Nomenclature and organization of the book source directories (for **pubformat**).
- Modifications made to the DTD.
- The OSF FOSI.
- Use of *ADEPT* Publisher formatting commands (**<?Pub ...>** markup).
- CALS table code (subject to *ADEPT* 5.0.2 table restrictions).
- Presence of (inert) conversion-generated markup preserving some of the original SML formatting commands.
- Miscellaneous.

These topics are described and discussed in more detail in the following sections.

6.4.1 Nomenclature and Organization of SGML Source

An illustration of the higher-level structure of the **doc** tree can be found in Chapter 3 of this document.

Broadly speaking, the organization of the SGML source is similar to that of the previous SML source form of the books: under **dce_books_sgml** lies a series of directories each of which contains the suite of source files that make up a single separate book. Reference page source and PostScript figure files are held in subdirectories of the book directory. Within its directory, each book is organized at the highest level as a

*book_name***.sgml**

file.

Note: Since each book exists as an assemblage of lower-level source pieces brought together only via the file entity references in the **.sgml** file, only the **.sgml** file needs to have a **DOCTYPE** declaration. However, most of each book's lower-level files have had their own separate **DOCTYPE** declarations added in order to make them individually understandable to (and loadable by) the *ADEPT* software. These lower-level declarations are in the form of SGML comments which are understandable only to the *ADEPT* software. For further information, see ''Running docbook-to-man'' below.

Within each book's top-level **.sgml** file is a series of entity declarations and references to the file entities from which most of the source for the body of the book is drawn. Books' entity filenames are generally of the form

*name_string***.sgm**

for text files, and

*name***.**<*n*><*trigraph*>

for reference pages (where <*n*> is a single-digit section number, and <*trigraph*> is a 3-character DCE component abbreviation).

The name of each book's top-level **.sgml** file is identical to the name of the book directory which it inhabits. This structure is required by the **pubformat** utility which was used to batch-build the books at OSF (see ''DCE 1.2.2 Documentation Batch Build'' below).

Most book directories contain a pair of

BEntities.sgm
BOOKDEFS.sgm

files. The first-named of these contains a set of declarations for the book's file entities; the second file contains (rather optimistically) a complete set of entity declarations for the entire range of documentation produced by OSF, enabling any of these titles to be cited by entity reference rather than literal string. These two files were originally intended to be brought into each book's top-level **.sgml** file through a pair of parameter entity declarations. However, parameter entities do not work in *ADEPT* 5.0.2, so the declarations were instead coded directly in the **.sgml** files for DCE 1.2.2.

A third extra file, with a name of the following form:

book-name-string-**extent.sgm**

is present in some book directories. This file is a relic of an earlier stage of the SML-to-SGML conversion development effort, and should be ignored. It can even be deleted.

Multiple-volume DCE books are now (for all intents and purposes) organized as separate books in the SGML source directory structure. For example, where the *OSF DCE Application Development Guide* source used to be arranged as three volumes organized (in the SML source) under the **app_gd** directory, there are now three separate independent SGML source book directories named **app_gd_style** (*OSF DCE Application Development Guide—Introduction and Style Guide*), **app_gd_core**, (*OSF DCE Application Development Guide—Core Components*), and **app_gd_dir** (*OSF DCE Application Development Guide—Directory Services*). Similarly, the two volumes of the *OSF DCE Administration Guide* now exist as separate SGML source directories named **admin_gd_intro** (*OSF DCE Administration Guide—Introduction*) and **admin_gd_core** (*OSF DCE Administration Guide—Core Components*).

6.4.1.1 Names of Reference Page Subdirectories

In the *OSF DCE Application Reference* and the *Command Reference*, the reference page subdirectories have the same names as their SML originals: for example, **man3dce**, **man8cds**, and so on. However, the reference page subdirectories for the *DFS Administration Guide and Reference* and *GDS Administration Guide and Reference* have a different form of name: **MAN8**, **MAN8A** and **MAN8B** (three directories in all) for the GDS book, and a series of 32 subdirectories for the DFS book, with names of the form **DFS4** or **DFS8**, followed by another single uppercase character. (The *DFA Administration Guide and Reference*'s single reference page subdirectory is likewise named **MAN8**.) The unusual name scheme employed in DCE 1.2.2 for these reference page subdirectories is an inadvertent artifact of the SGML conversion, and will be corrected.

6.4.1.2 Declarations of File Entities

For all the DCE books, file entities referenced from the top-level **.sgml** file embodying book pieces fall (from the point of view of source file organization) into two broad categories:

- Files located at the same level of the book directory.
- Files located in subdirectories of the book directory.

A typical file entity declaration in an **.sgml** file is formed as follows:

```
<!ENTITY FIG6N14 SYSTEM "eps/A06-14.eps" NDATA EPS>
```

where the quoted string appearing after the keyword **SYSTEM** specifies the pathname of the file being declared (relative to the location of the file in which the declaration appears), and the string following **ENTITY** specifies the declared label for the file. The above declaration allows **FIG6N14** to be used later on to reference the PostScript figure file **A06-14.eps** and include it into the formatted book.

Users will notice a basic inconsistency in the way file entity pathnames are declared in the DCE books for files that lie in book subdirectories. PostScript figure files (known as ''graphics'' entities to SGML), such as the entity declared in the above example, are declared with pathnames relative to the book directory. However, *all other files* located in subdirectories are declared with pathnames relative to *the level above the book directory* (namely, **dce_books_sgml**). The inconsistency resulted from work done during DCE 1.2.2 to make the books understandable to the *ADEPT* software.

Both the window-based *ADEPT* Editor/Publisher and the command line-invoked *ADEPT* **pubformat** utility were used to format the DCE books during the DCE 1.2.2 work. And although local-relative pathnames for all entities are understood by the version 5.0.2 Editor/Publisher (provided the software is executed at the book's top directory level), **pubformat** can be successfully invoked *only from immediately above* a book directory, and requires the pathnames of all non-graphics file entities declared in the book to be expressed relative to that location. Why graphics files formed an exception to this rule was never understood, but they were left in their original book directory-relative form, since they worked that way.

The practical consequence of all this is—probably—the following:

- When editing or reformatting the books using *ADEPT* software, you must always be sure to invoke the Editor/Publisher (or **pubformat**) from *above* the book directory—that is, from the

  dce1.2.2-doc-dir/**src/dce_books_sgml**

  level.

- Using other SGML software with the books may possibly require some recoding of the file entity declarations, owing to the inconsistency described above.

6.4.1.3 DCE 1.2.2 Documentation Batch Build

A copy of the script used at OSF to do batch-builds of the SGML-converted books and to generate the contents of the distribution DCE 1.2.2 documentation directory

dce1.2.2-doc-dir/**src/built_books**

is included with DCE 1.2.2. The script is composed of two parts, and is run by executing

dce1.2.2-doc-dir/**src/exec/do_build.ksh**

—a wrapper which sets up logging and invokes

dce1.2.2-doc-dir/**src/exec/build_sgml.ksh**

—the main script. The script is completely dependent on the ArborText *ADEPT* **pubformat** book-formatting utility and fully reflects the latter's many limitations. It will

not be usable with other software. (The script also depends on the presence of the **psselect** utility for splitting up the very large single PostScript files produced by **pubformat**.) The script was run at OSF on an HP-UX version 10.10 platform. The script is not supported by OSF and is included for licensees' information only.

6.4.1.3.1 File Dependencies in Book Builds

Under the following directory:

dce1.2.2-doc-dir/**src/dce_books_sgml/entities**

the following files will be found:

common-characterfills.ent
common-characterfills.ent.grp
common-counters.ent
common-eics.ent
common-eics.ent.grp
common-textIDs.ent
common-textIDs.ent.grp
index-eics.ent
revflag.ent
stylesets.ent

These files, which contain definitions for various entities used by the OSF FOSIs, must be present at the above-described relative location to the SGML source in order for *ADEPT* Publisher formatting to work. If *ADEPT* Publisher does not find these entities, you will receive the following alarming message

```
FOSI compile failed
```

when you load any SGML source.

Note also that **pubformat** requires 7 of the above files to be present under the following names:

common-characterfills
common-counters
common-eics
common-textIDs
index-eics
revflag
stylesets

at the *book_directory* level (that is, in **dce_books_sgml**) in order to successfully build a book. The 7 files are identical to their similarly-named counterparts in the **entities** directory, but **pubformat** requires that they not have **.ent** extensions. The contents of the

dce1.2.2-doc-dir/**src/exec/build_sgml.ksh**

script (see ''DCE 1.2.2 Documentation Batch Build'' above) will show the details of how this was arranged at OSF.

6.4.1.4 DCE 1.2.2 Book Build Log

A log of a complete build of the DCE 1.2.2 SGML-converted documentation set can be found at:

dce1.2.2-doc-dir/**logs**

The log was generated by the build scripts described in ‘‘DCE 1.2.2 Documentation Batch Build’’ above.

6.4.2 Modifications to the DTD Files

Several minor modifications had to be made at OSF to the DocBook 2.4 DTD files mentioned above in order to enable certain necessary formatting features. This section describes these modifications.

- In the **dbpool.mod** file, the **local.graphics.attrib** entity, which originally had a null definition as follows:

  ```
  <!ENTITY % local.graphics.attrib "">
  ```

  was supplied with three attribute definitions as follows:

  ```
  <!ENTITY % local.graphics.attrib
          "Scalefit       NUMBER          #IMPLIED
          Reprodep        NUMBER          #IMPLIED
          Reprowid        NUMBER          #IMPLIED"
  >
  ```

  This change was made to support PostScript figure scaling.

- In the **DocBook.atd** (the ‘‘atd’’ signifies ‘‘auxiliary tag data’’) file, the graphic traits declarations were originally without definitions as follows:

  ```
  GraphicTraits
          GraphicTag:
          GraphicEntityAttr:
          GraphicFileNameAttr:
          GraphicProcTypeAttr:
          GraphicMagAttr:
          GraphicResAttr:
  ```

  At OSF the **GraphicEntityAttr** trait was defined as **Entityref** to support entity references (as opposed to literal pathnames) in <**Graphic**> elements, and the **GraphicScalefit**, **GraphicReprodep**, and **GraphicReprowid** traits were declared and defined in order to enable figure scaling. The modified lines of the file now appear as shown below:

```
GraphicTraits
        GraphicEntityAttr: Entityref

             <......>

        GraphicScalefit: Scalefit
        GraphicReprodep: Reprodep
        GraphicReprowid: Reprowid
```

6.4.2.1 DTD Installation

The DocBook 2.4 DTD was compiled and installed at OSF on an HP-UX version 10.10 platform using the ArborText *ADEPT* Document Architect 5.0.2 software (the ‘‘**bigdocarch**’’ version was used in order to accommodate the very large DCE books).

The DTD and FOSI were compiled from the following source files, all of which are supplied with DCE 1.2.2:

DocBook.dtd Document type definition (DTD) file. The following files contain various ingredients of the DTD and are compiled with it:

- **docbook.dcl** SGML declaration file. For *ADEPT* Document Architect 5.0.2 the contents of this file had to be inserted into the top of **DocBook.dtd** in order to successfully compile the latter. You will need to retain a separate copy of this file if you intend to run the **docbook-to-man** conversion program.
- **calstbl.mod** CALS table document type definition.
- **dbgenent.mod** DocBook additional general entities.
- **dbhier.mod** DocBook document hierarchy module V2.4.
- **dbpool.mod** DocBook information pool module V2.4.

DocBook.atd Auxiliary tag definition file.

docbook.cat DocBook catalog data V2.4. You will need this file if you intend to run the **docbook-to-man** conversion program.

docbook.lcl Auxiliary FOSI file.

DocBook.fos OSF ‘‘publish’’ style FOSI source.

draft.fos OSF ‘‘draft’’ style FOSI source.

pubrc *ADEPT* DocBook instance initialization file. This file must be present in order to allow *ADEPT* to choose between ‘‘draft’’ or ‘‘publish’’ style formatting at the beginning of an editing session on an SGML document.

Note that some filenames in the list above use the string ‘‘**DocBook**’’, while others have the same string spelled as ‘‘**docbook**’’. This seeming discrepancy is due to the fact that the 2.4 DocBook DTD was installed at OSF under the name **DocBook** (a directory name

in the *ADEPT* installation area), while the doctype name which it represents is a case-indifferent string in SGML and is usually spelled all-lowercase. No harm results from the variation in names. (But see also ‘‘Conversion of Reference Pages from SGML to nroff’’ below.)

For information about the FOSI used with the DTD to produce OSF style guideline-compliant formatting, see the following section.

6.4.3 The OSF FOSI

The DCE 1.2.2 documentation was formatted in compliance with a collection of specifications set forth in the *OSF DocBook Processing Guide for DCE* and realized by the *ADEPT* Publisher FOSI mechanism, as implemented in the FOSI source files supplied with DCE 1.2.2 (for the location of the FOSI sources, see ‘‘DTD Installation’’ above). Two formatting styles (‘‘publish’’ and ‘‘draft’’) are possible using the FOSIs. The ‘‘publish’’ style is used by default and it is the style in which the PostScript output distributed with DCE 1.2.2 were formatted.

The FOSIs are a completely *ADEPT*-dependent mechanism for specifying formatting style, so licensees not using *ADEPT* should read the *OSF DocBook Processing Guide for DCE* to learn which tags were used for DCE 1.2.2 and how they were stylistically interpreted by the FOSI. This document is supplied with DCE 1.2.2 and can be found (in PostScript ready-to-print form) at:

dce1.2.2-doc-dir/**src/dce_books_sgml/DocBook_info/DocBook_Processing_Guide_for_DCE.ps**

Details of the FOSIs, which were written by ArborText to order for OSF, can be found in the successive editions of the *Release Notes* which accompanied each of the six FOSI deliveries to OSF. (Note that the FOSIs are described in their *Release Notes* as having been developed for use with the 2.4.1 version of DocBook, although the DCE 1.2.2 books were in fact formatted using the 2.4 version.) Copies of these documents (in PostScript) are located at:

dce1.2.2-doc-dir/**src/dce_books_sgml/DocBook_info/FOSI_relnotes_1.ps**
dce1.2.2-doc-dir/**src/dce_books_sgml/DocBook_info/FOSI_relnotes_2.ps**
dce1.2.2-doc-dir/**src/dce_books_sgml/DocBook_info/FOSI_relnotes_3.ps**
dce1.2.2-doc-dir/**src/dce_books_sgml/DocBook_info/FOSI_relnotes_4.ps**
dce1.2.2-doc-dir/**src/dce_books_sgml/DocBook_info/FOSI_relnotes_5.ps**
dce1.2.2-doc-dir/**src/dce_books_sgml/DocBook_info/FOSI_relnotes_6.ps**

6.4.4 Employment of ADEPT-Specific Formatting Markup

During the SGML conversion of the DCE books it was occasionally found necessary to arrange for precise formatting of certain text and kinds of text (scaling of included PostScript figures could be considered another variety of this kind of activity; see ‘‘Treatment of Figures’’ below). Formatting of this kind, which falls outside the province of SGML itself, was accomplished through various *ADEPT* Publisher-specific commands of the following general form:

```
<?Pub {element | command} [parameter]>
```

For example, the following commands:

```
<?Pub _newline>

<?PubTbl row rht="0.1in" >

<?Pub _cellfont TypeSize="8pt">

<?Pub _font TypeSize="8pt">
<?Pub /_font>
```

were used to insert newlines, adjust row height in tables, change font size in table cells, and change (and unchange) font size in text, respectively. These and other similar commands will be found throughout the SGML source. They will have no effect on non-*ADEPT* SGML software.

6.4.5 Tables

The DCE 1.2.2 documentation uses the CALS DTD for table coding. Note the following important items:

- Various table-processing limitations discovered in *ADEPT* 5.0.2 during the SGML conversion work forced OSF to abandon the original converted table coding, in which tables combining fixed and relative column width settings were often used, just as they had been in the SML originals (relative column settings were the natural and legal CALS translation of the auto-formatted strings coded between ‘‘`T{`’’ and ‘‘`T}`’’ characters in SML).
- SGML coding was done at OSF, for the most part, in text editors such as emacs or vi (the *ADEPT* software was used to check the SGML afterwards); this helped to keep the procedures of code and text manipulation as little cumbrous as possible. Moreover, because the table code (and not only the table code) required extensive hand work, it became important to maintain the SGML source in as ‘‘human-readable’’ form as possible. The *ADEPT* Editor’s method of writing files, in which newlines were largely and randomly eliminated, interfered intolerably with this goal.

6.4.6 Preservation of Original SML Formatting Commands

Most of the original SML formatting markup has been preserved in the converted SGML files. The preserved markup is contained in inert special tagging of the following general form:

```
<?sml-original_SML_coding>
```

For example:

```
<?sml-need 3.6i>

<?sml-space .5>
```

The preserved coding has no SGML effect; its operation in the SGML is equivalent to a no-op.

6.4.7 Miscellaneous

The following items describe additional characteristics of the DCE 1.2.2 documentation source that have changed in comparison to the DCE 1.2.2 (SML) versions.

- The DCE books' copyright pages and the bulk of their preface material was, prior to DCE 1.2.2, drawn from an upper-level directory centrally located (relative to the book directories) at the following position:

  dce1.2.2-doc-dir/**src/dce_books/include**

  and included into each book at format time. In DCE 1.2.2, this copyright information and preface text has instead been propagated as duplicate copies of the same files into each book directory. It is likely that this scheme will be changed in a future version of DCE.

- A preliminary step in the SML-to-SGML conversion of each DCE book involved compiling a comprehensive list of every unique string in the book that was found to have received special SML font coding. The lists were then examined and a careful attempt was made (within the restraints imposed by scheduling) to determine the proper SGML tagging for each marked string. The results of this work were fed into the automated SGML converter, which used them to provide tags for SML-marked text when encountered during the conversion. Inevitably, this bulk method resulted in a certain number of semantically mis-assigned tags. These inaccuracies will be refined out of the source in future versions. In any case, considerable attention was paid to making sure that the appearance of the formatted text was not changed as a result of the conversion. Even though a string may have received an incorrect tag, the tag should still produce the correct formatted appearance for the string.

6.5 Treatment of Figures

Most of the figures in the SML versions of the books were produced from encapsulated PostScript files included in the output at format time. A small number of figures were generated from **pic** code either embedded in the SML source or included from separate files. During the SGML conversion all figures were converted into separate encapsulated PostScript files. It was then found that many figures had to be scaled in order to fit them acceptably into the formatted output (see ‘‘Modifications to the DTD Files’’ above for details on how the scaling mechanism was set up). Typical coding for a scaled figure has the following appearance:

```
<Figure>
<Title>Conventions Used in Authentication Walkthrough Illustrations</Title>
<Graphic Entityref="DCEADG.SECAN.ent.1" Scalefit="1" Reprodep="350" Reprowid="350">
</Graphic>
</Figure>
```

where the substring `Scalefit="1" Reprodep="350" Reprowid="350"` represents the portion of the <**Graphic**> entity definition that accomplishes the scaling. Experiment seemed to show that specifying values of ‘‘400’’ for **Reprodep** and **Reprowid** would usually produce a formatted figure whose size was the same as it would have been if left unscaled. However, this was by no means found always to be so.

Since the figure scaling mechanism used for the DCE 1.2.2 documentation is (inevitably) *ADEPT*-based, you cannot expect the scaling code to have any effect with other SGML software. For each DCE book, any PostScript figure files are held in a single subdirectory named **eps**.

6.6 Use of Vertical Change Bars in DCE 1.2.2 Documentation

Areas in the DCE books changed in DCE 1.2.2 from their DCE 1.2.1 (or earlier) content are indicated in the formatted output by vertical change bars which run continuously along the outer margin of all changed text. The SGML coding prescribed by the OSF FOSI for change bars consists of adding the following attribute definition

```
RevisionFlag="changed"
```

to the opening tag for the text entity that contains the changed text (see the *Open Software Foundation DocBook 2.4.1 FOSI Release Notes*, described above in ‘‘The OSF FOSI’’, for details).

The vertical change bars in the DCE 1.2.2 documentation are themselves a change from the method for showing changed text in earlier versions of DCE documentation, where horizontal lines delimiting the beginning and end of changed or new text were used. The

horizontal change bars used in earlier versions also included a short string of information relating the change to the number of the OT CR in which the change was described or requested. This information no longer accompanies the new-style vertical bars. However, all changes made to the documentation in the course of DCE 1.2.2 were made originally to the SML versions, and were marked in the source with the SML-style change bar coding. The books were then converted into SGML. In the SGML, the original change bar coding was preserved within inert pairs of tags of the following form:

```
<text>

<?og-ChangeStart enh,13609,R1.2.2,Added public key login &amp; fixed headings">

<changed or added text>

<?og-ChangeEnd enh,13609,R1.2.2,Added public key login &amp; fixed headings">

<text>

<?og-ChangeStart enh,13609,R1.2.2,Add PKSS info to public key protocol">

<changed or added text>

<?og-ChangeEnd enh,13609,R1.2.2,Add PKSS info to public key protocol">

<text>
```

(Note the use of SGML entities such as **&** for the ‘‘&’’ character above to represent SGML-reserved characters.) You can thus still search the SGML source for change information by using the `<?og-ChangeStart` and `<?og-ChangeEnd` strings as your search criteria.

Unlike the horizontal bars used in previous DCE documentation, the vertical change bars now employed do not indicate nested coincident changed areas—only one change bar will ever appear alongside text, even though the marked text may have been involved in more than one set of changes. However, this kind of ‘‘change nesting’’ can still be discovered (though with some difficulty) by searching the source for the `<?og-ChangeStart` and `<?og-ChangeEnd` markup as described above. Note that all DCE 1.2.2 change bar-associated tagging, both **RevisionFlag** attributes and preserved SML macros, will be removed from OSF’s sources prior to the beginning of work on future versions of the books.

Finally, note that the *OSF DCE Testing Guide* still employs horizontal change bars. The *OSF DCE/File-Access FVT User’s Guide* (which in fact has not changed since DCE 1.2.1) and the *OSF DCE Release Notes* (the book you are reading) carry no change indications at all.

6.7 Conversion of Reference Pages from SGML to nroff

The **docbook-to-man** batch converter (a public domain tool written by Fred Dalrymple, and contributed by the X Consortium) can be used to convert the DCE 1.2.2 SGML reference page (manpage) source into **nroff** source suitable for installation and use with the **man** command. The program is not distributed with DCE 1.2.2, but it is freely available on the World Wide Web from the following site:

ftp://ftp.ora.com/pub/davenport/README.html

docbook-to-man requires the presence of the public domain **nsgmls** parser, available (with documentation) at the following site:

ftp://ftp.jclark.com/pub/sp

The following sections describe how to build, install and run **docbook-to-man** to convert the DCE 1.2.2 reference pages.

6.7.1 Building and Installing nsgmls

The **docbook-to-man** program uses the **nsgmls** tool to parse the SGML reference page instances and generate the ESIS (Element Structure Information Set) which it then translates (using specifications found in the installed **docbook-to-man.ts** ''transpec'' file) into **nroff** code.

nsgmls was built successfully at OSF on an HP-UX 9.05 platform using the **gcc** version 2.7.1 Gnu C compiler. The following command:

```
gcc -v
```

will output the version number of **gcc**. Either of the two following commands:

```
$ uname -a
HP-UX machine_name A.09.05 A 9000/725 nnnnnnnnnn two-user license
```

or

```
$ uname -rmvs
HP-UX A.09.05 A 9000/725
```

will generate a message describing the version of HP-UX installed (the relevant portions of the returned string in each example are shown in **bold** font).

To build **nsgmls**, locate yourself in the top-level directory (which will be named something like **sp-1.1.1**) and type **make**. To install, edit the **exec_prefix** field in the **Makefile** at the top directory level to get the base install pathname desired, and then type **make install**. Finally, add the installed path, if necessary, to your **PATH**. No other steps were necessary to build and install the program at OSF.

For further instructions, see the **README** file in the top-level directory and the other accompanying documentation.

6.7.2 Building and Installing docbook-to-man

Once you have successfully built and installed **nsgmls**, you can build and install **docbook-to-man**. This program consists mostly of scripts which require nothing more than to be installed, but the **Instant** component of the program must be compiled.

Before building, do the following:

- In the

  DocBook-to-man/cmd/docbook-to-man.sh

  file, edit (if necessary) the following lines at the top of the file:

  ```
  ROOT=/usr/local
  SGMLS=$ROOT/lib/sgml
  DOCBOOK=$SGMLS/Davenport/dtd
  ```

  The **ROOT** variable specifies the root path not only of where **docbook-to-man** will be installed, but also (in combination with **DOCBOOK**) where the DocBook DTD is assumed to be installed. You should edit these lines if either of these installations is at different locations on your system. For example, at OSF these lines were edited to read as follows:

  ```
  ROOT=/u1/BOOKS/local
  SGMLS=/u1/adept/doctypes
  DOCBOOK=$SGMLS/DocBook
  ```

- In the following files:

  DocBook-to-man/Instant/Makefile
  DocBook-to-man/Transpec/Makefile
  DocBook-to-man/Makefile

  you should edit (if necessary) the following line:

  ```
  ROOT =    /usr/local
  ```

  to give **ROOT** the root installation location value desired, if different from the default location already coded.

- If you are installing **docbook-to-man** somewhere other than **/usr/local**, you should define the value of the **DEF_TPT_LIB** variable in either of the

  DocBook-to-man/Instant/Makefile
  DocBook-to-man/Instant/Imakefile

  files (as appropriate), or the

DocBook-to-man/Instant/general.h

file to the *full pathname* of the location at which the ‘‘transpec’’ files (the **.ts** files from which the program takes its translation instructions) will be installed. This pathname will be equivalent to the value of:

```
$(ROOT)/lib/tpt
```

(But note that you can specify a different location, if necessary, for the installed transpec files to **docbook-to-man** by giving the environment variable **TPT_LIB** the desired path value.)

- The DocBook 2.4 DTD (*not* 2.4.1, which was not available when DCE 1.2.2 conversion work began) was used at OSF with the DCE 1.2.2 books, and the books' public identifiers specify this version of the DTD. The **docbook-to-man** program is coded to use DocBook 2.4.1. Therefore, you must change in

  DocBook-to-man/cmd/docbook-to-man.sh

  the ‘‘`2.4.1`’’ in the following lines:

```
#if [ ! -f $PROLOG ]
#then   cat > $PROLOG <<!
#<!DOCTYPE RefEntry PUBLIC "-//Davenport//DTD DocBook V2.4.1//EN" [ <---CHANGE THIS
#<!ENTITY npzwc "">
#]>
#!
#fi
```

  to ‘‘`2.4`’’, as shown below:

```
#if [ ! -f $PROLOG ]
#then   cat > $PROLOG <<!
#<!DOCTYPE RefEntry PUBLIC "-//Davenport//DTD DocBook V2.4//EN" [<-------TO THIS
#<!ENTITY npzwc "">
#]>
#!
#fi
```

After you have completed the above steps, you can build and install **docbook-to-man** by typing **make install** in the

DocBook-to-man/Instant

directory. Finally, add the installed location, if necessary, to your **PATH**.

See the **README** in the

DocBook-to-man

directory for further information.

6.7.3 DTD Files Necessary for Running docbook-to-man

In order to run **docbook-to-man** successfully against the DCE 1.2.2 SGML reference pages, you must of course have a fully-installed DocBook 2.4 DTD on your system. In addition, the following three files:

docbook.cat	Catalog file.
docbook.dcl	SGML declaration file.
docbook.dtd	Document type definition (DTD) file. Contents identical to **DocBook.dtd**.

must be present at the installed DTD location you specified. Mention is made here specifically of these three files, because although they are included with the DCE 1.2.2 distribution in the

dce1.2.2-doc-dir/**src/dce_books_sgml/DocBook**

directory, they are not installed (because they are not needed) by the ArborText *ADEPT* 5.0.2 Document Architect software when the DocBook DTD is compiled. (Other SGML software may behave differently.) Consequently, you will have to manually copy **docbook.cat** and **docbook.dcl** from

dce1.2.2-doc-dir/**src/dce_books_sgml/DocBook**

to your installation area, wherever it is, before attempting to run **docbook-to-man**. As for the **docbook.dtd** file, it is sufficient to simply recopy **DocBook.dtd** as **docbook.dtd** in the DTD install area.

The reason why two identical DTD files with differently-spelled names (**docbook.dtd** and **DocBook.dtd**) are required in the installed DTD area is that the 2.4 DocBook DTD was installed at OSF under the name **DocBook**, while **docbook-to-man** expects to find a file named **docbook.dtd**. Two of the filename strings are defined at the top of the

DocBook-to-man/cmd/docbook-to-man.sh

file as follows:

```
CATALOG=$DOCBOOK/docbook.cat
DECL=$DOCBOOK/docbook.dcl
```

6.7.4 Running docbook-to-man

The **docbook-to-man** program is invoked as follows:

docbook-to-man *refpage_sgml_source_file* > *refpage_nroff_source_file*

The program converts a single SGML instance each run. To convert a group of reference pages, it should be invoked in a script.

Each of the DCE 1.2.2 reference page source files is provided with a ‘‘fragmentary’’ formal public identifier declaration near the top of the file, as follows:

```
                <ODE history header>
#
# EndLog
-->
<!-- Fragment document type declaration subset:
ArborText, Inc., 1988-1993, v.4001
<!DOCTYPE Book PUBLIC "-//Davenport//DTD DocBook V2.4//EN" [
]>
-->
<RefEntry Id="DCEADR.MAN14.rsml.1">
<RefMeta>

                <SGML source>
```

Note that the **DOCTYPE** declaration occurs within a comment; this form of declaration is an ArborText *ADEPT*-specific feature which allows document pieces (such as single reference pages) to be loaded and understood as SGML by the *ADEPT* Editor. Almost all the lower-level DCE 1.2.2 SGML source files have such declarations, as their presence greatly facilitated development work during the SGML conversion. Only the highest-level (**.sgml**) files, one for each book, have a full uncommented **DOCTYPE** declaration.

The commented declarations, however, will not be recognized by **docbook-to-man**. Hence you will have to remove the comment delimiters (and, probably, the ODE source control history header text which appears at the very beginning of most files) before attempting the **docbook-to-man** conversion.

You may note errors such as the following (reported by **nsgmls**) when running the program:

```
unknown declaration type "SGML"
"DOCTYPE" declaration not allowed in DTD subset

character "]" not allowed in declaration subset
document type does not allow element "REFENTRY" here

open elements: REFENTRY REFSECT1[1] REFSECT2[2] ITEMIZEDLIST[1] LISTITEM[1] \
               PARA[1] LITERAL[1] (#PCDATA[1])

open elements: REFENTRY REFSECT1[1] REFSECT2[2] ITEMIZEDLIST[1] LISTITEM[1] \
               PARA[1] LITERAL[1] (#PCDATA[1])

open elements: REFENTRY REFSECT1[1] REFSECT2[2] ITEMIZEDLIST[1] LISTITEM[1] \
               ITEMIZEDLIST[1] LISTITEM[1] PARA[1] LITERAL[1] (#PCDATA[1])

open elements: REFENTRY REFSECT1[1] REFSECT2[2] ITEMIZEDLIST[1] LISTITEM[1] \
```

```
                ITEMIZEDLIST[1] LISTITEM[1] PARA[1] LITERAL[1] (#PCDATA[1])

    no document element
```

None of the above errors, all of which were noted during testing at OSF, interfered with successful conversion.

Finally, note that **nsgmls** expects to find the public SGML entities (such as **iso-amsa.gml**, **iso-amsb.gml**, **iso-amsc.gml**, and so on) installed in the **DocBook** directory. If it does not find them there, you will receive error messages such as the following one:

```
general entity "amp" not defined and no default entity
```

(caused by the ‘‘ampersand’’ entity not being found). To correct this error, copy the entity files (found in *ADEPT* installations in the

your-Adept-install-root/**adept/entities**

directory) into the **DocBook** installation directory you specified to **docbook-to-man**.

6.7.5 Change Bars in Converted Reference Pages

The DCE 1.2.2 reference pages employ the same vertical change bars as were described above for the documentation set as a whole (see ‘‘Use of Vertical Change Bars in DCE 1.2.2 Documentation’’), and are coded identically. However, the change bar coding will be lost in the **docbook-to-man** conversion, as there is no specification in the transpec file for handling the **RevisionFlag** attribute.

6.7.6 DCE 1.2.2 Reference Page Source Paths

Following is a complete list of the locations of all reference page source directories in DCE 1.2.2.

- *OSF DCE Command Reference* reference pages:

dce_books_sgml/command_ref/man1dce
dce_books_sgml/command_ref/man1rpc
dce_books_sgml/command_ref/man5sec
dce_books_sgml/command_ref/man8cds
dce_books_sgml/command_ref/man8dce
dce_books_sgml/command_ref/man8dts
dce_books_sgml/command_ref/man8rpc
dce_books_sgml/command_ref/man8sec

- *OSF DCE Application Development Reference* reference pages:

dce_books_sgml/app_ref/man3dce
dce_books_sgml/app_ref/man3dts
dce_books_sgml/app_ref/man3rpc
dce_books_sgml/app_ref/man3sec
dce_books_sgml/app_ref/man3thr
dce_books_sgml/app_ref/man3xds
dce_books_sgml/app_ref/man3xom
dce_books_sgml/app_ref/man4xds
dce_books_sgml/app_ref/man4xom
dce_books_sgml/app_ref/man5dce

- *OSF DCE DFS Administration Guide and Reference* reference pages:

dce_books_sgml/dfs_admin_gdref/DFS4
dce_books_sgml/dfs_admin_gdref/DFS4A
dce_books_sgml/dfs_admin_gdref/DFS4B
dce_books_sgml/dfs_admin_gdref/DFS4C
dce_books_sgml/dfs_admin_gdref/DFS4D
dce_books_sgml/dfs_admin_gdref/DFS8
dce_books_sgml/dfs_admin_gdref/DFS8A
dce_books_sgml/dfs_admin_gdref/DFS8B
dce_books_sgml/dfs_admin_gdref/DFS8C
dce_books_sgml/dfs_admin_gdref/DFS8D
dce_books_sgml/dfs_admin_gdref/DFS8E
dce_books_sgml/dfs_admin_gdref/DFS8F
dce_books_sgml/dfs_admin_gdref/DFS8G
dce_books_sgml/dfs_admin_gdref/DFS8H
dce_books_sgml/dfs_admin_gdref/DFS8I
dce_books_sgml/dfs_admin_gdref/DFS8J
dce_books_sgml/dfs_admin_gdref/DFS8K
dce_books_sgml/dfs_admin_gdref/DFS8L
dce_books_sgml/dfs_admin_gdref/DFS8M
dce_books_sgml/dfs_admin_gdref/DFS8N
dce_books_sgml/dfs_admin_gdref/DFS8O
dce_books_sgml/dfs_admin_gdref/DFS8P
dce_books_sgml/dfs_admin_gdref/DFS8Q
dce_books_sgml/dfs_admin_gdref/DFS8R
dce_books_sgml/dfs_admin_gdref/DFS8S
dce_books_sgml/dfs_admin_gdref/DFS8T
dce_books_sgml/dfs_admin_gdref/DFS8U
dce_books_sgml/dfs_admin_gdref/DFS8V
dce_books_sgml/dfs_admin_gdref/DFS8W
dce_books_sgml/dfs_admin_gdref/DFS8X
dce_books_sgml/dfs_admin_gdref/DFS8Y
dce_books_sgml/dfs_admin_gdref/DFS8Z

- *OSF DCE GDS Administration Guide and Reference* reference pages:

dce_books_sgml/gds_admin_gdref/MAN8
dce_books_sgml/gds_admin_gdref/MAN8A
dce_books_sgml/gds_admin_gdref/MAN8B

- *OSF DCE DFA Administration Guide and Reference* reference pages:

  dce_books_sgml/dfa_admin_gdref/MAN8/

6.8 Printing the DCE 1.2.2 Documentation

DCE 1.2.2 includes formatted PostScript for all the books in the documentation set. The PostScript files for each book were generated from the book's SGML source located in the

dce1.2.2-doc-dir/**src/dce_books_sgml/***book*/

directories, where *book* signifies the name (**intro**, **command_ref**, and so on) of a specific book. The PostScript was generated by the ArborText **pubformat** utility, as described above (see ''DCE 1.2.2 Documentation Batch Build'' above). The books' PostScript output files can be found in the

dce1.2.2-doc-dir/**src/built_books/***book*/

directories, where *book* signifies (as in the **dce_books_sgml** directory) a specific DCE 1.2.2 book.

Within each **built_books/***book* directory, each book's PostScript output is *duplicated in two sets of files* as follows:

- For each book, a single file with the following name:

  *book_name***_publish_WHOLE_BOOK.ps**

  (where *book_name* is the book's name, a string identical to the name of the directory in **built_books** in which the files are found) contains the PostScript output for *the entire* book. (The string ''**publish**'' identifies the file as having been formatted in OSF ''publish'' style; see ''The OSF FOSI'' above.)

- For each book, in the same directory, a series of files with names of the following form:

  *book_name***_publish-***number***.ps**

  consists of contents identical to that of the ''**_WHOLE_BOOK.ps**'' file described above, but broken up into a continuous series of 30-page segments per file, numbered from ''**01**''.

To print any DCE 1.2.2 book, locate yourself in the book's output directory:

dce1.2.2-doc-dir/**src/built_books/***book*/

and type

lp *book_name***_publish_WHOLE_BOOK.ps**

to send the entire book's output *en bloc* to your favorite (or other) printer, or type

lp *book_name***_publish_***number***.ps**

to print only a single 30-page section of the book. (Wildcards can be used to print series of sections with one command.)

Note that the contents of the books' output files are no longer coterminous with the books' structural pieces (preface, chapter, appendix, and so on), as was true in previous versions of DCE. If you wish to print some specific part of a book, you should first print the book's table of contents, and then attempt to calculate which separate numbered PostScript files correspond to the desired book part. However, because the pagination of books is not continuous, but begins at ''1'' for each chapter, some trial-and-error will be necessary to precisely determine which files correspond to a given stretch of the printed book.

For a complete list of the DCE documentation supplied with DCE 1.2.2, see the ''Documentation Supplied in DCE 1.2.2'' section in Chapter 5 of this document.

6.8.1 Problems You May Encounter Printing the DCE PostScript Files

Some of the DCE 1.2.2 documentation PostScript output files were truncated when they were processed by some printers. The problem, when it occurred, was caused by a memory limitation in the printers.

In the DCE documentation, the graphical contents of diagrams and figures are held in separate PostScript files which are included into the output stream at format time. Some printers will not have sufficient memory to simultaneously handle the definitions in an output file's overall PostScript prolog and the prologs for any PostScript figures that may have been included in the file. In such a case, the file will be printed only up to the page that contains the first figure that has too many definitions, and then will be abruptly terminated (with the remainder of the file's contents being truncated in the output). In many cases, no error message will be issued by the printer when this happens.

6.8.2 Printing the DCE 1.2.2 Release Documentation

The DCE 1.2.2 release documentation consists of the following three books:

- *OSF DCE 1.2.2 Release Notes* (the document you are now reading)
- *OSF DCE 1.2.2 Testing Guide*
- *OSF DCE/File-Access FVT User's Guide*

The *Release Notes* and *Testing Guide* files are located in the

dce1.2.2-doc-dir/**src/release_doc/relnotes/**

dce1.2.2-doc-dir/**src/release_doc/test_gd/**

directories. To print the formatted *OSF DCE Release Notes* or *OSF DCE Testing Guide*, do the following:

1. Change directory to the desired book's top-level directory. For example, to print the *Release Notes*, do the following:

cd *dce1.2.2-doc-dir*/**src/release_doc/relnotes**

2. Enter the following:

lpr *

The output form of both books consists of a series of PostScript files, each of which corresponds to some part (i.e., chapter, appendix, and so on) of the book. Each of these files has a name that begins with a 3-digit string indicating the file's contents' position in the printed book. Thus, printing either book's output files in the natural collating order of their names (as is done in the example immediately preceding) will result in the parts of the complete book issuing from the printer in their correct book order.

6.8.2.1 OSF DCE/File-Access FVT User's Guide

This manual describes how to verify that the Distributed File Access software has been ported and built correctly. The book is located at:

dce1.2.2-doc-dir/**src/release_doc/dfa_fvt_users_gd/**

The book's source files are the following:

fvtpo.rtf	The Microsoft Word source for Chapters 1-9 plus the start of Chapter 10.
dfsstrct.xls	Excel source for the table that ends the book and which constitutes most of Chapter 10.

(Note that the source files for this book are included with DCE 1.2.2 solely for the information of licensees.)

The book's PostScript output files are the following:

fvtpo.ps	PostScript for Chapters 1-9, plus the start of Chapter 10.
dfsstrct.ps	PostScript output of the Excel source file table of Chapter 10.

To print the *FVT User's Guide*, locate yourself in the book's directory and enter the following command:

lpr *.ps

Chapter 7. Distributed File-Access (DFA) Release Notes

This chapter contains information describing how to build and install the Distributed File-Access (hereafter abbreviated as DFA). In addition, several sections at the end of this chapter contain information on how to port DFA to a platform other than the reference platform (AIX) for which it is distributed.

7.1 DFA Documentation

The DFA documentation set is made up of the following two books:

- *OSF DCE/Distributed File-Access User's Guide*

  This book is located at

  dce1.2.2-doc-dir/**src/dce_books_sgml/dfa_users_gd**

  (where *dce1.2.2-doc-dir* is an abbreviation for the pathname

  your-root-dir/**dce/V1.2.2/doc**

  which is where you are assumed to have unloaded the DCE 1.2.2 documentation from the distribution media, as described in Chapter 6).

  This book contains general DFA user's information.

- *OSF DCE/Distributed File-Access Administrator's Guide and Reference*

  This book is located at

  dce1.2.2-doc-dir/**src/dce_books_sgml/dfa_admin_gdref**

  (where *dce1.2.2-doc-dir* has the same meaning as explained above).

  This book contains DFA administration information.

In addition, a third document, the *DFA FVT User's Guide*, contains information about running the functional tests supplied with DFA. This book is located at

dce1.2.2-doc-dir/**src/release_doc/dfa_fvt_users_gd**

(where *dce1.2.2-doc-dir* has the same meaning as explained above).

For further details about these books, see Chapter 5.

README files located in each book's directory describe the structure of the books' subdirectories.

7.2 Pathnames

Throughout the remainder of this chapter, it is assumed that the DCE 1.2.2 archives are found at the following pathname on your system:

your-root-dir/**dce/V1.2.2**

—where *your-root-dir* is the location on your disk where you installed the archives, as described in Chapter 4 of this document. For the sake of brevity and clarity, the pathname above will be abbreviated as

dce1.2.2-root-dir

in the remainder of this chapter. Immediately beneath this pathname will be found several directories corresponding to the separate archives extracted from the DCE 1.2.2 distribution media, as described in Chapter 4. For example, the directory

dce1.2.2-root-dir/**dce**

will contain the extracted **dce** archive, which consists of the DCE 1.2.2 source tree; and the directory

dce1.2.2-root-dir/**doc**

will contain the extracted **doc** archive, which consists of the DCE 1.2.2 documentation set (further described in Chapters 5 and 6).

Thus, the following pathname

dce1.2.2-root-dir/**dce/src/dfam/agent/h**

is equivalent to:

your-root-dir/**dce/V1.2.2/dce/src/dfam/agent/h**

7.3 DFA Software Prerequisites

DFA requires the following versions of NetWare and DCE:

- NetWare Version 3.12.

  CLIB.NLM must be version 3.12g (or later).

- DCE reference platform

7.4 Installing the DFA Agent

This section describes how to install the DFA agent on the DCE 1.2.2 reference platform (IBM RS/6000).

Before beginning the installation, make sure that your system is using the following software:

- AIX version 3.2.5 or later
- OSF DCE reference platform

The DFA agent source code is located under the following subdirectory:

dce1.2.2-root-dir/**dce/src/file/dfam/agent**

To install the DFA agent, do the following:

1. If you have not already done so, create the ODE sandbox, as described in Chapter 4.
2. Attach encryption/decryption modules (see Section 7.16.1 for details).
3. Execute the **build** command (with target **build_all**) to build the following object modules and message catalogs:

TABLE 7-1. Files Built

Filename	Contents
dfaagt	dfaagt load module
setdfakey	setdfakey load module
dfaagt.cat	dfaagt message catalog
setdfakey.cat	setdfakey message catalog

4. Execute the **build** command (with target **install_all**) to install the built files to the installation directory *your-install-dir* (where *your-install-dir* is the directory designated in the **build install_all** command line or in the **TOSTAGE** variable in the

   dce1.2.2-root-dir/**dce/src/dce/Buildconf.exp**

   file). The object modules will be installed under the

   your-install-dir/**bin**

   directory, and the message catalog will be created under the

   your-install-dir/**nls/msg/C**

   directory.

5. Copy the files built in Step 3 and the on-line manual under

   dce1.2.2-root-dir/**dce/src/file/dfam/agent/man/C**

   to the following directories:

TABLE 7-2. Files to Copy

File to be Copied	Destination
dfaagt	**/opt/dcelocal/bin/**
setdfakey	**/opt/dcelocal/bin/**
dfaagt.cat	**/usr/lib/nls/msg/C/**
setdfakey.cat	**/usr/lib/nls/msg/C/**
dfaagt.n	**/usr/share/man/mann/**
setdfakey.n	**/usr/share/man/mann/**
dfaagt.n.Z	**/usr/share/man/catn/**
setdfakey.n.Z	**/usr/share/man/catn/**

6. Change the owner of the above files to ‘‘root’’ and the group of the files to ‘‘bin.’’ Change the file mode of **dfaagt** to ‘‘0550’’, **setdfakey** to ‘‘4550’’, and **dfaagt.cat** and **setdfakey.cat** to ‘‘0444.’’

   Change the owner and group of the above files to ‘‘root’’ and ‘‘bin’’ respectively. Change the file mode of **dfaagt** and **setdfakey** to ‘‘4550’’, and **dfaagt.cat** and **setdfakey.cat** to ‘‘0444.’’

7. Link **dfaagt** and **setdfakey** to the **/usr/bin** directory.

8. In superuser mode, create the **dfam-agent** work directory:

   /opt/dcelocal/var/dfa/adm

9. Create the master key file:

   /opt/dcelocal/var/dfa/dfakey

You have now completed installation of the DFA agent.

7.5 Installing the DFA Gateway

This section describes how to install the DFA gateway on your IBM PC/AT (or compatible) system.

Before beginning the installation, make sure that your system is using the following software:

- MS-DOS version 5.0 or later
- Microsoft Windows version 3.1

And, in addition to the above, either the following combination:

- Novell SDK Vol.4
- Watcom C/C++ version 10.0 compiler

or the following combination:

- Novell SDK CD-ROM, Release 7 (hereafter called Novell SDK Vol.7)
- Watcom C/C++ version 10.5 compiler

Note that *only* the above two combinations are allowed. Other combinations (such as, for example, SDK Vol. 4 and Watcom 10.5 compiler) are not allowed.

To install the DFA gateway, do the following:

1. Install Novell SDK and the Watcom C/C++ compiler in the root directory of C Drive. The default names are used for the directory names.
2. Creating a directory for **make**

   You must create a directory named **GATEWAY** immediately under the root directory of C Drive. C Drive will now have the following directory structure:

```
C:\ ---+--- SDKCD_4\ (Novell SDK Vol.4)
       |
       +--- WATCOM\  (Watcom C/C++ 10.0)
       |
       +--- GATEWAY\  (Gateway)
```

   or (if you are using the second combination of software described above):

```
C:\ ---+--- SDKCD_7\ (Novell SDK Vol.7)
       |
       +--- WATCOM\  (Watcom C/C++ 10.5)
       |
       +--- GATEWAY\  (Gateway)
```

3. Copying files

   Copy all the files and directories under

   dce1.2.2-root-dir/**dce/src/file/dfam/gateway**

   to **C:\GATEWAY**. Note that *the copy must be done in binary mode*.
4. Attach encryption/decryption modules (see Section 7.16.2 for details).
5. Setting an environment variable

   Prior to making the gateway load modules, you need to set a value for the **QMKVER** environment variable. The value to be set varies depending on the module to be made (see the below table for a list of the modules made for the gateway).

TABLE 7-3. Gateway Modules to be Made

Load Module Names	Makefile Names
DFA.NLM	**DFA.MKF**
DFADCE.NLM	**DFADCE.MKF**
DFALIB.NLM	**DFALIB.MKF**
DFAADM.NLM	**DFAADM.MKF**
DFARAS.NLM	**DFARAS.MKF**

The values to set **QMKVER** to are as follows:

- For **DFA.NLM**

  The following table shows the values effective for **QMKVER** when you build **DFA.NLM**. For example:

  SET QMKVER=p50

  if you need to build **DFA.NLM** for 50 users.

TABLE 7-4. QMKVER Values

QMKVER Value	Meaning
p5	**product option (5-user version)**
p10	**product option (10-user version)**
p25	**product option (25-user version)**
p50	**product option (50-user version)**
p100	**product option (100-user version)**
p250	**product option (250-user version)**

- Modules other than **DFA.NLM**

  Set **QMKVER** to ‘‘p’’

6. Do the following to build the executable modules in the current directory:
   1. Change directory (**cd**) to **C:\GATEWAY**.
   2. If you use the Novell SDK Vol. 7 and Watcom C/C++ 10.5 compiler, modify the **C:\GATEWAY\MAKEINIT** file as described below.

      The file *before* modification should appear as follows (lines to be changed and deleted are indicated in the right margin; what the to-be-changed lines should be changed to will be shown in the second example):

```
#
#  makeinit file for makefiles created with QMK386
#
#  Novell's NLM SDK v3.01
```

```
#
#  $(watdrive) is the drive letter where the WATCOM tools are installed                    |
#  $(novdrive) is the drive letter where the NOVELL tools are installed                    |
#                                                                                          | delete
watdrive = C                                                                              |
nlmdrive = C                                                                              |
#                                                                                         |
#   Directories for both the WATCOM and NOVELL tools
#
wat386loc = $(watdrive):\WATCOM\                                                          |
nlm386loc = $(nlmdrive):\SDKCD_4\NLM\                                                     | change 1
nlm386hdr = $(nlm386loc)NOVH
nlm386imp = $(nlm386loc)NOVI
nlm386lib = $(wat386loc)LIB386;$(wat386loc)LIB386
#
#   Define this macro with your copyright statement
#
#copyright = (C) Copyright 199x  NONAME, Inc.  All Rights Reserved.
#
#   Macros that point to various tools we'll need to compile
#
wcc386r = $(wat386loc)BINB\WCC386          # location of 386 real mode compiler           |
wcc386p = $(wat386loc)BIN\WCC386P          # location of 386 protected compiler           | change 2
wcc386 = $(wcc386r)                        # version we want to use

linkr = $(wat386loc)BIN\WLINK              # location of real mode linker                 |
linkp = $(wat386loc)BIN\WLINKP             # location of protected linker                 | change 3
linker = $(linkr)                          # version we want to use

nlmlinkr = $(nlm386loc)NOVBIN\NLMLINKR     # location of real mode Novell linker          | change 4
nlmlinkp = $(nlm386loc)NOVBIN\NLMLINKP     # location of protected Novell linker          | delete
nlmlinker = $(nlmlinkr)                    # version we want to use

nlmpackr = $(nlm386loc)NOVBIN\NLMPACK      # location of real mode NLM compression utility |
nlmpackp = $(nlm386loc)NOVBIN\NLMPACKP     # location of protected NLM compression utility | change 5
nlmpack = $(nlmpackr)                      # location of NLM compression utility

inc_386 = $(nlm386hdr)                     # location of include files
lib_386 = $(nlm386lib)                     # location of libraries files
code_386 = $(wat386loc)BIN\386WCGL.EXE     # location and name of code generator          | change 6
librarian = $(wat386loc)BINB\WLIB          # location of librarian
startup =                                  # in CLIB3S.LIB                                | change 7
#
#  NLM Import Files
#
clibimp = $(nlm386imp)\CLIB.IMP            # the clib import file
tliimp = $(nlm386imp)\TLI.IMP              # the tli import file
aioimp = $(nlm386imp)\AIO.IMP              # the aio import file
socklibimp = $(nlm386imp)\SOCKLIB.IMP      # the socketlib import file
mathlibimp = $(nlm386imp)\MATHLIB.IMP      # the math library import file
dsapiimp = $(nlm386imp)\DSAPI.IMP          # the NDS import file
```

```
nutimp = $(nlm386imp)\NWSNUT.IMP          # the NWSNUT import file
appleimp = $(nlm386imp)\APPLETLK.IMP      # the AppleTalk import file
```

The file *after* modification should appear as follows, where lines that have been changed are indicated in the right margin:

```
#
#  makeinit file for makefiles created with QMK386
#
#  Novell's NLM SDK v3.01
#
#   Directories for both the WATCOM and NOVELL tools
#
wat386loc = c:\WATCOM\                                                    |
nlm386loc = c:\SDKCD7\NLM\                                                | changed 1
nlm386hdr = $(nlm386loc)NOVH
nlm386imp = $(nlm386loc)NOVI
nlm386lib = $(wat386loc)LIB386;$(wat386loc)LIB386\NETWARE
#
#   Define this macro with your copyright statement
#
#copyright = (C) Copyright 199x  NONAME, Inc.  All Rights Reserved.
#
#   Macros that point to various tools we'll need to compile
#
wcc386r = WCC386                          # location of 386 real mode compiler  |
wcc386p = WCC386P                         # location of 386 protected compiler  | changed 2
wcc386 = $(wcc386r)                       # version we want to use

linkr = WLINK                             # location of real mode linker        |
linkp = WLINKP                            # location of protected linker        | changed 3
linker = $(linkr)                         # version we want to use

nlmlinkr = $(nlm386loc)TOOLS\NLMLINKR     # location of real mode Novell linker | changed 4
nlmlinker = $(nlmlinkr)                   # version we want to use

nlmpackr = $(nlm386loc)TOOLS\NLMPACK      # location of real mode NLM compression utility |
nlmpackp = $(nlm386loc)TOOLS\NLMPACKP     # location of protected NLM compression utility | changed 5
nlmpack = $(nlmpackr)                     # location of NLM compression utility

inc_386 = $(nlm386hdr)                    # location of include files
lib_386 = $(nlm386lib)                    # location of libraries files
#code_386 = $(wat386loc)BIN\386WCGL.EXE   # location and name of code generator | changed 6
librarian = $(wat386loc)BINB\WLIB         # location of librarian
#startup =                                # in CLIB3S.LIB                       | changed 7
#
#  NLM Import Files
#
clibimp = $(nlm386imp)\CLIB.IMP           # the clib import file
```

```
tliimp = $(nlm386imp)\TLI.IMP              # the tli import file
aioimp = $(nlm386imp)\AIO.IMP              # the aio import file
socklibimp = $(nlm386imp)\SOCKLIB.IMP      # the socketlib import file
mathlibimp = $(nlm386imp)\MATHLIB.IMP      # the math library import file
dsapiimp = $(nlm386imp)\DSAPI.IMP          # the NDS import file
nutimp = $(nlm386imp)\NWSNUT.IMP           # the NWSNUT import file
appleimp = $(nlm386imp)\APPLETLK.IMP       # the AppleTalk import file
```

3. Issue the following command:

   WMAKE /F *name_of_makefile*

7. Copy the following modules and files to the **SYS:\SYSTEM** directory on the NetWare server:
   - The executable modules built in Step 6 above
   - **C:\GATEWAY\DFASTOP.NCF**

You have now completed installation of the DFA gateway.

7.6 Installing the DFA Client

This section describes how to install the DFA client on your IBM PC/AT (or compatible) system.

Before beginning the installation, make sure that your system is using the following software:

- MS-DOS version 5.0 or later
- Microsoft Windows version 3.1
- Novell SDK Vol.4 or Vol. 7
- Microsoft Visual C++ version 1.0 or later

To install the DFA client, do the following:

1. Install Novell SDK and Microsoft Visual C/C++ in the root directory of C Drive. The default names are used for the directory names.
2. Creating directories for **make**

   You must create two directories named **CLIENT** and **GATEWAY** immediately under the root directory of C Drive.

   C Drive will now have the following directory structure:

```
C:\ ---+--- SDKCD_4\ (Novell SDK Vol.4)
       |
       +--- MSVC\    (Microsoft Visual C++)
       |
       +--- CLIENT\  ---+--- LOGIN\ (LOGIN utilities)
       |                |
       |                +--- FILER\ (FILER utilities)
       |
       +--- GATEWAY\  (Gateway)
```

or (if you are using Novell SDK Vol. 7):

```
C:\ ---+--- SDKCD7\ (Novell SDK Vol.7)
       |
       +--- MSVC\    (Microsoft Visual C++)
       |
       +--- CLIENT\  ---+--- LOGIN\ (LOGIN utilities)
       |                |
       |                +--- FILER\ (FILER utilities)
       |
       +--- GATEWAY\  (Gateway)
```

3. Copy the files and directories under

   dce1.2.2-root-dir/**dce/src/file/dfam/client**

   to **C:\CLIENT**, and those under

   dce1.2.2-root-dir/**dce/src/file/dfam/gateway**

   to **C:\GATEWAY**. Note that *the copy must be done in binary mode*.

4. Attach encryption/decryption modules (See Section 7.16.2 for details).

   All the files shared between the gateway and the client are located under the gateway directory (**GATEWAY**). You need the files under **GATEWAY** even if you only build the client utilities.

5. Building the executable modules

   To build the libraries and executable modules, you must start Microsoft Visual C++ in Windows, setup the directories (from the Options menu, choose Directories), open a project (‘‘**makefile**’’), and start the build (**make**).

   - Setting the directory paths

     Set the directory paths for the SDK include files and libraries.

TABLE 7-5. Setting the Directory Paths

Novell SDK	Directories
Novell SDK Vol.4	include files **C:\SDKCD_4\CLIENT\INCLUDE** libraries **C:\SDKCD_4\CLIENT\DOS\MSC**
Novell SDK Vol.7	include files **C:\SDKCD7\LEGACY\CLIENT\INCLUDE** libraries **C:\SDKCD7\LEGACY\CLIENT\DOS\MSC**

- Building the libraries

  Prior to building the client utilities, you must build the libraries (routines shared among client utilities) listed below:

TABLE 7-6. Libraries

Library	Makefile
DFAM_X.LIB	**C:\CLIENT\LOGIN\DFAM_X.MAK**
CLIFIL_L.LIB	**C:\CLIENT\FILER\CLIFIL_L.MAK**

- Building the executable modules

  After the the libraries are built, build the following utilities:

TABLE 7-7. Utilities

Utility	Makefile
DLOGIN.EXE	**C:\CLIENT\LOGIN\DLOGIN.MAK**
DLOGOUT.EXE	**C:\CLIENT\LOGIN\DLOGOUT.MAK**
DSETPASS.EXE	**C:\CLIENT\LOGIN\DSETPASS.MAK**
DLIST.EXE	**C:\CLIENT\LOGIN\DLIST.MAK**
DGRANT.EXE	**C:\CLIENT\FILER\DGRANT.MAK**
DREVOKE.EXE	**C:\CLIENT\FILER\DREVOKE.MAK**
DREMOVE.EXE	**C:\CLIENT\FILER\DREMOVE.MAK**
DTLIST.EXE	**C:\CLIENT\FILER\DTLIST.MAK**
DRIGHT.EXE	**C:\CLIENT\FILER\DRIGHT.MAK**
DLISTDIR.EXE	**C:\CLIENT\FILER\DLISTDIR.MAK**
DNDIR.EXE	**C:\CLIENT\FILER\DNDIR.MAK**

6. Copying the modules to an executable directory

   Copy the executable modules made in Step 5 above to **SYS:\PUBLIC** on the NetWare server where the DFA gateway is installed (or will be installed).

You have now completed installation of the DFA client.

7.7 Installing the Agent Test Suite

Note: You do not need to install the agent test suite unless you do not plan to run the DFA test suite.

This section describes how to install the DFA agent test suite on the DCE 1.2.2 reference platform (IBM RS/6000).

Before beginning the installation, make sure that your system is using the following software:

- AIX version 3.2.5 or later
- OSF DCE reference platform

The agent test suite source code is located under the following subdirectory:

dce1.2.2-root-dir/**dce/src/test/systest/file/dfam/dfaagtp**

To install the DFA agent test suite, do the following:

1. Building the executable modules

   At

   dce1.2.2-root-dir/**dce/src/test/systest/file/dfam/dfaagtp**

in the ODE sandbox you created (see Chapter 4), execute the **build** command (with target **build_all**) to build the following executable module:

TABLE 7-8. Executable Module Listing

Executable Module
dfaagtp

2. Execute the **build** command (with target **install_all**) to install the module built in Step 1 to the

   your-install-dir/**test/systest/file/dfam/dfaagtp**

   directory (where *your-install-dir* is the directory designated in the **build install_all** command line or in the **TOSTAGE** variable in the

   dce1.2.2-root-dir/**dce/src/dce/Buildconf.exp**

   file).

3. Copying the modules to an executable directory

   The module once built can be executed without moving it to some other location. You may need to modify the configuration file so that it fits the actual environment. See the *DFA FVT User's Guide* for details.

You have now completed installation of the DFA agent test suite.

7.8 Installing the Gateway Test Suite

Note: You do not need to install the gateway test suite unless you do not plan to run the DFA test suite.

This section describes how to install the DFA gateway test suite on your IBM PC/AT (or compatible) system.

Before beginning the installation, make sure that your system is using the following software:

- MS-DOS version 5.0 or later
- Microsoft Windows version 3.1

And, in addition to the above, either the following combination:

- Novell SDK Vol.4
- Watcom C/C++ version 10.0 compiler

or the following combination:

- Novell SDK Vol. 7
- Watcom C/C++ version 10.5 compiler

Note that *only* the above two combinations are allowed. Other combinations (such as, for example, SDK Vol. 4 and Watcom 10.5 compiler) are not allowed.

To install the DFA gateway test suite, do the following:

1. Install Novell SDK and the Watcom C/C++ compiler in the root directory of C Drive. The default names are used for the directory names.
2. Create a directory for **make**

   You must create a directory named **DFAGWTP** immediately under the root of C Drive.
3. Copy the files and directories under

   dce1.2.2-root-dir/**dce/src/test/systest/file/dfam/dfagwtp**

   to **C:\DFAGWTP**, made in Step 2. Note that *the copying must be done in binary mode*. C Drive should now have the following directory structure:

```
C:\ ---+--- SDKCD_4\ (Novell SDK Vol.4)
       |
       +--- WATCOM\  (Watcom C/C++ 10.0)
       |
       +--- DFAGWTP\ ---- TEST\ (Gateway Test Suite test items)
```

   or (if you are using the second combination of software described above):

```
C:\ ---+--- SDKCD7\ (Novell SDK Vol.7)
       |
       +--- WATCOM\  (Watcom C/C++ 10.5)
       |
       +--- DFAGWTP\ ---- TEST\ (Gateway Test Suite test items)
```

4. Setting an environment variable

   Set **QMKVER** to ‘‘p’’.
5. Building the executable modules

   Do the following to build the executable modules:

   a. Change directory (**cd**) to **C:\DFAGWTP**.

   b. If you use the Novell SDK Vol. 7 and Watcom C/C++ 10.5 compiler, modify the **C:\DFAGWTP\MAKEINIT** file as described below.

      The file *before* modification should appear as follows (lines to be changed and deleted are indicated in the right margin; what the to-be-changed lines should be changed to will be shown in the second example):

```
#
#  makeinit file for makefiles created with QMK386
#
```

```
#  Novell's NLM SDK v3.01
#
#   Directories for both the WATCOM and NOVELL tools
#
wat386loc = C:\WATCOM\
nlm386loc = C:\SDKCD_4\NLM\                                          | change 1
nlm386hdr = $(nlm386loc)NOVH
nlm386imp = $(nlm386loc)NOVI
nlm386lib = $(wat386loc)LIB386;$(wat386loc)LIB386
#
#   Define this macro with your copyright statement
#
#copyright = (C) Copyright 199x  NONAME, Inc.  All Rights Reserved.
#
#   Macros that point to various tools we'll need to compile
#
wcc386r = WCC386                        # location of 386 real mode compiler
wcc386p = WCC386P                       # location of 386 protected compiler  | change 2
wcc386 = $(wcc386r)                     # version we want to use

linkr = WLINK                           # location of real mode linker
linkp = WLINKP                          # location of protected linker
linker = $(linkr)                       # version we want to use
nlmlinkr = $(nlm386loc)NOVBIN\NLMLINKR  # location of real mode Novell linker  | change 3
nlmlinkp = $(nlm386loc)NOVBIN\NLMLINKP  # location of protected Novell linker  | delete
nlmlinker = $(nlmlinkr)                 # version we want to use

nlmpackr = $(nlm386loc)NOVBIN\NLMPACK   # location of real mode NLM compression utility  |
nlmpackp = $(nlm386loc)NOVBIN\NLMPACKP  # location of protected NLM compression utility  | change 4
nlmpack = $(nlmpackr)                   # location of NLM compression utility

inc_386 = $(nlm386hdr)                  # location of include files
lib_386 = $(nlm386lib)                  # location of libraries files
code_386 = $(wat386loc)BIN\386WCGL.EXE  # location and name of code generator  | change 5
librarian = $(wat386loc)BINB\WLIB       # location of librarian
#
#  NLM Import Files
#
#startup = $(nlm386imp)\prelude.obj
clibimp = $(nlm386imp)\CLIB.IMP         # the clib import file
tliimp = $(nlm386imp)\TLI.IMP           # the tli import file
aioimp = $(nlm386imp)\AIO.IMP           # the aio import file
socklibimp = $(nlm386imp)\SOCKLIB.IMP   # the socketlib import file
mathlibimp = $(nlm386imp)\MATHLIB.IMP   # the math library import file
dsapiimp = $(nlm386imp)\DSAPI.IMP       # the NDS import file
nutimp = $(nlm386imp)\NWSNUT.IMP        # the NWSNUT import file
appleimp = $(nlm386imp)\APPLETLK.IMP    # the AppleTalk import file
```

The file *after* modification should appear as follows, where lines that have been changed are indicated in the right margin:

```
#
#  makeinit file for makefiles created with QMK386
#
#  Novell's NLM SDK v3.01
#
#   Directories for both the WATCOM and NOVELL tools
#
wat386loc = C:\WATCOM\
nlm386loc = C:\SDKCD7\NLM\                                          | changed 1
nlm386hdr = $(nlm386loc)NOVH
nlm386imp = $(nlm386loc)NOVI
nlm386lib = $(wat386loc)LIB386;$(wat386loc)LIB386\NETWARE
#
#   Define this macro with your copyright statement
#
#copyright = (C) Copyright 199x  NONAME, Inc.  All Rights Reserved.
#
#   Macros that point to various tools we'll need to compile
#
wcc386r = WCC386                        # location of 386 real mode compiler
#wcc386p = WCC386P                      # location of 386 protected compiler  | changed 2
wcc386 = $(wcc386r)                     # version we want to use

linkr = WLINK                           # location of real mode linker
linkp = WLINKP                          # location of protected linker
linker = $(linkr)                       # version we want to use
nlmlinkr = $(nlm386loc)TOOLS\NLMLINKR   # location of real mode Novell linker  | changed 3
nlmlinker = $(nlmlinkr)                 # version we want to use

nlmpackr = $(nlm386loc)TOOLS\NLMPACK    # location of real mode NLM compression utility |
nlmpackp = $(nlm386loc)TOOLS\NLMPACKP   # location of protected NLM compression utility | changed 4
nlmpack = $(nlmpackr)                   # location of NLM compression utility

inc_386 = $(nlm386hdr)                  # location of include files
lib_386 = $(nlm386lib)                  # location of libraries files
#code_386 = $(wat386loc)BIN\386WCGL.EXE # location and name of code generator  | changed 5
librarian = $(wat386loc)BINB\WLIB       # location of librarian
#
#  NLM Import Files
#
#startup = $(nlm386imp)\prelude.obj
clibimp = $(nlm386imp)\CLIB.IMP         # the clib import file
tliimp = $(nlm386imp)\TLI.IMP           # the tli import file
aioimp = $(nlm386imp)\AIO.IMP           # the aio import file
socklibimp = $(nlm386imp)\SOCKLIB.IMP   # the socketlib import file
mathlibimp = $(nlm386imp)\MATHLIB.IMP   # the math library import file
dsapiimp = $(nlm386imp)\DSAPI.IMP       # the NDS import file
nutimp = $(nlm386imp)\NWSNUT.IMP        # the NWSNUT import file
appleimp = $(nlm386imp)\APPLETLK.IMP    # the AppleTalk import file
```

c. Issue the following command to build the modules:

WMAKE /F DFAGWTP.MKF

The executable modules are placed in the current directory. The executable module and the matched makefile are as follows:

TABLE 7-9. Libraries

Executable Module	Makefile
DFAGWTP.NLM	**DFAGWTP.MKF**

6. Copying the modules to an executable directory

   a. Create the following two directories on the NetWare server where the DFA gateway test suite is installed (or to be installed): **DFAGWTP** under **SYS:\SYSTEM**, and **TEST** under **SYS:\SYSTEM\DFAGWTP**.

   b. Copy the following files to the following destinations:

TABLE 7-10. Files to Copy

File	Destination
DFAGWTP.NLM	**SYS:\SYSTEM\DFAGWTP**
DFAGWTP.CFG	**SYS:\SYSTEM\DFAGWTP**
DFAGWTP.FVT	**SYS:\SYSTEM\DFAGWTP**
All the files under C:\DFAGWTP\TEST	**SYS:\SYSTEM\DFAGWTP\TEST**

You have now completed installation of the DFA gateway test suite.

7.9 Installing the Client Test Suite

Note: You do not need to install the client test suite unless you do not plan to run the DFA test suite.

Note: The DCE 1.2.2 **unintegrated** archive contains a number of files for the client test suite which incorporate bug fixes made too late in the DCE 1.2.2 release process to be included in the **src** tree. (The problem is documented in DCE defect report 13440, which can be found in the DCE defects database, located in the

dce1.2.2-root-dir/**project/defect.summary**

directory.) To incorporate the fixed versions, you should copy the contents of the

dce1.2.2-root-dir/**unintegrated/src/test/systest/file/dfam/dfacltp/answer**

directory to:

dce1.2.2-root-dir/**dce/src/test/systest/file/dfam/dfacltp/answer**

before building the DFA client test suite.

This section describes how to install the DFA client test suite on your IBM PC/AT (or compatible) system.

Before beginning the installation, make sure that your system is using the following software:

- MS-DOS version 6.2
- Microsoft Windows version 3.1
- Novell SDK Vol.4 or Vol.7
- Microsoft Visual C++ version 1.0 or later

To install the DFA client test suite, do the following:

1. Install Novell SDK and Microsoft Visual C/C++ in the root directory of C Drive. The default names are used for the directory names.
2. Creating a directory for **make**

   You must create a directory named **DFACLTP** immediately under the root of C Drive.
3. Copy the files and directories under

   dce1.2.2-root-dir/**dce/src/test/systest/file/dfam/dfacltp**

   to the directory **C:\DFACLTP** created in Step 2 above. Note that *the copying must be done in binary mode*. C Drive should now have the following directory structure:

```
C:\ ---+--- SDKCD_4\ (Novell SDK Vol.4)
       |
       +--- MSVC\    (Microsoft Visual C++)
       |
       +--- DFACLTP\ ---+--- TEST\ (Client Test Suite test items)
                        |
                        +--- ANSWER\ (Client Test Suite oracles)
```

   or:

```
C:\ ---+--- SDKCD7\ (Novell SDK Vol.7)
       |
       +--- MSVC\    (Microsoft Visual C++)
       |
       +--- DFACLTP\ ---+--- TEST\ (Client Test Suite test items)
                        |
                        +--- ANSWER\ (Client Test Suite oracles)
```

4. Building the executable modules

   To build the executable modules, you must start Microsoft Visual C++ in Windows, setup the directories (from the Options menu, choose Directories), open a project (‘‘**makefile**’’), and start the build (**make**).

   - Setting the directory paths

     Set the directory paths for the SDK include files and libraries

     TABLE 7-11. Setting the Directory Paths

Novell SDK	Directories
Novell SDK Vol.4	include files **C:\SDKCD_4\CLIENT\INCLUDE** libraries **C:\SDKCD_4\CLIENT\DOS\MSC**
Novell SDK Vol.7	include files **C:\SDKCD7\LEGACY\CLIENT\INCLUDE** libraries **C:\SDKCD7\LEGACY\CLIENT\DOS\MSC**

   - Building the libraries

     The executable modules and their matched makefiles are as follows:

     TABLE 7-12. Executable Modules and Makefiles

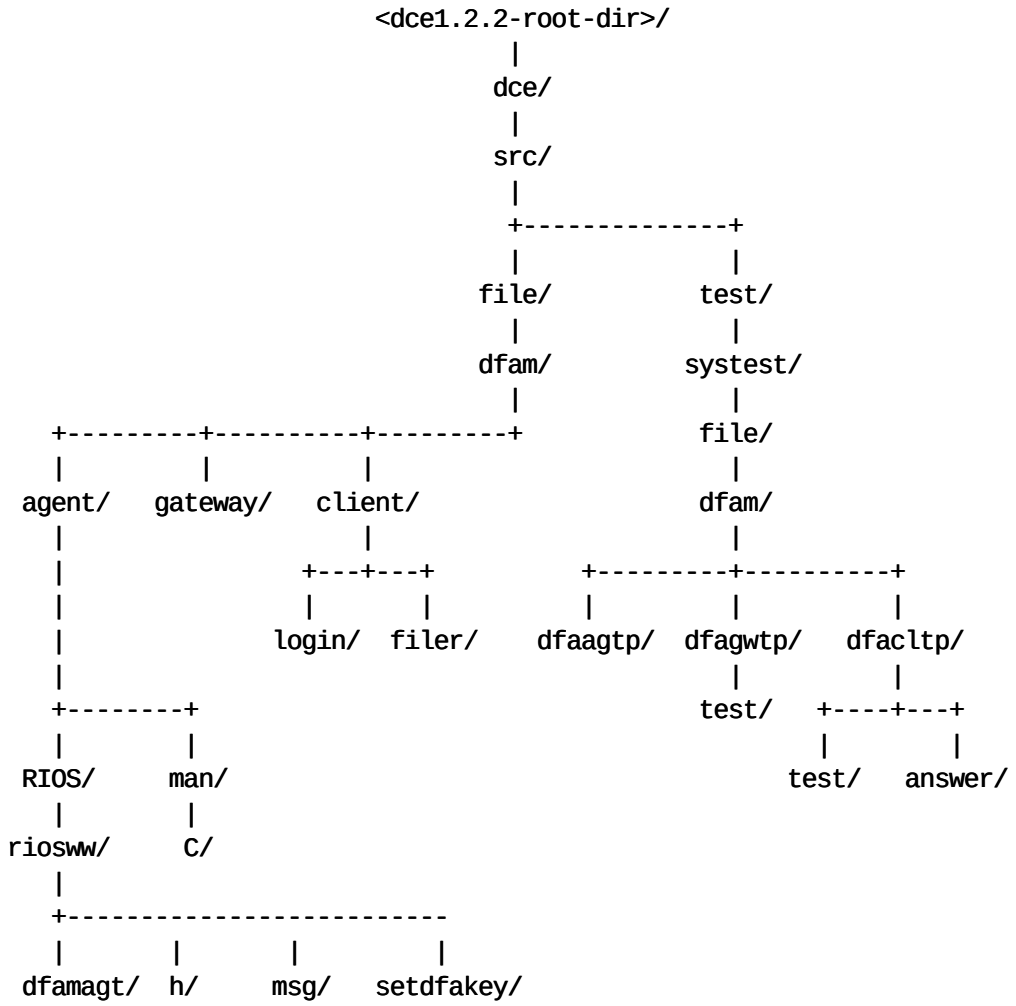
Executable Module	Makefile
DFACLTP.EXE	**C:\DFACLTP\DFACLTP.MAK**
WAIT.EXE	**C:\DFACLTP\WAIT.MAK**
SHELL.EXE	**C:\DFACLTP\SHELL.MAK**

5. The built module can be executed without moving it to another location. Run the client test suite on **C:\DFACLTP**.

You have now completed installation of the DFA client test suite.

7.10 Directory Structures

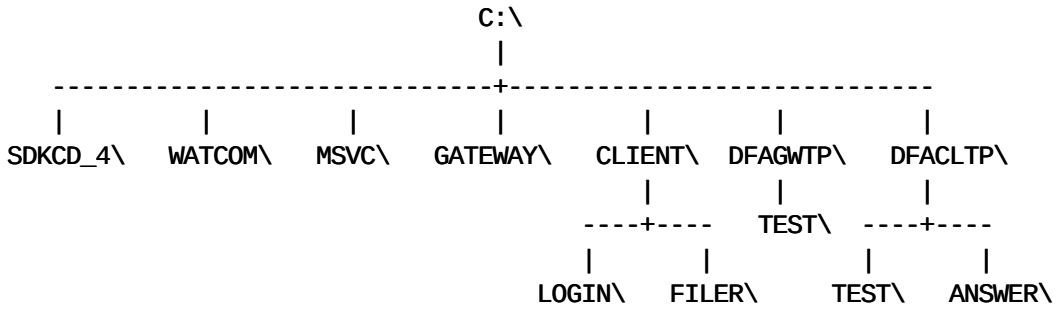
Figure 7-1. DFA Source Code Tree Structure



7.10.1 DFA Client Machine Build Environment Structure

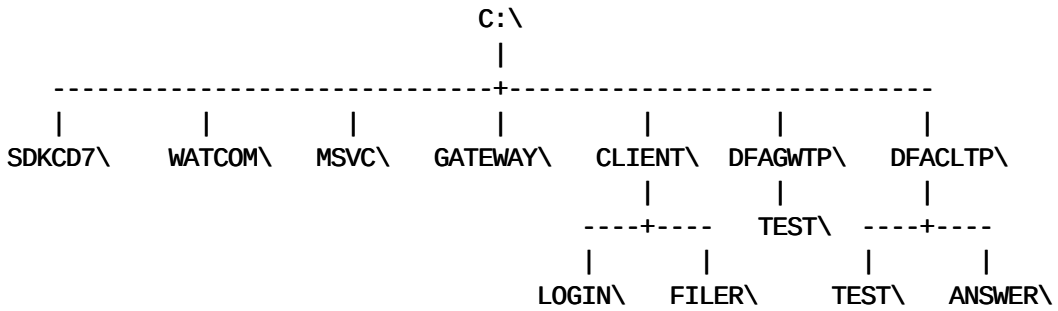
If you build the entire DFA (gateway, client, gateway test suite, and client test suite), the build environment on the client machine should have the following structure (using Novell SDK Vol.4 and Watcom C/C++ 10.0 compiler):

Figure 7-2. DFA Client Machine Directory Structure (with SDK Vol.4 and Watcom 10.0)



If you are using the Novell SDK Vol.7 and Watcom C/C++ 10.5 compiler, the client machine's build environment should have the following structure:

Figure 7-3. DFA Client Machine Directory Structure (with SDK Vol.7 and Watcom 10.5)



7.11 DFA Usage

The following subsections describe how to maximize your DFA power and functionality. A list of ‘‘Frequently Asked Questions’’ concludes this section.

7.11.1 Recommended Practices

The following sections describe various recommended uses for DFA.

7.11.1.1 Company-wide Document Sharing

DFS is one of the most practical solutions for document sharing among physically separated office locations when security and performance are serious considerations. On the other hand, NetWare, a PC-based network operating system, is widely used as an office-wide network system. DFA can combine a world-wide network with the office-wide system, and allow NetWare users to access files located at different office locations without having knowledge or experience of DFS.

7.11.1.2 Secure File Sharing with UNIX Server WS and PC

When you build a system where PCs use resources stored on UNIX server machines, you must furnish a means to access the files on the UNIX machines from the PCs. One of the key issues for such a system is user authentication of these PC-to-UNIX accesses. DFA enables PC users to access UNIX machines while retaining their access rights.

7.11.1.3 Interchanging the Applications on UNIX and PCs

The strength of UNIX applications lies in the manipulation of large masses of data, while PCs are good at, for example, formatting presentation slides. DFA can combine the strengths of both kinds of application, allowing PC users to pretty-print data processed by UNIX machines.

7.11.1.4 Centralization of Physically Distributed Office Data

If office locations are physically scattered, and the distributed data needs to be centrally collected and processed, DFA can easily be used to build a system where applications do not have to bother about data location.

7.11.2 Non-Recommended Uses

The following sections describe various uses that are *not* recommended for DFA.

7.11.2.1 Sharing Database Files

Since DFA copies the entire target file from the DFS client to the NetWare server, it suffers from processing overhead when used to access small portions of huge files. In addition, there is no record locking mechanism between users on different NetWare servers. Thus DFA cannot be used to share database files that require such a mechanism.

7.11.2.2 File Sharing in a Small PC Network

In a small PC network system that shares files only among PCs, using NetWare alone is more effective than accessing the files via DFA.

7.11.3 DFA Characteristics

The following sections describe various characteristics of DFA operation that may be of interest.

7.11.3.1 Performance

Because DFA employs a gateway-based mechanism, it is less effective than direct access to a NetWare server and DFS. DFA overhead is affected by network speed, machine performance, and the options set for the gateway.

7.11.3.2 Security

NetWare users and DCE users are mapped at a DFA gateway. DFA handles user authentication to DCE by using the DCE passwords stored on the NetWare server. Since these passwords can be accessed by any SUPERVISOR-equivalent users, the NetWare server on which the DFA gateway is located must be situated in a logically and physically secure place.

7.11.3.3 Compatibility

DFA allows DOS/Windows applications to access DFS with the same operations as they would use to handle a NetWare volume. Although almost all DOS/Windows applications can access DFS, there are a few applications that cannot handle DFS, and some functions are incompatible. See Section 7.14, ‘‘DFA Restrictions’’, for details.

7.12 Tips for Better DFA Performance

The following sections describe various ways in which you can improve DFA performance.

7.12.1 Network and DFS Configuration

The following recommendations apply to various general characteristics of the network and Distributed File Service (DFS) configuration in which DFA is used:

- DFS client and gateway

  To minimize the overhead of a gateway-based method, a high-speed LAN is recommended for connecting a DFS client with a NetWare server.

- Wide area connection

  To take advantage of DFS caching, we recommend using DFS for the wide area connection using lower speed lines.

- Replica

  We strongly recommend using the DFS replica for sharing read-only files over the wide area connection.

- Mount point

  To lessen network load, the DFS mount point defined in the gateway should be limited to a minimum set.

7.12.2 Tuning the NetWare Server

The following sections describe various ways in which you can fine-tune a NetWare server's DFA performance.

7.12.2.1 Estimating Volume Size

The gateway volume holds a temporary copy of every opened file. Therefore the gateway volume must have enough space to hold copies for all files open at any given time. In addition, it is recommended that the gateway volume have a work area of the same size as that reserved for the open files. Thus the recommended volume size is calculated by the following formula:

```
Volume size = (maximum number of login users) *
              (average number of the opened files per user) *
              (average file size) * 2
```

You will need to adjust this formula in accordance with the actual system used.

7.12.2.2 Directory Depth

The default value for the directory depth on a NetWare server is 25. You can change this (to any value between 10 and 100) by using the **SET** utility in **STARTUP.NCF**. If the directory depth under your DFS mount point is greater than 25, you will need to set a deeper value. As a rule of thumb, the value you should set will be equal to the directory depth under the DFS mount point, *plus* the estimated maximum directory level built under the deepest directory.

You should be aware that some DOS/Windows applications set their own limitations on the directory depth and pathname length. In such cases, you can use **MAP ROOT** to shorten the appearance of the directory depth.

7.12.2.3 Additional Memory

Besides the memory required to run a NetWare server, you need to allocate extra memory to run the gateway. The amount of additional memory required is given by the following formula (assuming a typical gateway installation):

```
Extra memory (KB)= (500 + 30
                    * (number of simultaneous login users)
                    + 30
                    * (number of simultaneously opened files))
```

7.12.3 Tuning the DFS Client

The following sections describe various ways in which you can fine-tune a DFS client's DFA performance.

7.12.3.1 Data Cache

In order to allow large numbers of files to be cached simultaneously, we recommend that the cache size be greater than the total size of the frequently accessed files.

7.12.3.2 Status Cache and Name Cache

We recommend that the size of the status cache and name cache be greater than the total number of files and directories under the DFS mount point.

7.12.3.3 Additional Memory

In addition to the memory used to run a DFS client, the DFS client needs to allocate extra memory for the agent programs. As a rule of thumb, the size of the extra memory is given by the following formula (the size depends on the installed DCE):

```
Extra memory (MB)= (30 + 10
                    * (number of connected gateways)
                    + 5
                    * (number of simultaneous login users))
```

7.12.4 Optional Gateway Parameters

You can set three parameters to tune the gateway:

- Copyback delay parameter
- File-sparsing delay parameter
- Directory synchronization parameter

These parameters can be set independently; they are described in more detail below.

7.12.4.1 Copyback Delay

When a user closes a file, the gateway stores the file back to DFS. This is called ‘‘copyback.’’ The copyback delay parameter specifies the waiting time for the copyback (i.e., the period during which the file remains on the gateway volume). If you reopen the file before it is copied back to DFS, the copy remaining on the gateway volume will be reused instead of the original being fetched from DFS. The copyback delay parameter is useful in reducing overhead in particular for applications that repeatedly open and close files. We recommend that the copyback delay time be set between several seconds and 10 seconds.

7.12.4.2 File-Sparsing Delay

When the copyback is completed, the file is ‘‘sparsed’’, to reduce the disk space it occupies, until it is opened again. The file-sparsing delay parameter specifies the amount of time to wait before sparsing the file. This wait period will reduce file access overhead, because if the file is accessed within the waiting period, the file on the gateway volume will be reused instead of the file being fetched again from DFS. However, keep in mind that the the counterpart (‘‘original’’) file on DFS is locked during the sparse delay period, during which other gateways will not be able to access the file. We recommend that the value of this parameter be between one and several minutes.

7.12.4.3 Directory Synchronization

DFA periodically copies the DFS directory structure to the gateway volume to maintain consistency between the two. This is called ‘‘directory synchronization.’’ The directory synchronization parameter specifies the interval of time to elapse between synchronizations. Since all files and directories under the DFS mount point are checked during directory synchronization, overly frequent synchronization will create unnecessary loading at the DFS client. The optimum value for this parameter depends on

the frequency of file creations/deletions: set several minutes if files are frequently created and deleted, several hours if files are less frequently created and deleted.

7.13 Tips for Easier Usage

The following sections describe various ways in which you can make DFA use more convenient.

7.13.1 Setting the NetWare Environment

- Automatic login via Login script

  You can automatically enter DCE if you set a DCE login command in the NetWare login script.

- Timezone of the NetWare server

  If you fail to properly set the NetWare timezone, DFA cannot read the make time of the DFS files.

7.13.2 Optional Gateway Parameters

- Time synchronization

  You can select whether or not to adjust the NetWare clock to the time on the DFS client at every login. We recommend that this parameter be set, in order to guarantee that DFA properly reads the make time of the DFS files.

7.14 DFA Restrictions

The following sections describe various restrictions on DFA operation.

7.14.1 Unusable DOS Commands

Currently the following DOS commands have been reported as unusable via DFA:

- **ATTRIB**

If used, an improper result may be returned, or a system malfunction may occur.

7.14.2 Unusable NetWare Commands

Do not use the NetWare commands supplied for file manipulation. If you do, there may be adverse system consequences.

7.14.3 Unusable Functionality

The following functionality is currently not usable via DFA:

- DOS/Windows

  Currently the following DOS/Windows APIs are reported as unusable via DFA:

  - Record locking among users on different NetWare servers

    Record locking is effective within the same gateway.

  - Operations related to file attribute

    An improper result may occur.

  - Salvage of the gateway volume

    The salvage function is not usable, or the salvage may be improperly performed.

- NetWare

  Currently the following NetWare functions are reported as unusable via DFA:

  - Account

    Since a file is temporarily stored on the NetWare server, the accounting information may not be accurate.

7.14.4 Incompatibility with NetWare

Complete DFA compatibility is not guaranteed for the following operations:

- Deletion of a file being opened

  A file being opened by a gateway can be deleted from other gateways.

7.14.5 DFS-Related Restrictions

The following DFS restrictions are also reflected in DFA:

- Deletion of replicas

  When replica files are deleted, the maximum number of the replica files you can delete at a time is limited (see DFS documents for details).

7.14.6 NetWare PC Client Configurations Supported by DFA

DFA supports VLM clients in Bindery Emulation mode, but only with NetWare 3.12.

DFA supports Windows 95 clients running Novell Client16 (VLM).

DFA support for Windows 95 clients running Novell Client32 (VLM) has not yet been tested, but it is expected that this will work.

DFA support for Windows NT clients running Novell Client for NT has not yet been tested, but it is expected that this will work.

DFA does not support NETX clients on MS-DOS or MS-Windows 3.1x.

DFA does not support VLM clients in NDS mode.

DFA does not support Macintosh computers running MacIPX or MacNDS.

The following client software versions are capable of interoperation:

DFA Agent	AIX version 3.2.5 (or later) DCE version 1.0.3 (or later)
DFA Gateway	MS-DOS 5.0 (or later) NetWare Server 3.12
DFA Client	MS-DOS 5.0 (or later) NetWare Client 3.12

No enhancement of the DFA functions or the scope of the targeting software/hardware configurations on DCE 1.2.2 is planned for DCE 1.2.2. Only bug fixing will be done on DFA for DCE 1.2.2.

7.15 Frequently Asked DFA Questions

Following are some frequently-asked DFA questions (and answers):

Q1 Can I use the files on other NetWare servers?

A1 No, you cannot. DFA allows you to share only the files in DFS; i.e., the files to be shared must be placed in DFS.

Q2 Can I share a printer?

A2 No, you cannot. DFA does not provide a printer sharing function.

Q3 How do the number of gateway users and the number of NetWare users affect the maximum number of DFA users?

A3 Whichever of the two (gateway users or NetWare users) is the smaller will be the maximum number of DFA users.

Q4 What is ‘‘copyback’’?

A4 A gateway volume temporarily keeps copies of all currently-opened files. When a user closes such files, gateway copies the files back to DFS. This is called ‘‘copyback.’’

Q5 Is DFA compatible with NetWare 3.11?

A5 No, DFA is not guaranteed to run on NetWare 3.11.

Q6 How does DFA enforce security?

A6 The DCE passwords stored in the NetWare server are accessible only in **SUPERVISOR** mode (assuming that the server is logically and physically secure). When the gateway sends a DCE password to the agent on the DFS client, the password is encrypted using a one-time key. Whenever a password is encrypted, the encryption key is newly generated from a random number.

7.16 Distributed File-Access (DFA) Porting Information

The Distributed File-Access (hereafter abbreviated as DFA) software package distributed by OSF with DCE 1.2.2 does not contain any modules for encrypting and decrypting user IDs and passwords. The following sections describe how you should attach your encryption/decryption modules to DFA.

7.16.1 DFA Agent

The following sections describe how to add encryption/decryption modules to the DFA agent.

7.16.1.1 Encryption/Decryption Modules

The DFA agent's login function allows you to have multiple encryption/decryption modules, and thus use different sets of encryption/decryption methods to encode/decode IDs and passwords. To administer the encryption/decryption methods, DFA uses:

- An *encryption protocol number* to identify each encryption/decryption method
- A table of pointers, by means of which the encryption/decryption functions are accessed

DFA determines the encryption/decryption method to be used by referring to the encryption protocol number (see Section 7.16.3, ''Encryption Protocol Numbers'', below) before login.

7.16.1.1.1 Prerequisites

In the following sections, it is assumed that the following things are true:

- The DFA agent files reside under the following directories on the AIX platform:
  - Headers

    dce1.2.2-root-dir/**dce/src/file/dfam/agent/RIOS/riosww/h**
  - Source code (**dfaagt**)

    dce1.2.2-root-dir/**dce/src/file/dfam/agent/RIOS/riosww/dfamagt**
  - Source code (**setdfakey**)

    dce1.2.2-root-dir/**dce/src/file/dfam/agent/RIOS/riosww/setdfakey**

- You have built the executable files in the ODE environment.

7.16.1.1.2 Changes to agt_login.c

For the AIX reference platform, the table (**crypt_tbl**) which contains the encryption/decryption module functions is found in:

dce1.2.2-root-dir/**dce/src/file/dfam/agent/RIOS/riosww/dfamagt/agt_login.c**

To add a new encryption/decryption module, you must add a new pair of adaptor functions (which handle the differences between the function interfaces) to the table. The registration of the adaptor functions in the table must be in ascending order of the encryption protocol numbers.

Users must furnish the adaptor functions in the following formats. The encryption function has the following fully-defined prototype:

```
void cryptfunc(char *data, unsigned long datalen, unsigned long mkey[]);
```

—where *cryptfunc* is an arbitrary function name. This function uses **mkey** as an encryption key, and encodes the data pointed to by **data**. **datalen** indicates the number of bytes to be encrypted, and the encrypted data will be written to the area pointed to by **data**, overwriting the original data.

The fully-defined prototype for the decryption function is as follows:

```
void decryptfunc(char *data, unsigned long datalen, unsigned long mkey[]);
```

— where *decryptfunc* is an arbitrary function name. This function uses **mkey** as a decryption key, and decodes the data pointed to by **data**. **datalen** indicates the number of bytes to be decrypted, and the decrypted data will be written to the area pointed to by **data**, overwriting the original (encrypted) data.

The execution steps of an adaptor function are as follows:

1. Convert the input values (**mkey**, **data**, **datalen**) so that the provided encryption/decryption functions can handle them.
2. Execute the provided encryption/decryption functions.
3. Convert the results of Step 2 into the **data** format.

The data structure, which should not have to be changed, is defined (for the AIX platform) in:

dce1.2.2-root-dir/**dce/src/file/dfam/agent/RIOS/riosww/h/agt_crypt.h**

The definition is as follows:

```
struct d_crypt_tbl{
    int     crypttype;    /* the encryption protocol number */
    void    (*crypt)();   /* the pointer to the encryption function */
    void    (*decrypt)(); /* the pointer to the decryption function */
}crypt_tbl[n];
```

—where **n** is the number of crypto types available. (For further details on **crypttype**, the encryption protocal number, see Section 7.16.3, ‘‘Encryption Protocol Numbers’’, below.)

As mentioned above, the encryption/decryption module table format is defined (for the AIX platform) in:

dce1.2.2-root-dir/**dce/src/file/dfam/agent/RIOS/riosww/dfamagt/agt_login.c**

You must re-define this table to add encryption/decryption modules to DFA. The table has the scheme shown in the following pseudo C fragment:

```
struct d_crypt_tbl crypt_tbl[] = {
    {0x00000000, DumFunc, DumFunc},

    >>> Add an adaptor function here (the encryption protocol
    >>> numbers must be arranged in ascending order)

    {0xffffffff, NULL, NULL}
};
```

For example, if you were to add ‘‘**Multi0**’’ as a new encryption module, you would define the table format as follows:

```
struct d_crypt_tbl crypt_tbl[] = {
    {0x00000000, DumFunc, DumFunc},
    {0x00000002, EncryptbyMulti0, DecryptbyMulti0},
    {0xffffffff, NULL, NULL}
};
```

where:

0x00000002	is the encryption protocol number
EncryptbyMulti0	is the encryption adaptor function
DecryptbyMulti0	is the decryption adaptor function

7.16.1.1.3 Define Prototype

The following modules need the prototype definition to use the adaptor functions provided by the users:

- *dce1.2.2-root-dir*/**dce/src/file/dfam/agent/RIOS/riosww/dfamagt/agt_login.c**

  Needs prototype definition for encryption and decryption functions.

- *dce1.2.2-root-dir*/**dce/src/file/dfam/agent/RIOS/riosww/setdfakey/agt_sdk.c**

  Needs prototype definition for encryption function.

7.16.1.1.4 Changes to Numeric Data Format of Encryption Key

In order for DFA to share the same encryption key between NetWare (DFA gateway) and UNIX (DFA agent), you may need to make some adjustments in the numeric data representation (e.g., big-endian/little-endian).

7.16.1.2 Error Numbers

DFA was developed on Hitachi's HI-UX, and you may need to modify the DFA error numbers when porting DFA to a different platform. DFA, however, can handle the IBM AIX error numbers; you do not need to make any modifications if you are porting DFA to an IBM AIX system or any other system that uses the same error numbers.

Errors that are not supported by HI-UX but are supported by your target platform must be mapped to other appropriate HI-UX error numbers.

When making changes to error number mapping, the files you will need to modify are:

- *dce1.2.2-root-dir*/**dce/src/file/dfam/agent/RIOS/riosww/dfamagt/errno.c**

  This is the module which converts error numbers.

- *dce1.2.2-root-dir*/**dce/src/file/dfam/agent/RIOS/riosww/h/hi_err.h**

  This header file contains the HI-UX error numbers.

The following code fragment shows how you modify **errno.c**:

```
/* Change ENOCONNECT into HI_ECONNREFUSED   */

case ENOCONNECT:
    *errno_OUT = HI_ECONNREFUSED;
    return TRUE;
```

7.16.2 Gateway and Client

The following sections describe how to install an encryption module in the DFA gateway and client.

7.16.2.1 Prerequisites

The following sections assume that the DFA gateway and the DFA client have been properly installed as described earlier in this chapter. Hereafter, ‘‘gateway directory’’ signifies the directory where the DFA gateway was installed, and ‘‘client directory’’ signifies the directory where the DFA client resides.

For the details of DFA installation, refer to Sections 7.4 - 7.6 (for the agent, gateway, and client), and 7.7 - 7.9 (for the tests) above.

7.16.2.2 How to Add the Encryption Module

Since the DFA gateway and the DFA client share the same encryption module, and the DFA client uses the encryption source code in the gateway directory, you only need to modify the encryption source code in the DFA gateway.

The steps to install the encryption module are as follows:

(i) Add Encryption/Decryption Functions

Since the DFA source code shipped from OSF does not contain encryption/decryption functions, you must add encryption/decryption modules. Among the functions to be added, the ones used by the adaptor function and the master key generation function (to be mentioned below) need a prototype definition.

If you want to make the encryption/decryption module an independent and differently-named module, you must add the module name to the proper makefile.

For example, The following steps show how you should add a file named *newfile*.**c** (containing a new encryption/decryption module you wish to add) to the DFA gateway and DFA client makefiles:

- Gateway makefile

  1. Use a text editor to open **DFA.MKF** in the gateway directory.
  2. Add the following line:

     ```
     objlst = $+$(objlst)$-newfile.obj
     ```

     between the following two lines:

     ```
     objlst = $+$(objlst)$-GDIRSYNC.obj
     ```

     ```
     objlst = $+$(objlst)$-$(startup)
     ```

- Client makefile

  1. Start Microsoft Visual C++.
  2. Select ‘‘Open...’’ from the ‘‘Project Menu’’, and open **dsetpass.mak** in the client directory.
  3. Select ‘‘Edit...’’ from the ‘‘Project Menu’’, and display the edit screen. This screen shows the files to be compiled and linked.
  4. If you want to append a file:

     (a): Set a directory name to ‘‘Directories:’’ to specify the directory where the file to be added resides (this operation will list the source files registered in the directory designated by ‘‘File Name:’’); then:

     (b): Move the cursor to the target file name, and double-click the mouse. The selected file will be added to ‘‘Files in Project.’’
  5. If you want to delete a file, move the cursor to the target file in ‘‘Files in Project,’’ then press the ‘‘Delete’’ button.
  6. When you have completed editing, press the ‘‘Close’’ button to exit the text editing. If you want to undo the entire operation, press the ‘‘Cancel’’ button.

(ii) Modify the Encryption/Decryption Function

The adaptor routines for encryption/decryption are found in

dce1.2.2-root-dir/**dce/src/file/dfam/gateway/gencrypt.c**

The functions have the following names:

fnSetEncryptData( ) Adaptor function (encryption)

fnGetDecryptData( ) Adaptor function (decryption)

In DFA as supplied with DCE 1.2.2, which does not have the encryption/decryption modules, the adaptor functions receive and send a plaintext file.

You will need to add the process to invoke the encryption/decryption routines to each adaptor function.

(a) Description of **fnSetEncryptData( )**

The source for this function is found in:

dce1.2.2-root-dir/**dce/src/file/dfam/gateway/gencrypt.c**

fnSetEncryptData( ) encrypts the designated character string, and returns (in memory pointed to by **pbyEdata**) an encrypted character string. There is no return code.

TABLE 7-13. Parameters for fnSetEncryptData()

Name	Type	In/Out	Description
auiSkey	**unsigned long ***	in	encryption key
pbyData	**unsigned char ***	in	data to be encrypted (plaintext)
iLen	**int**	in	data length (bytes)
pbyEdata	**unsigned char ***	in/out	pointer to the encrypted data

Note: The caller of this routine must have enough data space to store the encrypted data.

If you add new processing to **fnSetEncryptData( )**, you should do so according to the following scheme:

```
#include <string.h>

void fnSetEncryptData( auiSkey, pbyData, iLen, pbyEdata )
unsigned long   auiSkey[2];
unsigned char   *pbyData;
int     iLen;
unsigned char   *pbyEdata;
{

 auiSkey[0] = auiSkey[0];              <---Delete these lines
 memcpy( pbyEdata, pbyData, iLen );    <---

       >>>
       >>> Add statements here
       >>>

    return;
}
```

(b) Description of **fnGetDecryptData( )**

The source for this function is found at:

dce1.2.2-root-dir/**dce/src/file/dfam/gateway/gencrypt.c**

fnGetDecryptData( ) decrypts the encrypted string, and returns (in memory pointed to by **pbyPdata**) the plaintext data. There is no return code.

TABLE 7-14. Parameters for fnGetDecryptData()

Name	Type	In/Out	Description
auiSkey	**unsigned long ***	in	decryption key
pbyData	**unsigned char ***	in	data to be decrypted
iLength	**int**	in	length of the decrypted text (bytes)
pbyPdata	**unsigned char ***	i/o	pointer to the decrypted data

Note: The caller of this routine should calculate the length of the decrypted data.

If you add new statements to **fnGetDecryptData( )**, you should do so according to the following scheme:

```
#include <string.h>

void fnGetDecryptData( auiSkey, pbyData, iLength, pbyPdata )
unsigned long   auiSkey[2];
unsigned char   *pbyData;
int     iLength;
unsigned char   *pbyPdata;
{

 auiSkey[0] = auiSkey[0];              <---Delete these lines
 memcpy( pbyEdata, pbyData, iLength ); <---

       >>>
       >>> Add new statements here
       >>>

 return;
}
```

(iii) Add Encryption/Decryption Modules to the Master Key Generation function

The DFA administration utility (**DFAADM.NLM**) uses the master key generation function (defined in **uadmagt.c**; see below for the location of this file), which synchronizes the master key between the DFA gateway and the DFA agent. Since DFA as supplied with DCE 1.2.2 simply returns the fixed value, you will need to add to this file calls to the encryption/decryption functions. How to do this is explained below.

(a) Description of **MakeMasterKey( )**

The source for this function is found at:

dce1.2.2-root-dir/**dce/src/file/dfam/gateway/uadmagt.c**

MakeMasterKey( ) converts the designated character string into an encryption key, and returns the key (in memory pointed to by

aszMkeystr) as a master key character string. There is no return code.

TABLE 7-15. Parameters for MakeMasterKey()

Name	Type	In/Out	Description
pszKeystr	**BYTE ***	in	character string (1 - 8 characters) to be used as a seed for the master key
aszMkeystr	**BYTE ***	in/out	pointer to the character string (the master key in hexadecimal format). The first and the last 4 bytes are spaces

If you add new statements to **MakeMasterKey( )**, you should do so according to the scheme shown in the following pseudo C code fragment:

```
#include <stdio.h>

void MakeMasterKey( pszKeystr, pszMkeystr )
BYTE *pszKeystr;
BYTE *pszMkeystr;
{

      memcpy( pbyEdata, pbyData, iLength );      <---
      pszKeystr = pszKeystr;                     <---Delete these lines
      sprintf(pszMkeystr, "deadbeef deadbeef"); <---

         >>>
         >>>  Add new statements here
         >>>

    return;
}
```

(iv) How to Add/Change the Encryption Protocol Number

In DFA as supplied with DCE 1.2.2, the encryption protocol number is set to the dummy value 0 (see Section 7.16.3, ‘‘Encryption Protocol Numbers’’, below for details). You will need to set an appropriate number corresponding to the encryption scheme you will employ.

The protocol numbers as currently assigned by Hitachi are defined in:

dce1.2.2-root-dir/**dce/src/file/dfam/gateway/dfamdef.h**

The current definitions are as follows:

```
#define CIPHER_NONE            0
#define CIPHER_MULTI2          1
#define CIPHER_MULTI0          2
```

```
#define CIPHER_OSF              3
```

(See Section 7.16.3, ‘‘Encryption Protocol Numbers’’ below for the meanings of the numbers.)

iCipherType is defined in the structure **Gagent_t** in the header file:

dce1.2.2-root-dir/**dce/src/file/dfam/gateway/dfamstr.h**

Set an appropriate value for **iCipherType** in the **AddToGAgent_t( )** function in:

dce1.2.2-root-dir/**dce/src/file/dfam/gateway/gmaitbl.c**

You should make your changes in accordance with the pseudo C code fragment shown below:

```
struct GAgent_t * AddToGAgent_t()
     {
         struct GAgent_t * pTblp;
         struct GAgent_t * pWorkp;

        GW_tsh_functrace(FUNC_ADDTOAGENT, GW_FUNC_START, NULL);

        pTblp = (struct GAgent_t *)GW_malloc( sizeof(struct GAgent_t) );
                    :
         strcpy(pTblp->aszIcat, AGT_ICAT);
         pTblp -> pUshirop = (struct GAgent_t *)0;


                                                <---
         pTblp -> iCipherType = CIPHER_NONE;    <---Change here
                                                <---

         pTblp -> iCipherType = CIPHER_MULTI0; <---Example


         pTblp ->    bAdmin_live = FALSE;

                       <...>

         EndOfFunc:
         GW_tsh_functrace( FUNC_ADDTOAGENT, GW_FUNC_END, NULL);

         return(pTblp);
     }
```

7.16.3 Encryption Protocol Numbers

DFA uses an *encryption protocol number* to check whether or not the DFA gateway and the DFA agent are using the same encryption scheme.

The current set of encryption protocol numbers, as assigned by Hitachi, are as follows:

TABLE 7-16. Encryption Protocol Numbers

Numbers	Types	Description
0	**No encryption**	A dummy used in the original DFA
1	**MULTI 2**	ECB (electronic codebook) mode, shuffle 8 times, effective key=all zeros
2	**MULTI 0**	ECB (electronic codebook) mode, shuffle 8 times, effective key=all zeros
3	**OSF recom.**	The encryption scheme recommended by OSF

The protocol numbers are maintained by Hitachi. If you need to add a new protocol number, you should contact Hitachi at the following address:

hashimoh@soft.hitachi.co.jp