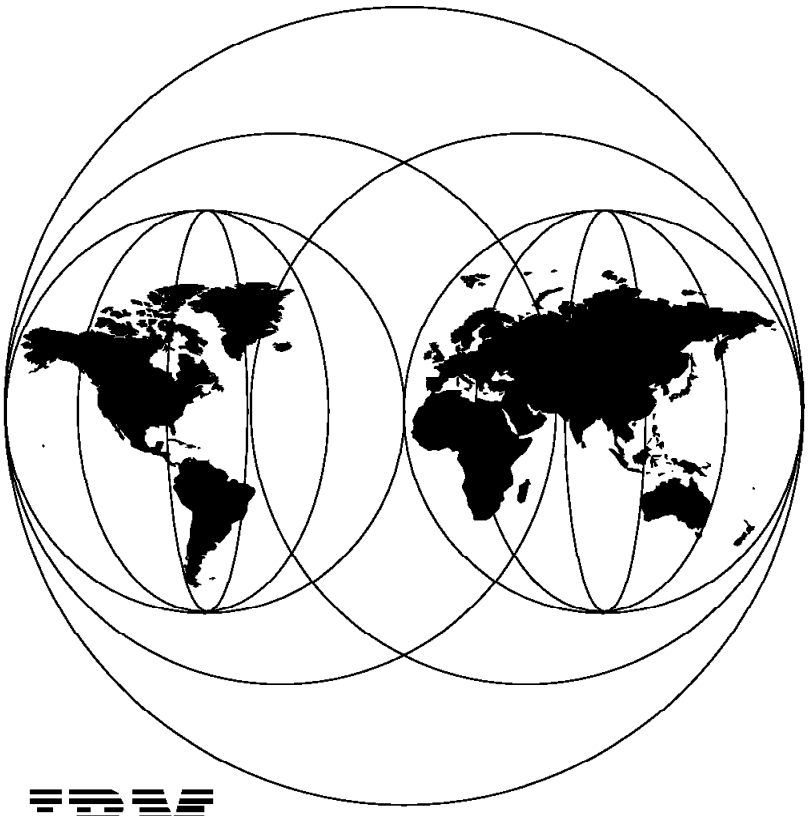
SG24-4714-00

Administering IBM DCE and DFS Version 2.1 for AIX (and OS/2 Clients)

August 1996



IBM

International Technical Support Organization

Austin Center

International Technical Support Organization

SG24-4714-00

Administering IBM DCE and DFS Version 2.1 for AIX (and OS/2 Clients)

August 1996

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 305.

First Edition (August 1996)

This edition applies to the IBM DCE Version 2.1 Family for AIX Version 4.1 and the IBM DCE 2.1 for OS/2 Warp Beta Program.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 045 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Preface

Setting up an initial DCE/DFS cell with all the required machine roles and with users is fairly easy. However, if you want to do it right or if you want to reconfigure (parts of) your cell, you need in-depth knowledge — and guidance. This book is unique in its coverage of many of the planning and administration aspects of OSF's Distributed Computing Environment (DCE) and the Distributed File System (DFS).

The book begins by presenting a high-level overview of DCE and IBM's product packaging on the AIX and OS/2 Warp platforms. Then, beginning with a discussion of business requirements, guidance is provided on structuring the DCE/DFS cell-component layout for optimum performance and availability.

It walks step-by-step through the installation and customization of DCE and DFS services in both AIX and OS/2 Warp, putting particular emphasis also on the replication of DCE/DFS servers and DFS filesets.

After these basic configuration instructions, the discussion continues with different network topologies. Shortcut, step-by-step instructions for the implementation of several LAN-only, LAN/WAN, and intercell scenarios are provided, and special issues as well as performance and availability considerations for each of these environments are discussed.

The major part of this book is dedicated to DCE/DFS administration tasks ranging from day-to-day maintenance, such as backups and user management, to challenging endeavors, such as changing IP addresses or moving servers around. The following task-groups are covered in detail:

- Migration
- Splitting/joining cells, moving services, changing IP addresses
- Backup/restore
- Client cache and disk-space management
- Mass user administration and cell_admin account
- NIS/NFS integration and migration
- Managing remote services and servers
- Authenticated batch jobs

On the diskette that comes with this book are tools that support the administrator in tasks such as changing IP addresses, moving or copying a clearinghouse, refreshing client caches, and managing a large number of users. These tools currently only run on an AIX platform, but can be used to manage heterogeneous cells.

The first two chapters are intended for anyone who needs to understand the basic DCE components and related planning issues. System administrators and people involved in the marketing of DCE will gain insight as to what components must be used in certain business environments. The rest of the document is mainly intended for DCE administrators. It gives them guidance on how to lay out and implement the DCE components for different network topologies and how to perform many important administration routines.

This document is an update of the *Using and Administering AIX DCE 1.3* (GG24-4348) redbook, which remains valid for AIX 3.2.5/DCE 1.3 environments.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “Introduction” on page 1

  This chapter describes distributed client/server environments in general. It gives a short introduction on each DCE component, administration commands, and packaging.

- Chapter 2, “Planning DCE Cells” on page 19

  This chapter gives planners and administrators all the information they need to lay out a cell with all its servers and clients based on customer and business needs. It discusses technical feasibility and summarizes performance and availability issues that affect the cell layout.

- Chapter 3, “Implementing DCE Cells” on page 37

  This chapter walks step-by-step through the basic installation and configuration of the DCE core services on the AIX and OS/2 Warp platforms.

- Chapter 4, “Implementing DFS” on page 75

  This chapter covers the configuration of the Enhanced DFS environment. After detailed basic configuration instructions, it emphasizes fileset configuration and replication.

- Chapter 5, “Implementing Various LAN/WAN Scenarios” on page 105

  This chapter provides step-by-step configuration instructions for scenarios with different network topologies. It discusses laboratory experiences as well as performance and availability issues for each scenario.

- Chapter 6, “Administering DCE Cells” on page 139

  This chapter is organized in a task-oriented format. It describes certain administration routines in step-by-step detail. The tasks range from performing daily administration tasks to reconfiguring certain aspects of entire cells. It also covers integration/migration from an NFS/NIS environment as well as management of remote servers. System administrators will find this chapter especially beneficial.

- Chapter 7, “Miscellaneous Tools and Technologies” on page 241

  This chapter describes, as kind of a product history, the features that were new in DCE 1.3 and the ones that are new in DCE 2.1. It also introduces a framework of shell scripts to facilitate the entire user management of a cell.

The Team That Wrote This Redbook

Through the efforts of a team of specialists from around the world, this redbook was written at the International Technical Support Organization, Austin Center.

Rolf Lendenmann is an advisory system engineer at the International Technical Support Organization, Austin Center. He writes extensively on all areas of DCE. Before joining the ITSO two and a half years ago, Rolf worked for nine years in the AIX Technical Support department in Zurich, Switzerland as a product specialist and consultant supporting all areas of AIX system management, networking (SNA, TCP/IP), and middleware (DCE).

Brice Muang-Khot is a system engineer at the Field Support Center in Paris, France. He provides DCE support (design, implementation, problem resolution) to customers and IBM sales and technical representatives

Hanspeter Nagel is an advisory system engineer at the Banking Unit in Basel, Switzerland. He has eight years of experience in the distributed client/server field. His areas of expertise include middleware (DCE, CICS) and systems management. He has written extensively on DCE Administration.

Egide van Aerschot is a consultant system engineer at IBM's Banking Unit in Brussels, Belgium. He joined IBM 29 years ago, during which he has worked with customers for 26 years and taught classes at the International Education Center in La Hulpe for three years. His current activities include supporting installations of IMS, MQI/MOSeries, and DCE as well as designing and prototyping applications under Common Language Environments (LE) written in PL/I, ASM, and C using APPC and DCE with access to DLI, DB2, and MQI resources.

Manabu Kurasawa is a system engineer at the Field Support Center in Tokyo, Japan. He joined IBM 5 years ago. Working as an IT specialist, he has 4 years of experience in distributed processing middleware on AIX (DCE, Encina, and MQSeries).

Scott Vetter is an ITSO VM Area Specialist in the United States. He has 5 years of experience in his current field. He has completed 12 years with IBM. His areas of expertise include Open Edition functionality covering the POSIX, DCE, and GUI components. He has written many redbooks that cover DCE, POSIX, GUI, and core VM functionality. His latest redbook is the VM/ESA DCE Notebook.

Heinz Johner is an advisory system engineer from IBM Switzerland with several years of field experience in large UNIX customer environments. His major work areas in these projects were distributed-systems management and implementation of DCE. He is currently a staff member at the ITSO Center in Austin and is assuming the responsibilities for DCE on AIX and OS/2 that were previously held by Rolf Lendenmann.

This document is an update of the document *Using and Administering AIX DCE 1.3* (GG24-4348), which remains valid for AIX 3.2.5/DCE 1.3 environments. The previous document was created by:

Rolf Lendenmann	IBM ITSO Austin
Brice Muang-Khot	IBM France
Hanspeter Nagel	IBM Switzerland
Salvatore La Pietra	IBM Germany
Jacques Dubuquoy	IBM Belgium

We would like to offer our thanks to Marcus Brewer of the ITSO Austin and to the IBM Austin DCE and DFS development teams, the DCE sales team, and the DCE brand management team for their invaluable advice and assistance in the reviewing of this document.

Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address:

redbook@vnet.ibm.com

Your comments are important to us!

Chapter 1. Introduction

Distributed client/server (C/S) environments represent a way to quickly and transparently deliver information to users from various different sources and locations. C/S technology can help companies reduce cost and time as well as improve quality and customer satisfaction. However, this technology must be well understood before being deployed and should be supported from company management down to system administrators and users. The C/S distributed environment sets new dimensions for its design, architecture and management. Even company policies should adapt to meet the challenge of C/S distributed environments. Benefits and new implications must be carefully studied when you design and architect new applications based on C/S technology.

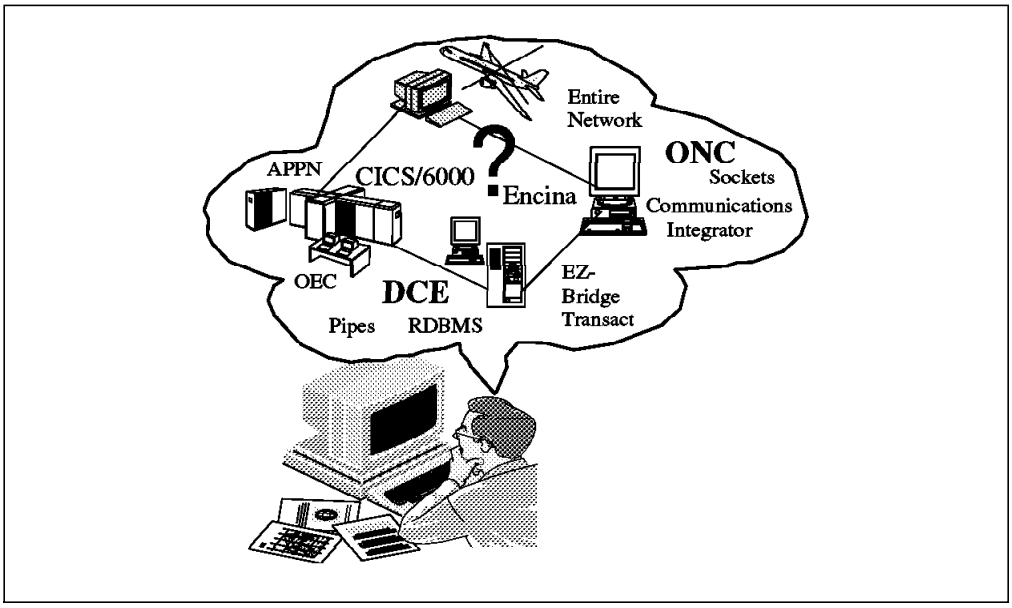


Figure 1. Options in a Distributed Environment

When looking for right-sizing and re-engineering some of their existing applications with distributed C/S technology, most customers have no idea how much this transition phase will cost in terms of:

- New hardware equipment
- Software
- Time
- Resources
- Skills
- Administration
- Security

and how much C/S technology will change the way companies do business.

In the last few years technologies have evolved faster than businesses. What appeared innovative turned out to be a waste of capital investment. Nevertheless, most customers agreed a distributed C/S environment is a good choice because it will:

- Leverage the mainframe investment
- Localize problems and solutions

- Reduce software development cost
- Reduce software/hardware maintenance cost
- Better organize data and applications
- Increase application portability
- Increase scalability and migration
- Improve system and network performance
- Allow for a multi-vendor environment with a wider choice of platforms
- Make users more autonomous by moving applications closer to them
- Facilitate the use of standards and acceptance of open systems

In the following sections of this chapter, we will present an overview of C/S technologies and the Open Software Foundation Distributed Computing Environment (OSF DCE) components and architecture. We also explain the DCE packaging for AIX and OS/2.

1.1 Overview of Client/Server Technologies

The phrase *client/server* was first adopted by Sybase in the late 1980s to market their database technology. The C/S model implies cooperative and distributed processing. C/S computing relies on a message-based communication between a requester (or a client) that asks for a specific service and a responder (or a server) that provides the information. The message exchange can be synchronous or asynchronous. Examples of synchronous communications are Remote Procedure Calls (RPCs) or System Network Architecture (SNA) LU 6.2 conversations. Asynchronous examples are the Encina Recoverable Queueing System (Encina RQS) or the Message Queueing Interface (MQI), which is part of the IBM Messaging and Queueing Series (MQSeries) as defined in the IBM Open Blueprint.

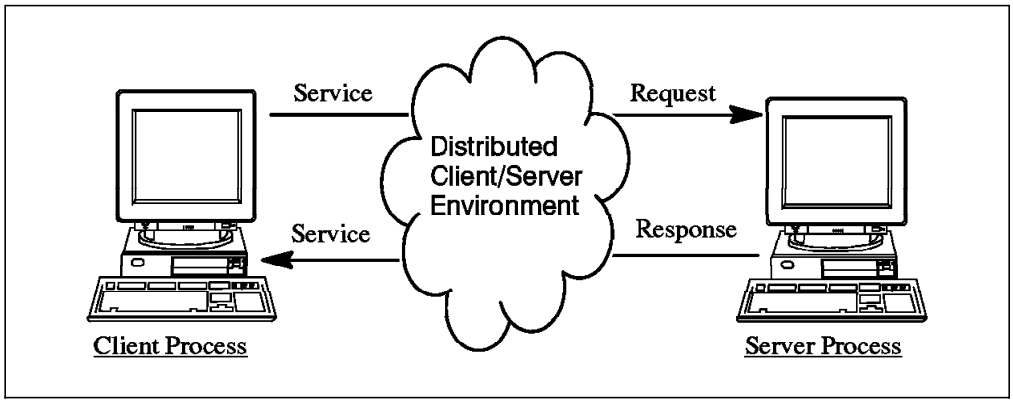


Figure 2. Client/Server Model

The simplest form of C/S computing has only two pieces: a client process and a server process connected via a network. The server process is the provider of services, and the client is a consumer of services. Clients usually manage the user-interface portion of the application, while server programs generally receive requests from client programs, execute the specified action, and dispatch the response to clients. C/S programming has become the most widely accepted paradigm to develop distributed applications that interoperate across a network.

Distributed C/S systems enable users to make better use of their computer resources. They provide better control over the applications and help integrate diverse sets of hardware and software. The flexibility introduced by distributed C/S systems brings with it several questions, options, and approaches on how to plan, configure, and manage all the resources in such an environment. Resources can be computers, devices, applications, users, groups, and so on.

In a C/S distributed environment, the distributed services can be replicated for high availability; a server request can turn into a client request to another server, and multiple servers can run on the same system and so on.

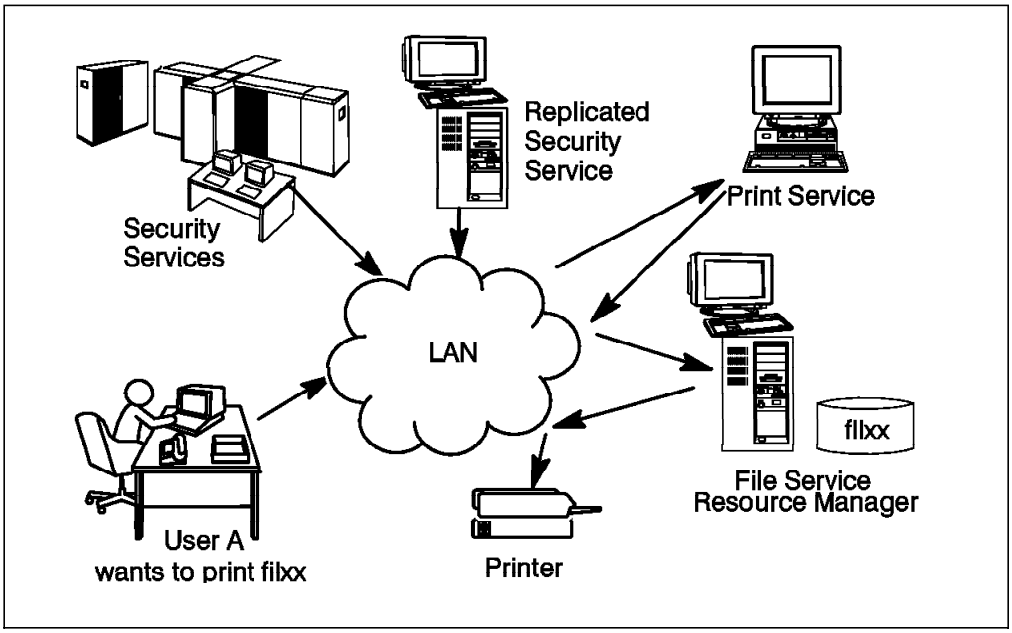


Figure 3. Example of C/S Application

Although the C/S model is the same, a small distinction should be made between a two-tier model and a three-tier model.

1.1.1 Two-Tier Client/Server Model

The two-tier model, also called data-oriented model, shown in Figure 4 on page 4, is the classic C/S computing model where the client sends a request for data and the server searches and sends the data back to the client. Remote Procedure Call services based on DCE or Open Network Computing (ONC) map to this model.

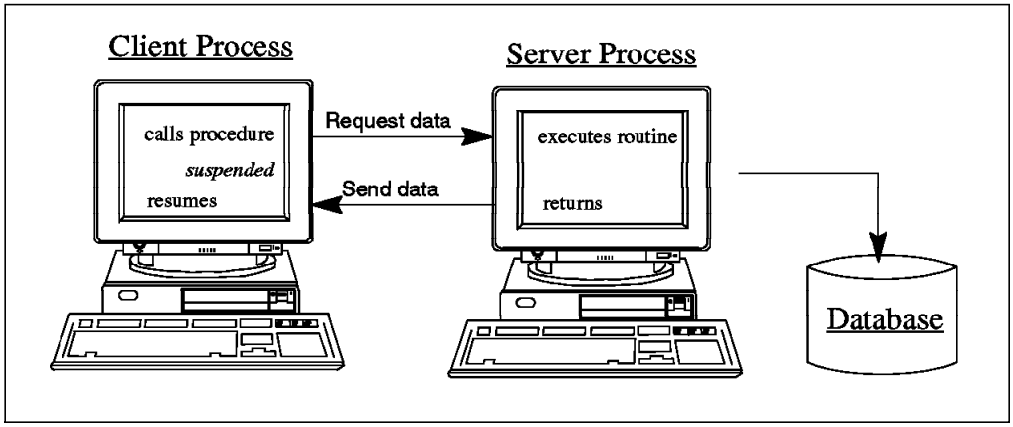


Figure 4. Two-Tier Model

The most popular representation of a two-tier model are Relational Database Management Systems (RDBMSs). The client machine accesses a server that holds data. The application logic is on the client, which prepares SQL data access commands and may receive a large amount of data to process.

1.1.2 Three-Tier Client/Server Model

For the three-tier model, also called application-oriented, shown in Figure 5, a monitor is included. The clients request application services from the monitor and supply the required parameters. The monitor locates the desired service, verifies the security credentials, and schedules the request for execution by an application service.

Products such as Encina and CICS working on top of DCE map to such a model.

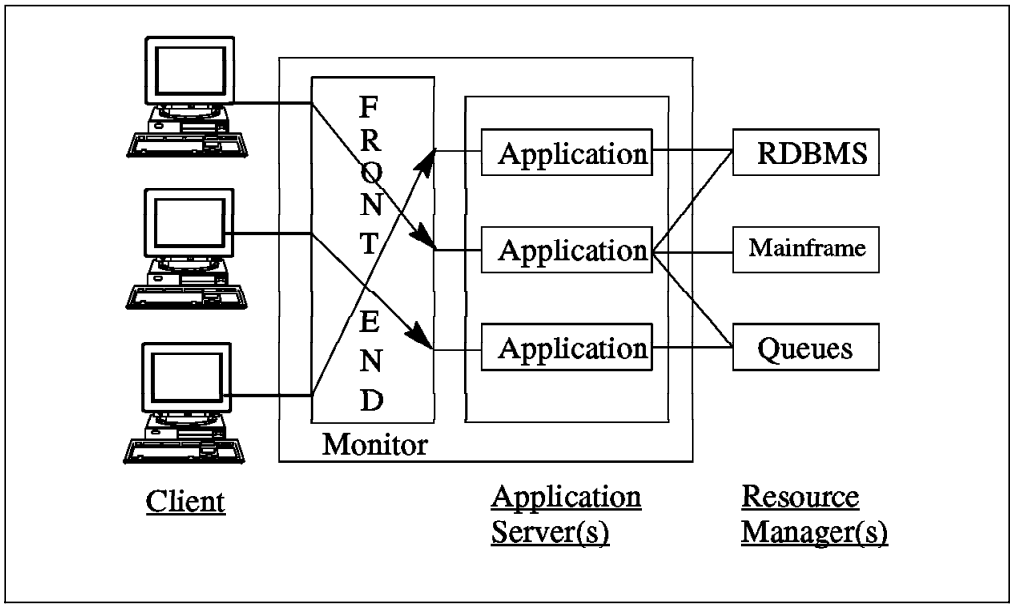


Figure 5. Three-Tier Model

Data access and processing is made on fast distributed and replicated servers which are connected with powerful database machines over fast links. The clients are only front ends that need not be too powerful and need not be connected with fast links. So the three-tier model has several advantages over the two-tier model:

- Better availability
- Better scalability
- Better load balancing
- Lower costs for client machines and networks
- More application- and business-process oriented

The elegance of any type of C/S model and its ability to mix and match languages and HW/SW platforms is obvious. However, this flexibility makes administration difficult in this environment.

1.2 DCE Overview

DCE has established a defacto standard in the client/server arena that is available on all IBM operating systems or platforms, such as AIX, OS/2, MVS, VM, and OS/400. It is also available on major competitor platforms, such as Microsoft Windows (Windows 3.1, Windows 95, and Windows-NT), DEC VMS, Siemens-Nixdorf BS/2000, HP MPE/ix, and on all the UNIX flavors including: AIX, Digital UNIX, Solaris, HP/UX, DG/UX, Sinix, and SCO. All this has evolved in less than three years from the first release of OSF DCE 1.0. You can find up-to-date information on supported platforms and available DCE applications on the OSF World Wide Web at page `http://www.osf.org/dce`.

In this documentation, we want to help system engineers (SEs), customers, and marketing representatives to:

- Plan for DCE
- Understand what configuration can best map the customer's business environment
- Recognize the most common administration tasks in a DCE environment
- Understand what to do when migrating from NIS to DCE or from an environment of networked systems to DCE
- Optimize performance and availability of your DCE environment

We try to answer all these questions and provide a good, fair DCE perspective.

1.2.1 OSF DCE Architecture

OSF DCE is a complete architecture that takes full advantage of the client/server paradigm. It offers a set of services and APIs that can be used to build distributed applications and a set of management tools to manage the distributed environment. It can interoperate with other environments.

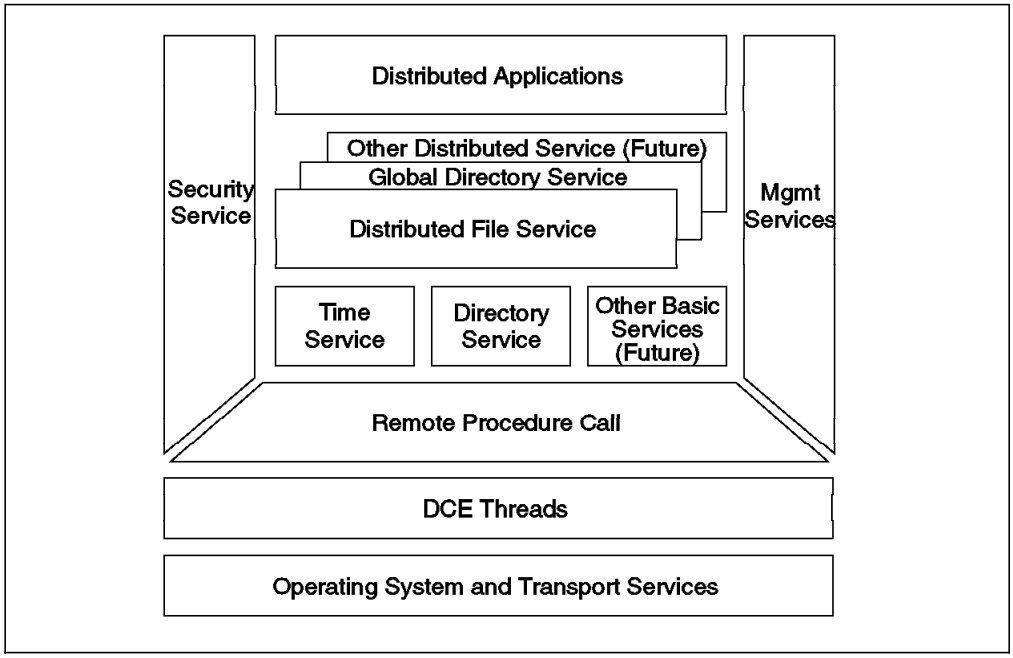


Figure 6. DCE Architecture

If we consider DCE as *Middleware* (like other Network Operating Systems, such as Novell NetWare), the operating system is hidden to a certain extent by a set of core and extended services offered by DCE. The users will only see the distributed client/server application. It will be completely transparent to them whether the application is local or distributed and what operating system is underneath a distributed service. The architecture viewing DCE as middleware is explained in the following sections.

1.2.2 DCE Threads

Threads support the creation, management, and synchronization of multiple concurrent execution paths within a single process. This provides a great deal of flexibility to application developers in a variety of areas, such as parallel execution, task control, exploitation of multiprocessor machines, and faster task switching. On the other hand, threads introduce considerable, additional complexity.

The DCE core services and all dependent applications use threads. This all happens behind the scenes. Customer applications may or may not use threads for their own purposes. However, application developers must know they *are* using threads anyway through the DCE RPC run-time services, unless they explicitly implement a single-threaded server.

DCE threads are originally based upon Digital's implementation of Concert Multithread Architecture (CMA). The current DCE threads implementation (OSF DCE 1.1) is based on the POSIX specification 1003.4a Draft 4, called the *pthreads* interface. It is designed as a user-level thread package that can run on operating systems that do not support threads in their kernel.

If the operating system, such as AIX 3.2, does not support threads in its kernel, the threads are running in (non-privileged) user mode. The kernel is not aware of threads running in a process — it can only see (and dispatch) the process as a whole. The problem is that one thread could put the entire process into a wait

state, thereby making all other threads also wait. Programmers have to be aware of this situation if they use threads.

If an operating system has an alternative implementation of POSIX-compatible threads, the DCE thread calls may be mapped directly to kernel threads, and the DCE threads library just has a mapping function. The (kernel) threads library for AIX 4.1 follows the POSIX 1003.4a Draft 7 specification. AIX 4.1 offers a DCE pthreads compatibility library which is an implementation of the user-level threads on top of its kernel threads. This compatibility library maps the DCE threads (Draft 4) to the AIX kernel threads (Draft 7). However, the signal semantics adheres to POSIX 1003.4a Draft 7. This gives us source compatibility for existing multithreaded programs.

For a complete overview on the threads facility, see Chapter 11 in the ITSO Austin redbook *Understanding OSF DCE 1.1 for AIX and OS/2* and the DCE-related documentation listed in Appendix C, “Related Publications” on page 307.

1.2.3 DCE Remote Procedure Call

The DCE Remote Procedure Call (RPC) facility allows individual procedures in an application to run on a computer somewhere else in the network. DCE RPC extends the typical procedure call model by supporting direct calls to procedures on remote systems. RPC presentation services mask the differences between data representations on different machines and networking details to allow programs to work across heterogeneous systems.

DCE RPC provides programmers with several powerful tools necessary for building client/server applications. Development tools consist of:

- Interface Definition Language (IDL) and related compiler *idl*
- *uuidgen* — generates UUIDs (a 32-digit number) to uniquely identify resources, services, and users in DCE independently from time and space
- Run-time service — implements the network protocol and communication between the client and server applications

Using RPC in combination with threads allows a client application to call several servers at once, for instance, for parallel calculation processes. For a complete overview on the RPC facility, see Chapter 10 in the ITSO Austin redbook *Understanding OSF DCE 1.1 for AIX and OS/2* and the DCE-related documentation listed in Appendix C, “Related Publications” on page 307.

1.2.4 DCE Security Service

Distributed computing encourages a free flow of data between nodes, thereby expanding the capabilities of interconnectivity and interoperability. Security breaches might come from any component of the distributed system. Security threats can be:

- Eavesdropping: Data can be read as it flows over the network.
- Masquerading: A system can pretend to be another system and thus gain unauthorized access to resources.
- Modification: Data can be modified as it flows over the network.
- Denial of service: Service can be denied from an unauthorized source.

These are just a few of the problems that can affect requirements such as:

- Confidentiality: Protection against unauthorized access to information
- Integrity: Protection against unauthorized modification of information
- Availability: Protection against unauthorized impairment of functionalities

The DCE Security Service is a strong building block of the DCE core services based on Kerberos technology that provides secure authentication, authorization, and auditing mechanisms for users and distributed client/server applications. For data encryption, DCE uses either Data Encryption Standard (DES) or the Common Data Masking Facility (CDMF). CDMF works with a 40-bit encryption key in contrast to the 52-bit DES key. CDMF is allowed to be exported from the USA, whereas DES underlies certain export restrictions.

Security is one of the main reasons why customers are interested in DCE. Developers can use the DCE Security Service to make their distributed client/server applications or products secure. They do not necessarily need to use the DCE RPC API. The GSS-API allows interaction with the DCE Security Service without using any other DCE components. In their Open Blueprint strategy paper, IBM announced that it would integrate the DCE Security Service into other products to provide them a higher and common level of security.

Administrators must understand all the concepts and components of the Security Service. As other products, such as RACF, DB2, LAN Server, CICS/6000, or AIX security in general, imbed a DCE security layer, the importance of DCE security is growing dramatically. DCE becomes the only choice for a centralized secure environment for all the IT security needs of a company.

The DCE security component comprises three services running on the security server and several other facilities. Most of the DCE security is related to the concept of a principal. A principal is an entity that can be securely identified and can engage in a trusted communication. A principal usually represents a user, a network service, a particular host, or a cell. Each principal is uniquely named and identified by its principal UUID. A record for each principal containing the name, the private keys, and the expiration date is kept in the registry database on a highly secure system.

The three services are:

- Registry Service (RS) — A replicated service that maintains the cell's security database. This database contains entries for accounts, principals, groups, organizations, and administrative policies.
- Authentication Service (AS) — Used to verify the identity of principals. It contains a Ticket-Granting Service (TGS) that grants tickets to these principals and services so they can engage in a secure communication.
- Privilege Service (PS) — Certifies a principal's credentials that are going to be forwarded in a secure way to DCE servers. The credentials (see EPACs below) allow the target server to check the principal's access rights to resources.

These services are implemented in the security server daemon (secd).

The administrator can create several security replica servers to balance the load on the master security server and to preserve the cell in case the master becomes disabled. The sites where the security database will be replicated

must be as secure as the site where the master copy of the security database is stored.

> **Note**
>
> Each replicated security server requires a separate license of the program.

On each client machine, there is a Security Validation Service integrated into the DCE daemon process (dced) whose duties is to verify that the security server is authentic, to manage the machine principal and to certify the login contexts.

Client applications actually use the Security Service facilities and services via a Security Service API or the Generic Security Services API (GSS-API). The facilities serving these APIs are:

- Login Facility (LF) — Initializes a user's DCE security environment (login context) and provides them with their security credentials.
- Extended Registry Attribute (ERA) — Extends the standard set of registry database attributes (which cannot be changed) and allows for user-defined attributes.
- Extended Privilege Attribute Certificate (EPAC) — Basically a certified list of the principal's name, the groups to which the principal is a member, and the ERAs for an authenticated principal. A client must present its EPAC to a server when performing authenticated RPC. The server uses the EPAC to examine the client's access rights. Other information in the EPAC allows clients and servers to invoke secure operations through one or more intermediate servers (delegation).
- Access Control List (ACL) Facility — An ACL is a list of principals or groups and their access permissions. ACLs are assigned to any type of resource that DCE servers manage. The ACL facility provides a generalized means of checking a principal's access request against the ACLs on the requested resource.
- Key Management Facility — Enables non-interactive principals (such as application servers) to manage their secret keys.
- ID Map Facility — Allows intercell communication, mapping local cell principal names to global cell principal names and vice versa.
- Password Management Facility — Enables principal passwords to be generated and to be submitted to strength checks beyond those defined in DCE standard policy.
- Audit Service — Detects and reports events that are relevant to the management of a secure environment. Events are written in a log file called an audit trail file. The application programmers need to use an audit API to build auditing of relevant operations into their applications.

A complete set of DCE Security APIs is offered for writing trusted distributed applications. For a complete overview on DCE Security, see Chapter 3 in the ITSO Austin redbook *Understanding OSF DCE 1.1 for AIX and OS/2* and the DCE-related documentation listed in Appendix C, "Related Publications" on page 307.

1.2.5 DCE Directory Service

A distributed computing environment contains many users, computers, applications, and printers dispersed in the network. This creates a complex group of resources and users that somehow have to be located. These resources and users, also referred to as objects, can easily be located if we have a centralized process that keeps track of every change in the network.

The Directory Service is the component that makes it possible for the user to locate objects in the network without knowing their physical location. For the user, the distributed nature of the environment is hidden. It is like a telephone directory assistance service that provides the phone number when given a person's name.

Users do not normally access or use directory services directly. They run applications that may use the directory services to find objects. The only thing a user might have to know is object names and maybe the naming model.

The heart of this Directory Service is the Cell Directory Service (CDS) namespace. The CDS namespace consists of one or more clearinghouses. Each of them may be on a different system that runs a secondary directory server. Because the clearinghouses may be distributed over several CDS servers, the CDS namespace represents a highly distributed database with all its advantages and problems. We developed some scripts to help you to manage this distributed database. However, DCE administrators must understand the directory service in order to be able to administer objects and manage the service and its database.

The programmer can either use the X/Open Directory Service (XDS) API to directly access the directory service or the RPC Name Service Interface (NSI) from within DCE applications.

The DCE Directory Service includes the:

- Cell Directory Service (CDS)
- Global Directory Service (GDS)
- Global Directory Agent (GDA)
- Application Programming Interface (API)

The CDS manages information within a cell. The GDS is based on the CCITT X.500 name schema and provides the basis for a global namespace. The GDA is the CDS gateway to intercell communication. The GDA supports both Internet addresses and X.500 addresses. If the address passed to the GDA is an X.500 address, the GDA contacts the GDS. If the address passed to GDA is an Internet address, then the GDA uses the Internet Domain Name Service (DNS) to locate the foreign cell. Both CDS and GDS use the X/Open Directory Service (XDS) API as a programming interface.

For a complete overview on DCE Directory Service, see Chapter 2 in the ITSO Austin redbook *Understanding OSF DCE 1.1 for AIX and OS/2* and the DCE-related documentation listed in Appendix C, "Related Publications" on page 307.

1.2.6 DCE Distributed Time Service (DTS)

DTS provides precise, fault-tolerant clock synchronization on the computers participating in a DCE environment both over LANs and WANs. The synchronized clocks enable DCE applications to determine event sequencing, duration and scheduling. The core services, especially the Ticket Granting Service, rely heavily on synchronized clocks. Note that installing DTS is not a requirement; the clocks could be synchronized by other time services. However, the use of DTS is highly recommended because it uses security service and adjusts time smoothly rather than correcting system clocks all at once or even backwards. The DTS clerks obtain time information from at least three DTS servers in a LAN and adjust their time. If they do not receive the required number of time values in their LAN, they contact global DTS servers.

DTS is based on Coordinated Universal Time (CUT or UTC), an international time standard. Different types of time servers provide for different transmission delays between LANs and WANs that would influence correct time calculation:

- Local DTS Servers maintain synchronization within a LAN and synchronize their own clocks using the responses of all other DTS servers in the LAN. If they do not get at least responses from two other DTS servers in their own LAN, they have to contact global DTS servers.
- Global DTS Servers (usually at least one per LAN) advertise themselves into the CDS so other DTS servers or even clerks can contact them, if they do not have the required number of DTS servers in their own LAN. To adjust their own clocks, they act like local DTS servers. If they get their time from an external time provider, they do not adjust their clocks with values obtained from other DTS servers.
- Courier DTS Servers (usually one per LAN) maintain synchronization between multiple LANs. Any local or global DTS server can have a courier role. What is special about this role is, they *must* contact one global DTS server, even if they get enough time values from DTS servers in their own LAN.

A DCE DTS API is offered as well as a Time Provider Interface (TPI) which allows a time provider process to pass its UTC time values to a DTS server. Many standards bodies disseminate UTC by radio, telephone, and satellite. TPI also permits other distributed time services, such as the Network Time Protocol (NTP), to work with DCE.

Replication of DTS servers does not require additional licenses because DTS is included in the DCE base product. For a complete overview on the DTS facility, see Chapter 4 in the ITSO Austin redbook *Understanding OSF DCE 1.1 for AIX and OS/2* and the DCE-related documentation listed in Appendix C, “Related Publications” on page 307.

1.2.7 Distributed File System

The Distributed File System (DFS) is a DCE application that provides global file sharing. Access to files located anywhere in interconnected DCE cells is transparent to the user. To the user, it appears as if the files were located on a local drive. DFS servers and clients may be heterogeneous computers running different operating systems.

The origin of DFS is Transarc Corporation’s implementation of the Andrew File System (AFS) from Carnegie-Mellon University. DFS conforms to POSIX 1003.1

for file system semantics and to POSIX 1003.6 for access control security. DFS is built onto and integrated with all of the other DCE services and was developed to address identified distributed file system needs, such as:

- Location transparency
- Uniform naming
- Good performance
- Security
- High availability
- File consistency control
- NFS interoperability

DFS is a distributed file system which allows users to share files stored in a network of computers as easily as files stored on a local machine/workstation. The DCE Distributed File System uses the client/server model that is common to other distributed file systems. The file system gives users a uniform name space, file location transparency, and high availability. Reliability is enhanced with a log-based physical file system which allows quick recovery after server failures. Files and directories can be replicated to multiple machines to provide reliable file access and availability. Security is provided by a secure RPC service and Access Control Lists that conform to POSIX 1003.6. DFS implements a superset of that POSIX ACL Draft.

As shown in Figure 6 on page 6, DFS is an extended service and is built on the DCE core services: Security, CDS and DTS. When accessing remote data, DFS uses DCE Remote Procedure Calls (RPCs) to communicate between participating systems, exchanging authorization requests, access requests, file and directory data, and synchronization information. It uses the DCE Naming Services to resolve global names and the DCE Security Service to authenticate users and services. DFS requires synchronized clocks among all involved systems. This is achieved with the DCE Time Service.

The DCE Local File System (LFS) is a log-based file system that is integrated into the kernel. It is also based on aggregates that are equivalent to standard UNIX disk partitions or AIX logical volumes. Aggregates are logically composed of multiple filesets, which are mountable subtrees. Filesets share the disk blocks within an aggregate. Filesets can be administered and referenced individually. Quotas can be set on a per fileset basis. Filesets are the units that provide support for administrative functions needed in a distributed environment, such as replication, cloning, reconfiguration (move filesets for load balancing), and backup. The cloning function provides copy-on-write semantics so that double disk space is not needed when a fileset is cloned. Cloning also allows the above-mentioned functions to be performed while the filesets are on-line with minimal down time for users of the filesets.

Directories and files can be accessed by users anywhere on the network, using the same file or directory names, since all DCE resources are part of a global namespace. High performance is achieved through caching on the client side to reduce access time and network traffic.

DFS has many advantages over NFS:

- Stateful implementation allows for caching on client side
- Provides single site read/write semantics
- Fileset replication
- Security (Authentication and ACLs)
- Cloning

- Backup Servers

DFS files can be exported to NFS so that NFS clients can access them as unauthenticated users. The NFS/DFS Authenticating Gateway product provides a mapping of NFS users into authenticated DFS users. To achieve this, NFS users use the `dfsiauth` command to perform a DCE login to set up credentials for a certain combination of userIDs/nodeIDs, which will be revoked when the ticket expires.

For more information on the DFS facility, see the ITSO Austin redbook *The Distributed File System (DFS) for AIX/6000* and the DCE-related documentation listed in Appendix C, "Related Publications" on page 307.

1.2.8 Mutual Dependencies between DCE Components

Core services in DCE, such as the Cell Directory Service and Distributed Time Service, use the Security Service. The Security Service in turn uses CDS and RPC and so on. The following table shows what uses what.

Is using ->	DFS	GDS	CDS	DTS	RPC	Sec	Thr
DFS	*		√	√	√	√	√
GDS		*					√
CDS		(√)	*	√	√	√	√
DTS				*	√	√	√
RPC				√	*	√	√
Security			√	√	√	*	√
Threads							*

Figure 7. Dependencies between the DCE Components

CDS is not dependent upon GDS, but it can use it via the Global Directory Agent (GDA) component of CDS.

Because of these interdependencies, the services must be configured and started up in a certain sequence and there are auxiliary files to bypass yet-missing components.

1.2.9 DCE Management Services

Several administration tools are provided to manage DCE. The following commands are provided by OSF and are therefore available on all DCE implementations:

- Security Service

`rgy_edit`	Security registry management (principals, accounts)
`acl_edit`	Consistent interface to different ACL managers
`sec_admin`	Controls operation of the security servers
`rmxcred`	Purges expired tickets from the credentials directory
`passwd_import`	Creates registry entries from /etc/passwd files (UNIX)
`passwd_export`	Creates /etc/passwd type file out of registry entries (UNIX)

- Directory Service

`cdscp`	General CDS client and server management interface

cdsli	Listing of all CDS namespace entries
cdsbrowser	Query tool for CDS objects
cdsdel	Can recursively delete entire directory subtrees in the CDS namespace

- Remote Procedure Call

rpccp	Management of RPC daemon and RPC CDS entries

- Time Service

dtscp	Management of time servers

- Distributed File System

fts	Command suite for file server management
bos	Command suite management for general DFS management
bak	Command suite for data backup management
cm	Command suite for DFS client cache management

- DCE Control Program *dcecp*

  The DCE Control Program (the dcecp command) is the new administrator tool in OSF DCE 1.1 that integrates the functions of several tools used in OSF DCE 1.0.x, such as cdscp, rpccp, rgy_edit, acl_edit, and dtscp. The dcecp language is based on the Tool Command Language, generally known as Tcl (pronounced "tickle"). The dcecp commands are implemented as Tcl commands in a Tcl interpreter.

  For a complete overview on the dcecp facility and Tcl, see Chapter 9 in the ITSO Austin redbook *Understanding OSF DCE 1.1 for AIX and OS/2* and the DCE-related documentation listed in Appendix C, "Related Publications" on page 307.

IBM improved the DCE management aspects by creating new high-level configuration commands and integrating all procedures into SMIT, thus hiding some of the complexity of the OSF commands. Examples of high-level commands in AIX are:

mkdce	Defines a machine with all its roles into a cell
rmdce	Deletes a machine from a cell
mkdfs	Defines DFS services on a machine
rmdfs	Deletes DFS services from a machine
mkdfslfs	Creates an LFS fileset on a DFS server machine, exports and mounts it
rmdfslfs	Removes an LFS fileset from a DFS server machine
mkdfsjfs	Exports a JFS file system from a DFS server machine and mounts it
rmdfsjfs	Removes a JFS file system from the DFS file space
lsdce	Lists the configuration status for the machine
rc.dce, rc.dfs	Start up script for selected or all DCE (DFS) services
dce.clean	Stop script for selected or all DCE (DFS) services
dfs.clean	Stop script for selected or all DFS services only

The SMIT menus can easily administer single entities (users, groups, accounts, ACLs), but there is no convenient way to administer multiple users. There is also a lot to do in the area of reconfiguring parts of the cell. It is the objective of this publication to provide a set of tools and tips to improve the administration of DCE.

In OS/2 we find the following added-value commands:

`cfgdce`	Command-line interface (CLI) for DCE configuration
`cfgdceg`	Starts the graphical user interface (GUI) for configuration
`ucfgdce`	Deletes a machine from a cell
`showcfg`	Lists the configuration status for the machine
`dcestart`	Start up command for selected or all DCE (DFS) components
`dcestop`	Stop command for selected or all DCE (DFS) components

1.2.10 Structure of a Distributed Computing Environment

In DCE a cell represents the smallest unit of resources, such as systems, users, services, and nodes, that work together and are administered together.

A minimal cell must include threads, the RPC communication layer, and at least one instance of all the core services:

- Cell Directory Server
- Security Server
- Distributed Time Servers (optional; at least three are recommended)

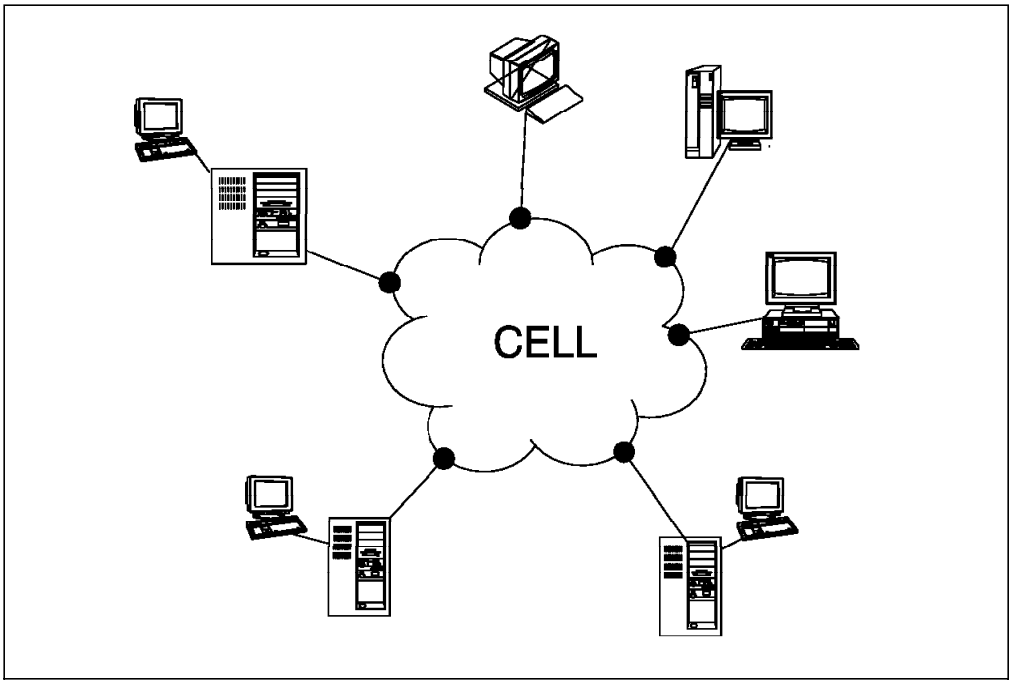


Figure 8. DCE Cell

Cells can be defined and configured in different ways depending on the user, administration and/or company requirements. For example, a small company that offers only one kind of service can be set up as a single cell as is shown in Figure 8.

Another example might be the faculty departments at the University of Texas at Austin. They can have their own manageable cells and use intercell communication for common services or data.

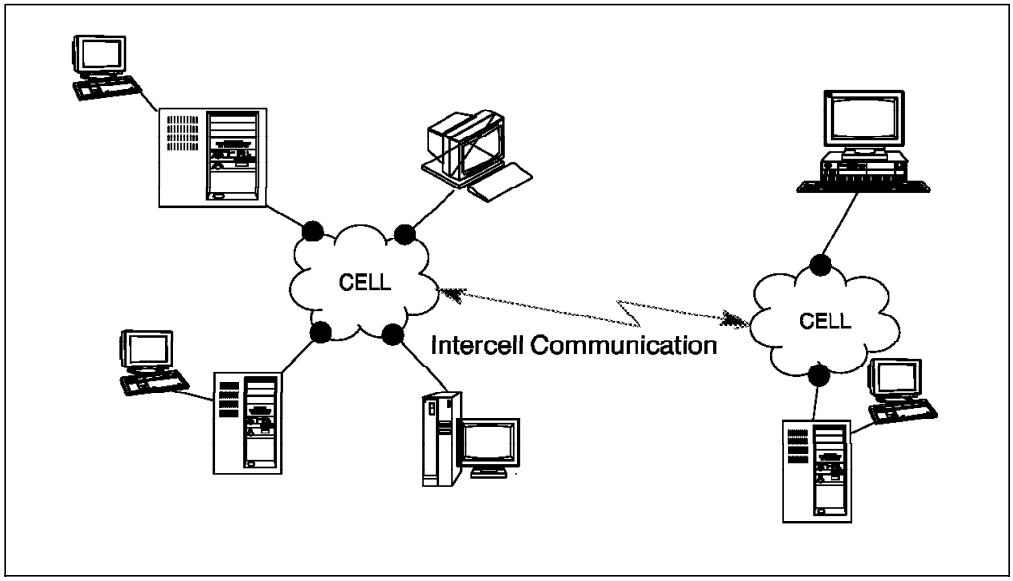


Figure 9. DCE Multicell Environment

Intercell communication is provided through GDA. The DCE architecture supports different types of network protocol families. The current OSF DCE reference implementation runs over the Internet Protocol (IP) family, using either UDP (User Datagram Protocol) or TCP (Transport Communication Protocol) as transport layers.

The home cell for a principal shows the cell where the information about the principal is stored. More generally speaking, a cell represents the collection of resources that use a common naming and security policy.

1.3 IBM DCE Product Information

There have been several IBM releases of DCE for both AIX and OS/2 DCE. Following is a description of the base OSF source code release and the major differences in packaging and prerequisites for each release.

1.3.1 DCE 1.3 Product Family for AIX 3.2.5

This release in based on OSF DCE V1.0.3 and consists of the following orderable packages:

- 5765-117, IBM DCE Base Services (DCE Client)

  Includes DCE Threads and all DCE client daemons plus DTS server and base DFS client and server, DCE Programming Tools and DES library option.
- 5765-118, IBM DCE Security Server
- 5765-119, IBM DCE Cell Directory Server
- 5765-120, IBM DCE X.500 Global Directory Server
- 5765-121, IBM DCE Enhanced DFS
- 5765-457, IBM DCE NFS to DFS Authenticating Gateway
- 5765-456, DCE Manager for NetView for AIX

1.3.2 DCE 2.1 Product Family for AIX 4.1.4

This release in based on OSF DCE V1.1 and consists of the following orderable packages:

- IBM DCE Base Services - integrated with AIX 4.1.x

  Like in DCE 1.3, DCE Threads and all DCE client daemons plus DTS server and base DFS client and server are included. However, the DCE Programming Tools as well as the DES and CDMF libraries are NOT included.

  AIX DES is available in 5765-418 and AIX CDMF is available in 5765-538 (User Data Masking Encryption Facility).

- 5765-533, IBM DCE Security Server
- 5765-534, IBM DCE Cell Directory Server
- 5765-537, IBM DCE Enhanced DFS
- 5765-540, IBM DCE NFS to DFS Authenticating Gateway
- 5765-532, Getting Started With DCE

  Contains all the DCE programming tools and sample programs. The IDL compiler also comes with this package.

- 5765-418, AIX Data Encryption Specification

  DES only needs to be ordered if you are writing encrypted applications. Only one license is needed per building.

- 5765-538, DCE User Data Masking Encryption Facility

  This encryption service can be used in countries where the export of DES is prohibited. Only one license is needed per building.

1.3.3 Directory and Security Server (DSS) for AIX, Version 4

DSS for AIX is the same code base as DCE V2.1 for AIX (with applied PTFs), but repackaged for ease of installation and ordering. It includes:

- Security Server
- Cell Directory Server
- Programming Tools

The DCE and DFS Base Services are still integrated with AIX V4.1.4+. However, for the customer's convenience, the DCE/DFS Base Services are also contained on the DSS distribution media. This is the package that can be ordered:

- 5765-639, Directory and Security Server (DSS) for AIX, Version 4

The DSS directory and security servers may be installed on separate physical RISC System machines. Extra individual DCE CDS or Security Servers can be ordered using the DCE V2.1 product numbers (above); Enhanced DFS must be ordered with the DCE V2.1 product number above.

1.3.4 DCE for OS/2 and Windows

The following packages are based on OSF DCE V 1.0.3:

- 5871-AAA, Feature Number 6199 - 96F8690 DCE SDK V1.0 OS/2 & Windows

  This package comes with the User Data Privacy (DES) option.

- 5871-AAA, Feature Number 6203 - 96F8691 DCE Client for OS/2

  This package comes with the User Data Privacy (DES) option.

- 5871-AAA, Feature Number 6201- 96F8692 DCE Client 1.1 for Windows

  This package comes with the User Data Privacy (DES) option.

1.3.5 Directory and Security Server for OS/2 Warp

This product is based on OSF DCE V1.1. The official Beta version was used in preparing this book. DSS for Warp will be generally available in 3Q96.

The part numbers for DSS for OS/2 Warp are as follows:

- 10H9754, DSS for OS/2 Warp, Version 4 (DES)
- 25H7945, DSS for OS/2 Warp, Version 4 (CDMF)

Both of these packages include the DCE base services, the CDS and Security servers, the DFS client, programming tools, and a software adoption layer for Warp Server integration.

DSS for Warp allows unlimited replication of the DCE Client for Warp.

Customers using non-Warp DSS servers, such as AIX DCE servers, should order one of the following DCE Client for Warp packages:

- 25H7940, DCE Client Including DFS, Version 4 (DES)
- 25H7946, DCE Client Including DFS, Version 4 (CDMF)

Chapter 2. Planning DCE Cells

This chapter intends to give planners and administrators all the information they need to lay out a cell, with all its servers and clients, based on customer and business needs, but also being aware of the technical feasibility and efficiency.

It summarizes the experiences we made during our testing or in discussions with development. It should also help provide a basic understanding of the DCE planning issues to readers not interested in technical details.

It explains, to a certain extent, how the different base components work as well as the technical restrictions implied by the DCE core servers and DFS. A planner must understand this to be able to design a solution for a customer which makes sense regarding:

- Reliability
- Availability
- Security
- Performance/Efficiency
- Cost

This chapter is organized in the following way:

2.1 General Considerations for DCE Cell Design

Today’s client/server computing systems are not only based on communication protocols and peer-to-peer connections but on a real network operating system. DCE is such a network operating system, and its functionality is as powerful as most of the known single-node operating systems, such as UNIX.

Based on the experiences we made during this project, we would like to give you some guidelines on how to design a DCE cell.

Prior to installing and configuring DCE, it is very important that you plan and design your cell carefully. Several aspects have to be taken into account. Therefore, you must clarify several questions beforehand:

1. Are you familiar with the different DCE core components?

   In order to understand how a DCE cell has to be designed, it is absolutely crucial that you really understand the core components of DCE and the way they work. For example, a high performance network may not improve your DCE performance when the preferred binding handles point to a slow interface. Or skulking over slow links (for example a 9600 baud connection)

may slow down the operation of your whole cell, if it is done too frequently. These are just two examples of things that can happen.

2. How is your company structured?

   - How large are the branches or regional offices?
   - How many and what kind of network services do the branches need?
   - How does the business data flow?
   - What kind of data and service access needs are there between branches and the main site?
   - What kind of data and service access needs are there from branch to branch?
   - What is the amount and frequency of such data access?

   The answers to these questions will help to decide whether we need a single or multicell design.

3. Does your company have naming conventions?

   Naming conventions are not only important for DCE but also for many other Information Technology (IT) areas. They are the base for security, stability, reliability, and accessibility in a network. Once you assign a name to any type of entity in a complex networked environment, it becomes very difficult to change it when necessary. Making changes is always more expensive than carefully planning ahead.

4. Does your company have security conventions?

   Most companies have security conventions or even a security department that takes care of all the security issues within the company. Since security is a major strength of DCE, it is absolutely necessary to get these people involved in your activities or at least to take their rules into account. Questions such as:

   - Where should a security server be placed?
   - Does a security server have to be a dedicated system?
   - Who is responsible for all user information?
   - Who is responsible for access control lists?

   are very important and must be answered properly in order for DCE to be part of the technologies that satisfy the company's security policy.

5. Does your company have system administration conventions?

   System management for distributed systems becomes more and more important as the size and the complexity of the distributed environment grow. The main disciplines of open client/server system management are:

   - Configuration and change management
   - Security management
   - Inventory, monitoring and reporting
   - Operations management
   - Client/server application management
   - Network management
   - Helpdesk

   While designing a DCE environment, you should consider who has to take care of these issues and how they are being solved. It may have an impact on how you will place certain services/servers and/or what kind of tools you are going to use.

6. Does your company already have any network operating systems?

If your company is already using network operating systems other than DCE, which is likely, you must consider how these different systems can coexist. In certain cases, these systems not only have to coexist but also interoperate with each other. For example, if you are already working with NIS, you probably have also installed NFS. As you are going to set up DCE/DFS, you will want to integrate NFS into your DCE/DFS environment. This may have an impact on where you place your services within your network.

7. What are the physical connection possibilities between the branches of your company?

   Your physical network has a big influence on where you place which services. Maintaining replicas through a slow communication link, may slow down the whole cell, or at least particular functions.

This means you must understand the way your whole company works, rather then just have IT experience. In the following sections, we will discuss the considerations for designing a cell in more detail.

2.2 Technical Implications Imposed by the Core Components

In order to decide how to lay out the DCE core services, we must know the way they work. The aspects to look at are:

- Replication capabilities
- Server selection mechanisms
- Login integration capabilities

2.2.1 Replication Capabilities

All DCE core and DFS servers can be replicated. Replication means that there are multiple instances of the service available in the cell, each of which controls its own copy of a replicated database.

So, there are multiple copies of the same databases in the cell, but each type of database has one master copy and possibly several read-only copies. Changes can only be made to the master copy. As long as the applications or the core service clients, respectively, only need read access, they can call any of the available servers. This increases:

- Performance: load balancing
- Availability: if one server fails, another can do the job

Since most of the accesses to the DCE core service databases are read-only, it makes sense to exploit these replication capabilities. Even to DFS many accesses are read-only. However, replication done the wrong way can cause slow operation of the whole cell.

The way in which the various DCE components replicate their data is different.

2.2.1.1 Security Replication

The security server has one master server which holds the master database. Replica servers can be configured. They hold a copy of the entire registry database. The administrator does not need to configure anything; the replica databases are automatically created and updated when the replica server is configured.

2.2.1.2 CDS Replication

The CDS database is called a clearinghouse. It is tree structured and has directories which can contain further directories or leaf objects. Replication is defined on a per-directory basis. Each copy of a directory is called a replica. All copies of a certain directory build its replica set. One of these replicas is the master replica; the others are read-only replicas.

Since replication is on a directory level, the CDS database is a distributed database. The master replicas of all directories in the namespace tree can be distributed over several clearinghouses.

In order for replication to happen, administrators must define every detail. They must explicitly create replicas, define the replica set with a master, and define the skulking intervals. Skulking is the process of copying a directory's content to all read-only replicas.

2.2.1.3 DFS Replication

DFS administers its own namespace. In the CDS namespace, it is just known as a junction. It uses a special global path name that follows the global naming convention to locate the binding information for a DFS name server. So the root of the DFS cell file system **/:** is resolved to the global name **/.../**<cell_name>/fs, which is an object in CDS.

The DFS name server is actually called the DFS Fileset Location Database (FLDB). It stores the location (the DFS file server) of all filesets in DFS.

When a DFS client wants to access a file, it first has to contact the FLDB to ask for a DFS file-server address. This involves a read access to the FLDB. Then, depending on the type of access desired, you either get the address of the file server with the master copy of the file or the address of one of the replica servers with a read-only copy of the fileset.

The FLDB and fileset data can be replicated.

Replication of the FLDB is achieved by just adding another FLDB server. Nothing more can be configured. The FLDB servers organize themselves. By means of internal library routines (called the *ubik* library), they determine a master server. The master server holds the master database, and the ubik routines update the databases of the slave servers. All servers hold the entire FLDB.

DFS data is replicated on a fileset level. One copy is the read/write copy and others are read-only copies. Updates from the master copy to the replicas are done on a file basis; in other words, when anything is changed in a file, the whole file is transmitted to every read-only fileset. The administrator determines the frequency with which the updates are performed.

Furthermore, DFS provides a fileset backup server that can also be replicated.

2.2.2 Server Selection Mechanisms

The clients all use a random server selection to access services that can be replicated as described in the previous section. If a DCE client or even another server needs access to one of these services, they call CDS for an address, a binding handle. These calls go to a CDS object which is an RPC group entry. The RPC group contains a list of servers with the same capabilities. In general, a requester can get all these addresses or one after the other. Depending on the call used, the sequence is either in the order the binding handles are stored in CDS, or it is a random order. All the above DCE servers use the random method. This is basically adequate to provide load balancing, but it can introduce a performance penalty if a server is connected via a slow WAN.

This is why special care must be taken for the choice of locations for replica servers.

Some services allow specifying preferences for a specific server. This option can be used if you have to implement replication over a WAN.

However, specifying certain preferences means manual configuration. Before you make use of this option, you should estimate the potential benefit. Only if access to a server is mostly read access and the service is accessed very frequently would you care where the calls go.

2.2.2.1 Security Service

The security service calls have a fallback method for locating the security service if CDS is not available. The binding handles of all security servers are stored in the file /opt/dcelocal/etc/security/pe_site.

If an environment variable `BIND_PE_SITE` is defined, the calls to the security API bypass CDS and get the binding information from that file. However, this requires manual configuration (editing) of this file on all client machines. See “Security” on page 128 for a discussion on this topic.

2.2.2.2 CDS

CDS has no option to bias the cds_clerk on which clearinghouse it should use. The `cdscp set cdscp preferred clearinghouse` command is only used for the `cdscp` command itself. However, the CDS clerk knows which clearinghouses are on the same LAN, and it always tries this clearinghouse first, whenever possible.

2.2.2.3 DFS

New in this release of DFS (2.1) is that the DFS clients follow the same algorithm to select an FLDB server as they used to select a DFS file server in the previous release (1.3). A `-fldb` flag has been added to the `cm setpreferences` command to allow administrators to set priorities. It is recommended that the number of the FLDB servers should be an odd number to ease voting process for a master.

The DFS client can specify a preferred file server or even a list of preferred file servers with priorities. The command is `cm setpreferences`. If the user does not specify priorities, the cache manager assigns default priorities according to whether there is a file server on the local machine, in the same subnetwork, in the same network, or in a different network. The lowest number has the highest priority.

2.2.3 Login Integration

Login integration is designed to present the typical end user with a single system image, rather than a separate image of a local operating system and a remote DCE system. This facility is available with AIX/DCE 2.1 on AIX 4.1.3+. Since this facility allows administrators to integrate all users or only particular users, care should be taken in terms of:

- Unavailability of DCE

  In effect, when DCE fails or is not available, the administrator can decide for all users, or for each user, which authentication protocol to use. If a local authentication protocol is used, then users must have an account on the machine that they are logging into.

- Protection of local resources

  The local administrator should also be aware about protecting local resources from wandering users. In this case, synchronization between AIX accounts and DCE accounts should be considered.

For more details on this subject, see 7.4, “Integrated Login AIX and DCE” on page 265.

2.3 Sizing Guideline

There are two aspects to this: static sizing (″How much resource do I need on my servers to support X users and Y client machines?″) and dynamic sizing (″How much resource do I need if Z users are running applications generating N requests per second?″).

2.3.1 Static Sizing

For the case of static sizing, any disk growth will take place in /var. Each extra user will take a little bit more registry space. Some quick experiments with adding a thousand principals and accounts show a pretty linear growth of 960 bytes per additional user in disk space requirements. Memory usage was less straightforward, presumably because the storage mechanism is a Btree, but at the upper end, the dominant factor seems to be a linear increase of about 3-4 KB per additional user.

Adding more client machines will take additional entries in the namespace. For each client, there is a new directory under /.:/hosts and four entries in that directory: cds-clerk, config, self, and profile. Some quick experiments adding the entries for a thousand client machines show a pretty linear growth of 1.4 KB per additional machine in disk space requirements. Memory usage increased by about 1.7 KB per additional machine. Remember, cdsd logs all changes as transactions and checkpoints them daily or when you shut down the server. You need to have enough room in /var/dce for two copies of the checkpoint file (old and new) plus the transaction log.

Additional clients are not the only contributors to namespace entries, though. The primary user of CDS is applications. So the size of your namespace is going to be very dependent on the applications that you develop and run. The highest we’ve taken these experiments so far has been 100,000 users and 100,000 client machines. The only limitation you’re likely to run into is with the registry since each security server holds the entire registry in memory. Thus,

the main limitation will be how much virtual memory you can make available for the security servers.

2.3.2 Dynamic Sizing

Dynamic sizing is a more difficult question. As an example, in a typical question:

> We are responding to a request for information (RFI) which requires a single DCE-configured cell to serve a potential of 100,000 registered users and 20,000 concurrent users. Each active user is expected to use between 1 to 2 transactions per second (TPS) with the servers in the cell. Some of my questions are:
>
> 1. How do I determine the number of machines to use in this cell?
> 2. What benchmarks should I use for the machines sizing to support the 20,000 concurrent users? Is tpcA (Sybase C/S) a good indicator?
> 3. Is there any information available that tells you how to size for DCE applications?

The 100,000 registered users is a static number. The 20,000 concurrent users using 1-2 TPS is a dynamic question. The answer is going to depend very much on what the applications are doing. For example, if your users all dce_login once in the morning, then start up a long-running application client that makes one CDS lookup to find its server and then uses that server for the rest of the day, the security and CDS servers are only going to see one request apiece from each of those 20,000 users per day. If, however, the users start a new client for each transaction, then the CDS server will be seeing 20,000 requests per second. That's a big difference.

There's also the application servers themselves. The sizing of these is going to depend on how heavyweight the transactions are. For example, compare the TPC-A benchmark, where numbers are expressed as transactions per second, to the TPC-C benchmark, where numbers are expressed in transactions per minute. A server is going to be able to support several orders of magnitude more TPC-A clients than TPC-C clients.

So, dynamic sizing is going to take several steps. First, you need to figure out what load your application will be putting on the security and CDS servers, and from that, figure out the server resources required. For a very rough estimate, assume a model 95 OS/2 or a model 520 AIX CDS server can handle about 500 requests per second and scale upward to the size machine you want to use. Since CDS keeps its directory in memory, it should be CPU-bound and scale about the same as standard benchmarks. We don't have any good numbers on the security server, but it tends to have a lot less interaction with applications.

2.4 Planning the User Namespace

Users working on a system are identified by means of their user ID (UID). All activities of users are associated with their UID and can be backtracked if accounting and audit features are configured accordingly. Access to files or permission to run a certain process are granted to certain users and groups. UNIX traditionally only distinguishes access rights for the owner, the primary group, and others. But with higher security requirements, customers tend to deploy Access Control Lists (ACLs), which allow a much finer granularity of access control.

The flexibility of ACLs has its price. ACLs need a lot more administration. This is where groups come into play. It is good practice to define only groups (and not single users) into ACLs of objects. The advantage is that the number of ACL entries is lower, the ACL itself is more static, and the granting of rights to users is much easier, because they have to be added to or deleted from groups.

The purpose of this discussion is to show the need for careful planning of the user and group namespace. Before you define users and groups, you should decide on:

- Security policy for *all* objects in the DCE environment
- Present and future cell layout within the entire company
- Potential number of users
- Login integration
- Amount of intercell access, in the case of multiple cells

If a company decides to implement multiple cells, the user names and UIDs should be unique across these closely related cells. This will make the job of joining cells much easier, should that ever be necessary. If there is a lot of intercell access, having unique UIDs is also useful with DFS so that a company-wide unique user name is shown as the owner of files when a directory is listed.

2.5 Planning the CDS Namespace

The Cell Directory Service (CDS) is a distributed, replicated database service. It is distributed because the information that forms the database is stored in different places. CDS consists of a hierarchical set of names called the namespace. Each name has a set of associated attributes. Given a name, its associated attributes can be looked up through CDS. For example, given the name of a print server, the directory server can return its location. This information is kept in the clearinghouse. A clearinghouse is a physical CDS database, a collection of directory replicas. By replicating a particular directory in different clearinghouses, you can increase the accessibility as well as the availability of information. On the other hand, the more replicas we have on different systems, the more complex the cell becomes.

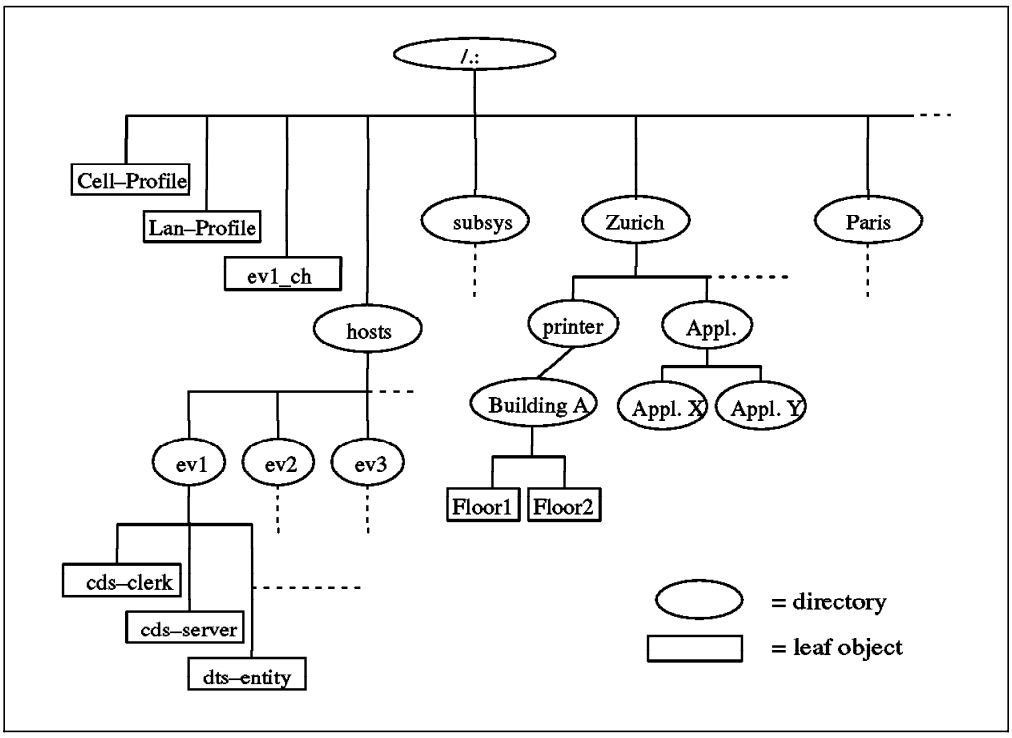


Figure 10. Example of a Cell Namespace

Based on the experiences with our scenarios, we can give you the following recommendations for building a CDS namespace:

1. Keep it simple! Start with one centralized clearinghouse that contains all master replicas and another one that contains all read-only replicas. This ensures the availability of the namespace.
2. Make your first hierarchy (right after root) location-dependent. Put all objects or directories particular to a specific location in the location-dependent directory. This allows you to have further clearinghouses created at the locations, if necessary, which can help to improve accessibility of DCE services at the location sites. If necessary, you can eventually define the master replicas of these directories on the locations where they are mostly accessed. This is useful when many write accesses occur from only one location.
3. Do not use soft links from one location to another. This makes you very dependent on other locations, which also means that it becomes more complex to manage a large cell.

2.6 Planning for Migration

When migrating an existing cell to a newer DCE version, you should consider the two following strategies. Which strategy you choose depends on the degree of availability you want to maintain for end users. The two strategies are:

- Migration of all machines in your cell at once

  This can be done easily if your cell is not large and the availability is not crucial. Save your environment and systems, and plan a weekend to migrate. Tools and hints are provided in this book to facilitate this task.

- Migration one-by-one

This strategy is the most recommended since the newer version of DCE can coexist with an older one. You can plan to migrate at your pace. And the most important thing to consider is the availability of the end users. The same tools and hints are also provided for a smooth migration.

For more details on this subject, see 6.1, “Migrating a DCE 1.x Cell to DCE 2.1” on page 140.

2.7 Conclusions and Planning Tips

This section summarizes the experiences we made while working with the different scenarios and provides recommendations for cell layout with respect to possible customer requirements, technical facts and geographical aspects (network topology).

User requirement considerations:

- Performance
- Availability
- Security
- Cost

Technical implementation considerations:

- Server selection mechanisms
- Replication capabilities

Geographical considerations:

- Business data and service flow
- Reliability of the network
- Transmission speeds and bandwidth

Combining all these factors and trying to come up with an optimal solution is not an easy task. As with many other decisions, it will result in a compromise. So, we cannot suggest concrete solutions, but we can give general recommendations and point out consequences of certain decisions.

2.7.1 One Cell or Multiple Cells?

One of the most important things you have to think about when you start to implement DCE at your company is the work and data flow of your business. This is dependent on:

- Structure of the company
- Existence of remote locations, called branches hereafter
- Type of business
- Size of the branches
- Number and kind of network services the branches need
- Business data flow
- Data and service access needs between branches and main site
- Data and service access needs from branch to branch
- Amount and frequency of such data access

There is no general recommendation for designing a DCE environment. Roughly we can say, if your company concentrates on one business area with a high degree of dependencies between its departments, it could be a good candidate for a one-cell scenario (for example, banking). The opposite of that could be a

company that is working in different business areas with many independent departments (for example, a university).

2.7.2 Tips for Service Layout and Application Design

What a DCE user perceives is how applications perform and whether they are reliable as far as availability is concerned. This actually depends on many different factors:

- The application's architecture and implementation in DCE
- The robustness of the application (replication)
- How the application copes with more users and data
- How the application uses the DCE core services
- How often an application uses the DCE core services
- The speed of the network connections
- The robustness of the network
- The robustness of the DCE core services (replication, availability)

The term *robustness* stands for the ability to deal with error conditions, provide high availability and optimize performance.

Once the decision is made about the number of cells a company is going to need, each cell has to be designed. For that decision, we believe that performance and availability aspects are the main issues, which means ease of use and reliability. Another high priority is ease of administration.

So, technical issues are the deciding factors for the cell layout rather than business needs. From a user perspective, it is not so relevant where the services or data are, but that they are easily and reliably accessible. However, business needs are the most important factors for DCE application design. Applications have to serve a certain business structure and have to make use of the technology to optimally achieve that. The application implementation should be such that installations can follow the same layout rules as for the core services outlined below. Last but not least, the application should be able to respond to growth; it should be scalable.

The following sections discuss each of the technical aspects listed above.

2.7.2.1 The Application's Architecture and Implementation in DCE

Applications are mainly designed according to business needs and data flow. The implementation should make the best use of DCE technology to provide the necessary performance, availability and scalability.

2.7.2.2 The Robustness of the Application (Replication)

Applications should implement replication of their servers, whenever possible. This increases performance and availability. If the application involves data access, replicated or distributed data storage is necessary. This requires coordination among the replicated application servers. For that, it is important whether data access is mostly read or write.

For a data access model, there are different solutions possible. They range from a networked file system like DFS, as a very simple model, to a state-of-the-art three-tier transaction model with a powerful database as the backend. See also 1.1.2, "Three-Tier Client/Server Model" on page 4.

2.7.2.3 How Does the Application Cope with More Users and Data?

An application is scalable when the administrator can install additional instances of the same service and the load from the clients is equally shared between all of the servers. This is what is described in the above item on replication.

2.7.2.4 How Does the Application Uses the DCE Core Services?

The application server should export its interfaces to CDS when it starts and remove them when it ends. This allows for load balancing, provided that replication is implemented. On the other hand, it saves clients from nasty time-outs, which can happen if a server stops but its interface information is still available from CDS.

The client side of the application should, in turn, be ready to try another server address when receiving a communication error because one server is unavailable. If this happens, the application client should immediately request new binding information from the CDS database rather than from the CDS clerk cache, thereby forcing a refresh of the clerk cache. That also saves other clients on the same machine from getting the same invalid binding information.

Furthermore, the application should provide a configuration option for the clients to declare a preferred server location. This is important in cells which involve slow communication links where you want to prevent RPC calls from using too many of these.

From a CDS point of view, application servers should use RPC groups, which are able to provide a random selection of server interfaces to application clients. Application clients should select the servers that make sense; that is, use servers on the LAN as opposed to going across a WAN to get to the identical server interface. To do so, they should use string bindings and explicit binding handles to be able to inspect the binding information received from CDS, and select the closest one or one according to their configured priorities.

2.7.2.5 How Often Does an Application Use the DCE Core Services?

This question must be answered to find out how sophisticated the layout of the core services needs to be.

If users log in once in the morning and start up one or two long-running applications, the usage of security service and CDS is low. Performance aspects of Security Service and CDS are not critical. You only have to make sure the services are available.

On the other hand, if users log in several times a day and use many different applications, or if a lot of applications are started and stopped again, Security Service and CDS are used frequently. CDS experiences a lot of write access if application servers start and stop. In such a situation, performance is an issue, and the cell layout has to take this into consideration.

2.7.2.6 The Speed of the Network Connections

If you have a company with a main site and several branch offices and you have decided to implement just one cell, the type of network connections have a big influence on the layout of the servers.

If all connections are fast and have enough bandwidth, you do not have to care where and how your DCE service calls travel.

2.7.2.7 The Robustness of the Network

This is actually the key to the availability of the whole DCE cell if remote locations are involved. For any type of topology, you want to make sure that any single part of the cell, which means each LAN, can continue to work if it is separated from the rest of the cell.

You could achieve this to a certain extent by replicating all services into each LAN, which can be a local segment or a distant LAN. This requires a lot of server licenses, is expensive to administer and, if used with slow WAN connections, can decrease performance in the whole cell. You could never guarantee write access to the servers.

We suggest implementing redundant network connections and relying on dynamic network routing for high availability of DCE services. The best solution is a reliable multiprotocol router network. This can offer dynamic routing or even dynamic load balancing or bandwidth assignment. On the low end, you could install a cheap switched line with SLIP which is only used when the primary link fails.

If you have a robust network, you can concentrate on performance issues for the cell layout.

2.7.2.8 The Robustness of the DCE Core and DFS Services

Or in other words: How good is the design of the DCE cell?

From an availability point of view, you would have to replicate all services to each separate local or distant LAN. From a performance point of view, this would lead to calls travelling all over the cell and slowing down operation of the whole cell, especially if slower links are involved.

We suggest implementing a reliable network to achieve high availability and concentrating on performance issues for the layout of DCE services in the cell. If there are no slow links, you can just replicate services to achieve load balancing. Note that all remote locations connected with fast links (fiber links with Asynchronous Transfer Mode (ATM) or Fiber Channel Standard (FCS) protocols) are not considered as remote in the following discussion.

If the cell topology includes slow communication links and replication of services is required across such links, the replication capability of each component has to be considered separately.

In the following sections we want to look at replication to locations connected via slow WAN links. We discuss the replication capability of each DCE component and how preferred servers can be defined to avoid unnecessary calls over the slow network connections.

CDS: See also 2.5, “Planning the CDS Namespace” on page 26. CDS supports replication on a directory level. CDS clients have no option to set a preferred CDS server. The CDS clerk knows which clearinghouses are in the same LAN and tries to access these whenever possible. If data is not available in the same LAN, the call randomly goes to any server that has the requested data. The only possibility we have to control where the calls go is to make sure that the requested data is in the location from where it is requested most often.

Start off with a simple CDS configuration. Design the namespace so that objects or directories are grouped together by location, if possible. Keep all master

replicas in one clearinghouse, which means do not distribute the CDS. Replicate the CDS only in the main site and never over slow links.

If you experience performance problems because of this CDS configuration, you can install secondary CDS servers in the remote location that has the problem and install replicas of these location specific directories. Directories that are used by all locations should not be replicated to a remote location, because all locations not having a replica would randomly access this remote location over the slow link.

If you want to avoid any calls for location-specific objects to the central site, move the particular master replica to the remote site. If you want to have a copy of it in the central site for backup purposes, you can create a separate clearinghouse in the central site, create a read-only replica in it, and disconnect it to prevent it from being regularly used. To refresh this backup clearinghouse, you would have to connect this clearinghouse regularly and trigger a skulk from the master copy.

Security: Start with a security server on the same machines as the CDS servers so that these two core components can optimally work together.

The security server database is replicated as a whole. Start replicating it in the central site only. If Security Service access becomes a performance bottle neck, you might consider replicating it to large remote locations. But as soon as you do that, you want to make sure security access calls go to the locations *you* want. So you have to work with the pe_site files as described in 2.2.2.1, “Security Service” on page 23.

DFS FLDB: If the DFS FLDB is replicated, the DFS clients can now be biased as to which FLDB server they should try to use. The FLDB-internal ubik routines determine among multiple copies of the database which one is the master. These routines keep communicating with each other. The optimum number of FLDB servers is three. This keeps the network traffic low between them and yet offers increased performance and availability. Few cells require more than three servers.

Preferably replicate the FLDB in the main site and avoid an FLDB across from a slow WAN link.

DFS Data: The DFS clients can be biased as to which DFS file server they should try to use.

Design the DFS fileset hierarchy so that subtrees of the file system can be grouped by location into filesets. Start off with central DFS file servers. Install file servers in the remote locations as soon as there is demand for them. This might be because of the size of the location or the amount of data accessed from this location or because of the network connection. Of course, in large cells, this might happen on the first day.

Once you start implementing file servers in the remote sites, location-specific files should be located where they are used. If they are used mostly for writing or updating, then the read/write copies should be in the locations. For filesets that are mostly accessed for read, it makes sense to replicate them and have, for instance, one copy in the central site and as many copies as necessary for a good load balancing on the remote sites.

Filesets that are accessed from all remote locations should have their read/write copy in the central site. When they are accessed read-mostly, they can be replicated, and replicas can be in the remote locations as demand requires. To make sure the DFS access calls stay within the remote location, you need to set the preferred file server(s).

Updates of file servers are based on whole files. So, when you have very large files and slow communication links, replication to remote file servers can take a long time, even if only a single byte in that file is changed. While a server is being updated, the data is not available to clients. On the other hand, DFS clients cache data in a finer granularity. In the case of (smaller) remote sites with slow communication links, consider having only DFS clients in that location and no file server. However, if the number of DFS clients is too high, this approach can be disadvantageous. A last resort in this case could be to have a small number of DFS clients running the NFS/DFS translator. All NFS clients use the same cache on the DFS client which they are using as their NFS server. This, however, means that you would also have NFS in your LAN with all its disadvantages.

2.8 Planning Summary

We discuss planning relevant issues in several sections of this book. The following is a list of the most important references:

- 2.7.1, “One Cell or Multiple Cells?” on page 28
- 2.4, “Planning the User Namespace” on page 25
- 2.5, “Planning the CDS Namespace” on page 26
- 2.7.2, “Tips for Service Layout and Application Design” on page 29
- Chapter 5, “Implementing Various LAN/WAN Scenarios” on page 105, performance and availability discussions in each scenario

The above-referenced sections can be summarized into the following planning tips:

- Decision for one or multiple cells

  This decision is dependent on the structure of the company and the type of business it is running. If there is a main site and many branch offices and their business requires a lot of data exchange among each other and they have common data, it is a candidate for a one-cell environment.

- Plan your user namespace

  Use unique user names and UIDs throughout the whole company if you decide to have multiple cells. It is much easier to migrate users from one cell to the other or to join cells should this ever become necessary.

- Plan your CDS namespace

  Begin with a simple non-distributed CDS server layout, which means all master replicas on the same machine. Create location-specific directories that contain all objects or subdirectories that are mostly used in a specific location. This enables you to easily create secondary CDS servers in remote locations and move the master copies of their directories and objects to these servers if you see performance problems with the central CDS server.

- Plan your DFS filespace

  You can choose basically the same approach as for CDS. Create location-dependent directories and fileset mount points that are high up in

the file hierarchy. This makes sure you have shorter path names and their resolution does not have to hop from network to network. Create separate filesets for entities that might have to be moved between locations as a whole, such as users.

- For availability, rely on the network

  Implement redundant links either with a multiprotocol router network or via switched backup lines. The underlying network can have sophisticated dynamic routing capabilities, whereas DCE relies on the error handling implemented by the application programmer.

- Do not export any (slow) WAN interfaces into CDS

  Again, rely on the routing capabilities of the network and exclude all interfaces not associated with a LAN. Use the environment variable `RPC_UNSUPPORTED_NETIFS`. This prevents clients from trying network addresses that might be temporarily unavailable or that do not follow the fastest path.

- Layout of DCE services only for good performance

  If there are no slow links in the cell, you can put replication servers anywhere to achieve load balancing. If you have slow links or pay the links based on data volume, replication has to be planned very carefully. With inadequate replication, you might experience slow performance in the whole cell because server selection is completely at random and may lead to unnecessary calls over the slow links. Each component has its own specific characteristics. For details see 2.7.2.8, “The Robustness of the DCE Core and DFS Services” on page 31.

  As a very simplified summary of Chapter 2, “Planning DCE Cells” on page 19 and Chapter 5, “Implementing Various LAN/WAN Scenarios” on page 105, we suggest the following layout for production cells:

  - In a LAN, install all master servers on one machine and replica servers on one (or more) other machine(s) as in 5.1.1, “Scenario 2: Master Servers on One Machine and Replicas on Another” on page 106.
  - For a production cell that goes across slow links, install and replicate the servers on the main site as described above for a LAN environment.
  - If remote locations are small, install only DCE clients there, but install a secondary communication link for increased availability as described in 5.2.3, “Scenario 6: A Branch Connected with Two Links” on page 130.
  - For larger remote locations where you want to install replicated services, replicate on only those parts of the server databases that are relevant for a specific remote location and define preferred servers, if possible. Each component has its own specific characteristics, for details see 2.7.2.8, “The Robustness of the DCE Core and DFS Services” on page 31.

- Follow the development guidelines when designing DCE applications

  Users work with DCE applications and not as much with the core services. The core services must be well planned and laid out to provide a good basis for the applications. It is very important that the application is designed and implemented properly to provide high performance and availability through replication and to avoid unnecessary time-outs by correctly handling interface exports to CDS. See also 2.7.2.4, “How Does the Application Uses the DCE Core Services?” on page 30.

- Use HACMP/6000 for highly available write access

Put the DCE core services, DFS, or any application server into cluster of systems running IBM High Availability Cluster Multi-Processing/6000 (HACMP/6000) when you need to guarantee write access all times. However, be aware that upon a takeover, DCE is restarted on the fallback system and long lasting client/server connections will be interrupted and may not be able to survive the takeover. What this means is that applications using context handles will lose their context and have to restart also the client side. Write access to CDS is needed when applications export and unexport their interfaces. If they do so, you might be unable to start these applications when the CDS master directory is unavailable. Write access to the security registry is only needed to add or change something, for example accounts. Tickets are granted without write access.

- Plan for a migration

  In the case of having an existing cell, two strategies should be considered when you want to migrate your cell to a newer version of DCE. Depending on your environment, migration can be done for all the machines in the cell at the same time, or it can be done one by one. Beyond the new capabilities provided by the a new version of DCE, you should have in mind that the availability issues for end users have also to be considered.

Note: Most often when talking about availability of server systems, people think of redundant hardware or software solutions, such as HACMP, to handle common hardware failures, such as a disk crash. In real life, however, hardware failures occur very seldom and do not usually keep system administrators very busy. Most server outages are caused by trivial accidents that could easily have been avoided if one had anticipated them. Such trivial things are cables, cable connectors, power connections, physical access to the system, other applications running on the same system, or allowing many interactive users on a critical server system. Thorough planning and installation of a system that runs critical services must therefore include adequate system-management functions and a proper physical installation. It is a good idea having DCE server systems locked up in dedicated system rooms and have no other applications or user accounts running on them.

Chapter 3. Implementing DCE Cells

In this chapter we extensively describe the installation and configuration of a cell that is composed of several distinct hardware and software platforms. We include two versions of AIX (3.2.5 and 4.1.4) and also OS/2 (V3, WARP). In Chapter 5, “Implementing Various LAN/WAN Scenarios” on page 105, we will give short-path instructions to quickly install different scenarios. Besides giving step-by-step installation instructions for different network topologies, we will also discuss performance and availability issues in that chapter.

After we will have installed and configured a basic DCE cell with all its core services, we will augment this cell with the DFS (Distributed File System). The DFS configuration will be explained in the next chapter.

3.1 Overview and Cell Layout

The following DCE and AIX levels are used in the configuration:

- DCE 2.1 for AIX 4.1.4
- DCE 1.3 for AIX 3.2.5
- DCE (BETA 1/21/96) for WARP OS/2 V3

The objective that we achieved is clearly explained in scenario 1 and depicted in Figure 11. The figure also shows what software is used on what platforms, and it indicates the distribution of the DCE functions among the machines in the cell. Besides the minimum required server configuration, we replicate the servers whose reliability has a high impact on the global functioning of the cell.

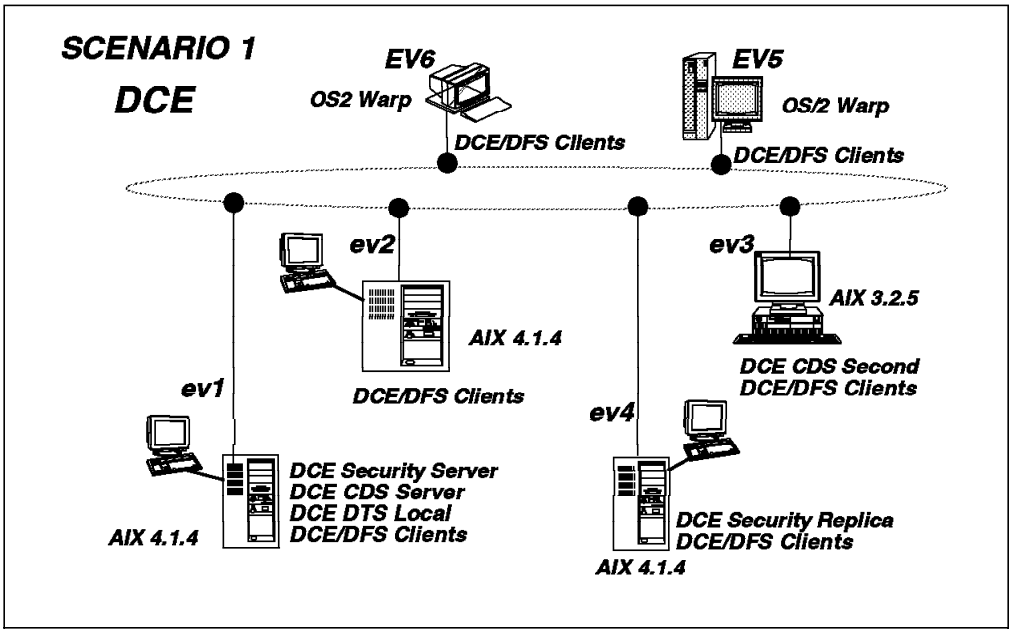


Figure 11. Cell with DCE Components and Related Platforms

According to the tasks that have to be executed to achieve our cell objective, we discuss the following topics:

- Preparing and installing the DCE software for configuration on AIX

- Configuring DCE core servers and clients on AIX
  - Configuring the master security server on host *ev1*
  - Configuring the CDS server on host *ev1*
  - Configuring a local DTS server on host *ev1*
  - Configuring DCE clients on *ev2*
- Preparing and installing DCE code for configuration on OS/2 Warp
- Configuring DCE core clients on OS/2 Warp
  - Configuring DCE clients on hosts *EV5* and *EV6* (OS/2 Warp)
- Configuring server replica on AIX
  - Replication of CDS on host *ev3* (DCE 1.3)
  - Replication of the security server on host *ev4* (DCE 2.1)
- Configuring server replica on OS/2 Warp
  - Replication of CDS on host EV5 (OS/2 Warp)
- Summary and conclusions

3.2 Preparing for DCE Configuration on AIX

The purpose of this section is to guide you through all the necessary preparation steps that are required for all scenarios. These are:

1. Preparing disk space
2. Checking network name resolution
3. Checking network routing
4. Checking the network interfaces
5. Synchronizing the system clocks
6. Language Environment Variable

3.2.1 Preparing Disk Space

Installation and configuration of DCE servers and clients require some reserved disk space that should not be overwritten or used by other components. The safest way to guarantee independence is to create separate file systems. These file systems should be created before DCE is installed, meaning before you execute `installp`.

We also suggest a careful review of the release notes associated with the delivered DCE products to determine the disk space requirements for each DCE component.

1. Paging space

   We suggest that twice the size of the installed RAM or at least 100 MB is allocated for paging space. Display the current settings using the following command:

   ```
   # lsps -a
   Page Space  Physical Volume Volume Group  Size   %Used  Active  Auto  Type
   hd61        hdisk1          rootvg        32MB     76     yes   yes    lv
   hd6         hdisk0          rootvg        32MB     75     yes   yes    lv
   ```

We have a total of 64 MB for paging space. Since more than 70 percent is used, we suggest you increase it. To increase both disks to 60 MB each, we have to add seven partitions of 4 MB:

```
# chps -s'7' hd6
# chps -s'7' hd61
```

2. Disk space for */var/dce*

   The /var file system is used by the operating system to store various files that can grow in size and number, such as the print spool and trace files. On the other hand, DCE also has some files that use more and more disk space, for instance, core server databases and credential files. It is important for AIX and DCE not to interfere with each other.

   The size of this file system actually depends on what is going to be installed on the system. The most current requirements for the specific components can be found in the release notes. Since we had enough disk space, we decided to use 20 MB on all systems to hold all possible server and client code:

   ```
   # crfs -v jfs -g'rootvg' -a size='40000' -m'/var/dce' -A'yes' -p'rw'
   # mount /var/dce
   ```

3. Disk space for */var/dce/adm/dfs/cache* (not required for DCE!)

   Creating this file system is helpful on a DFS client machine to avoid getting stuck. An incorrectly defined cache could fill up the /var/dce file system, or files underneath /var/dce could use up space meant to be reserved for DFS disk cache. The cache to be configured may at most be 85 percent of the actual disk space available. The following example makes room for a 10 MB cache on a 12 MB file system.

   ```
   # crfs -v jfs -g'rootvg' -a size='24000' -m'/var/dce/adm/dfs/cache'
     -A'yes' -p'rw'
   # mount /var/dce/adm/dfs/cache
   ```

4. Other candidates are ***/var/dce/directory*** and ***/var/dce/security***:

   These directories contain the databases of CDS and security service, respectively. If they become very large, separate file systems should be considered.

3.2.2 Checking Network Name Resolution

Some commands expect a hostname as an input parameter. Internally, this name is used to find the Internet address. Be sure that forward and reverse translation is correctly working for all involved systems.

1. Forward resolution. If you use the same hostname for different network interfaces, be sure the name resolves to the primary interface you want to be used. To achieve this, the primary interface must be defined first.

   ```
   # hostname
   ev1
   # host ev1
   ev1.itsc.austin.ibm.com is 9.3.1.68
   # host ev2
   ev2.itsc.austin.ibm.com is 9.3.1.120
   ```

2. Reverse resolution

```
# host 9.3.1.68
ev1.itsc.austin.ibm.com is 9.3.1.68
# host 9.3.1.120
ev2.itsc.austin.ibm.com is 9.3.1.120
```

If it is not working correctly or as expected, it should be fixed. To change the definitions of the system on which you are running the commands (ev1 in the above example), you call the following SMIT menu:

```
# smit tcpip
->  Minimum Configuration & Startup
```

If name resolution of a remote system returns an incorrect value or times out, you must fix the name server database or the /etc/hosts file if you are not running a DNS (domain name server).

3.2.3 Checking Network Routing

Before configuring DCE, make sure that all machines in your network communicate correctly with each other using the TCP/IP protocol. Use the `ping -R -c 1 <hostname>` command to test all connections and visualize the routes that the packets are following.

You can also use the following command to know which route is in use:

```
# netstat -r

Routing tables
Destination      Gateway            Flags  Refcnt Use
Interface
Netmasks:
(root node)
(0)0 ff00 0
(0)0 ffff ff00 0
(root node)

Route Tree for Protocol Family 2:
(root node)
default          ev4                UG          1      104  en0
193.1.10         ev3                U          10    47708  en0
127              loopback           U           1     3479  lo0
(root node)

Route Tree for Protocol Family 6:
(root node)
(root node)
```

Be very careful when using the default routing. Setting a default route prevents the node from doing dynamic routing. The recommendation is to use gated for routing, which is at present the most sophisticated routing daemon.

Most common source of failure

Incorrect routing is the most common source of failure not only in TCP/IP but also in DCE. Even if in TCP/IP you get through to another node, you might experience timeouts in DCE because DCE might first try to use another interface for which no route is available.

In order for a `ping` command to succeed, the route must be accurate in both directions.

3.2.4 Checking the Order of Network Interfaces

The order of the network interfaces determines the way a server's interfaces or binding handles are exported to CDS. You can check the network interfaces with this command:

```
# netstat -i
name  Mtu   Network     Address  Ipkts    Ierrs Opkts  Oerrs Coll
lo0   1536  <Link>               14992     0    14992  0     0
lo0   1536  127         loopback 14992     0    14992  0     0
tr0   1492  <Link>               17654     0    14115  0     0
tr0   1492  9.3.1       ev1      17654     0    14115  0     0
xs0   1500  <Link>                   3     0        3  0     0
xs0   1500  192.1.20    ev1          3     0        3  0     0
```

What is discussed here is not relevant for pure client systems. They do not export any interfaces.

As you can see, *tr0* comes before *xs0*, which is as it should be. When a server exports its interfaces, they are exported in the order they are listed with the `netstat` command. If a client performed a lookup in CDS, he would get the TCP binding handle associated with *tr0* first. Even though all DCE/DFS clients choose their server and binding handles at random, we observed that the first handle was chosen more often. So if you have multiple interfaces which are considerably different in speed, we recommend to change the order so that the fastest is on the top of the list. If you have, for instance, a fast FDDI connection and an Ethernet between the same systems, you probably want to give FDDI a higher probability of being chosen.

To move an interface from the top to the end of the list, you must delete it with `rmdev -dl` and redefine it.

Another option in AIX DCE is the ability to exclude a network interface from ever being exported into CDS. For example, if you want to only use `tr0`, you must set up an environment variable in the /etc/environment file as shown below before you configure any server:

```
 ..
RPC_UNSUPPORTED_NETIFS=xs0
export RPC_UNSUPPORTED_NETIFS
 ..
```

If you exclude *xs0* anyway, the order returned by the `netstat` command is not relevant in this case.

Exclude WAN interfaces

We recommend always excluding the WAN interfaces (X.25 or SLIP). You can always rely on the fast LAN interfaces and TCP/IP routing mechanisms. If a DCE service call actually involves two nodes connected over a WAN connection, it will find its way thanks to IP routing. In this way DCE never tries to connect over a specific WAN connection, but leaves the decision up to IP, which might have sophisticated routing selection mechanisms in place to find the fastest available route.

DCE, by itself, does not have any inherent algorithms. If you do not exclude these interfaces in DCE, you are more likely to experience timeouts because DCE might (randomly) choose a network link that is temporarily unavailable.

Even if you do not have redundant network links right now, you might put a sophisticated router network in place later on. It is much easier to implement if you do not have to get rid of unwanted binding information in the whole CDS.

3.2.5 Synchronizing the System Clocks

DCE services rely on highly synchronized time. If, for instance, the clock value of a client system requesting a ticket differs too much from the security server's clock, no ticket is granted. The first time this may happen is when you configure a DCE client.

It is very important that you start with synchronized clocks. Issue the `setclock` command on all systems to get one specific system's clock value, and use that on all other systems. For instance, to set the clock on *ev2* according to *ev1*'s clock, issue the following command on *ev2*:

```
# setclock ev1
```

Run setclock on DCE clients

It is good practice to run a setclock-equivalent command on DCE client systems that are regularly powered-off over nights, weekends, vacations, and so on before you start DCE on them.

3.2.6 Language Environment Variable

Problems can arise during installation and/or configuration if the `LANG` environment variable is not set to `C` or `En_US`. Some shell scripts in AIX DCE have hard-coded checks on English-language character strings. Also, our shell scripts provided on the diskette have not been tested in another language environment.

3.3 Installing the DCE Code on AIX

Installing DCE is the procedure of loading the software onto the harddisk. Call `smit installp` and choose the appropriate program objects to install.

The point we want to make here is that installation is a separate step that must be executed *after* all preparation steps, particularly after all the necessary file

systems have been created and mounted. If you have not done this yet, go back to 3.2, “Preparing for DCE Configuration on AIX” on page 38.

Check with the release notes whether certain PTFs are required.

If you are upgrading DCE in an existing cell, you do not have to unconfigure and reconfigure the cell. Perform the following steps:

1. Stop all DCE services
2. Install the new DCE release for *all* DCE components
3. Reboot the machine
4. Restart DCE

The DCE configuration and the databases are preserved. However, we recommend you back up all your DCE databases prior to the upgrade. See 6.3, “Backup/Restore and Other Housekeeping Tasks” on page 166 for details.

3.4 Configuring the Initial DCE Servers and Clients on AIX

This section describes how to initialize a cell with the core services. Here are the steps to follow:

1. Configure the security server
2. Configure the CDS server
3. Configure the DTS server

Installation of all basic *core* components will be done on host *ev1.* We use the cell name *itsc.austin.ibm.com*, which is also the name of the IP domain.

We could use both the ASCII interface (smitty) or the graphical interface for SMIT to configure the components. For documentation reasons, we chose the ASCII interface to perform all client and server configurations on AIX. You can navigate through the different SMIT screens step by step, or you can call SMIT with a fastpath, which brings you automatically to the correct configuration panel.

Most configurations can also be executed from the command line with the mkdce command. To get help on this command, just call it without arguments or call the DCE man page in the following way (on AIX 4.1):

```
# dceman mkdce
mkdce

Purpose

 Configures DCE components.

Format

 mkdce [-n cell_name] [-h dce_hostname] [-a cell_admin] [-s
 security_server] [-c cds_server]
 [-P min_principal_id] [-G min_group_id] [-O min_org_id] [-M
 max_UNIX_id] [-p profile]
 [-t time_courier_role] [-R] [-r sec_rep_name] [-o full | -o local | -o
 admin -i machine identifier]
 component ...
```

3.4.1 Configuring the Initial Security Server on AIX

The first role that has to be configured in a cell is the initial *security server*. Later, it can be replicated on another platform. Call SMIT and follow the indicated path, or call SMIT with the fastpath name:

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Servers
         -> SECURITY Server
            -> 1 primary
               (fastpath = mkdcesecsrv)
```

```
                                SECURITY Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* CELL name                                           [/.../itsc.austin.ibm.com>
* Cell ADMINISTRATOR's account                        [cell_admin]
  Machine's DCE HOSTNAME                              [ev1]
  PRINCIPALS Lowest possible UNIX ID                  [100]
  GROUPS Lowest possible UNIX ID                      [100]
  ORGANIZATION Lowest possible UNIX ID                [100]
  MAXIMUM possible UNIX ID                            [32767]


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Selecting **Enter=Do** starts the configuration of the master security server. You will notice by the progress list that besides the sec_srv, two additional components, the *RPC Endpoint Mapper* and the security client sec_cl are configured. In reality, the sec_cl is included in the dced process, which encompasses also the endpoint mapper function, while the sec_srv is instantiated in the secd daemon process. The following list shows the progress report during the configuration:

```
Password to be assigned to initial DCE accounts:
Re-enter password to be assigned to initial DCE accounts:

Configuring RPC Endpoint Mapper (rpc)...
RPC Endpoint Mapper (rpc) configured successfully
Configuring Security Server (sec_srv)...
Password must be changed!
Configuring Security Client (sec_cl)...
Security Client (sec_cl) configured successfully
Security Server (sec_srv) configured successfully

Current state of DCE configuration:
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
sec_srv      COMPLETE   Security Server (Master)
```

The following command has the same effect:

```
# mkdce -n itsc.austin.ibm.com sec_srv
```

3.4.2 Configuring the Initial CDS Server on AIX

At this point, you can start to configure a CDS server function. This can be done on the same machine as the security server machine or on another machine in your network. If you configure the CDS server on another machine, you have to make sure that the route between the security server machine and the machine where you are going to configure the CDS server are set up correctly. Call SMIT and follow the indicated path or call SMIT with the fastpath name:

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Servers
         -> CDS (Cell Directory Service) Server
            -> 1 initial
               (fastpath = mkcdssrv)
```

```
                      CDS (Cell Directory Service) Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* CELL name                                          [/.../itsc.austin.ibm.com>
* SECURITY Server                                    [ev1]
* Cell ADMINISTRATOR's account                       [cell_admin]
* LAN PROFILE                                        [/.:/lan-profile>
  Machine's DCE HOSTNAME                             [ev1]


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

The CDS server will reside on the same platform as the security server. An **Enter=Do** starts the configuration. You will notice in the progress report that besides cds_srv, the CDS client, cds_cl, also gets configured. The cds_srv function is contained in the cdsd daemon, while the cds_clerk is part of the CDS advertiser process, cdsadv. This is the progress list:

```
Enter password for DCE account cell_admin:

Password must be changed!
Configuring Initial CDS Server (cds_srv)...
Configuring CDS Clerk (cds_cl)...
Waiting (up to 2 minutes) for cdsadv to find a CDS server.
Found a CDS server.

        Initializing the namespace ...
           Modifying acls on /.:
           Creating /.:/cell-profile
           Exporting cds-clerk and cds-server attributes
           Modifying acls on /.:/subsys/dce/sec
           Modifying acls on /.:/cell-profile
           Modifying acls on /.:/lan-profile
           Modifying acls on /.:/hosts
           Modifying acls on /.:/sec
           Modifying acls on principal ...
           Modifying acls on principal/krbtgt ...
           Modifying acls on principal/hosts/ev1 ...
           Modifying acls on group ...
```

```
          Modifying acls on group/subsys ...
          Modifying acls on group/subsys/dce ...
          Modifying acls on org ...
          Modifying acls on policy ...
          Modifying acls on /.:/sec/replist
          Modifying acls on /.:/ev1_ch

Initial CDS Server (cds_srv) configured successfully
CDS Clerk (cds_cl) configured successfully
Current state of DCE configuration:
cds_cl       COMPLETE   CDS Clerk
cds_srv      COMPLETE   Initial CDS Server
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
sec_srv      COMPLETE   Security Server (Master)
```

This CDS server configuration could have been done via the command line:

```
# mkdce cds_srv
```

If you wanted to install the CDS server on another system in the same LAN, you could run this command:

```
# mkdce -n itsc.austin.ibm.com -s ev1 cds_srv
```

3.4.3 Configuring the DTS Server

DCE cell functioning is highly reliant on time. It is advisable to configure enough "time distributors" in the cell. We will configure a *local DTS* role on *ev1*. Call SMIT and follow the indicated path, or call SMIT with the fastpath name:

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Servers
         -> DTS (Distributed Time Service) Server
            (fastpath = mkdtssrv)
```

```
                                DTS Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                      [Entry Fields]
  Type of SERVER                                       local       +
  Type of COURIER                                      noncourier  +
* CELL name                                           [/.../itsc.austin.ibm.com]
* SECURITY Server                                     [ev1]
  CDS Server (If in a separate network)               []
* Cell ADMINISTRATOR's account                        [cell_admin]
* LAN PROFILE                                        [/.:/lan-profile]
  Machine's DCE HOSTNAME                              [ev1]
```

The following shows the progress list:

```
Enter password for DCE account cell_admin:
Configuring Local DTS Server (dts_local)...
Local DTS Server (dts_local) configured successfully

Current state of DCE configuration:
cds_cl       COMPLETE   CDS Clerk
cds_srv      COMPLETE   Initial CDS Server
dts_local    COMPLETE   Local DTS Server
```

```
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
sec_srv      COMPLETE   Security Server (Master)
```

Normally, we need to have at least three DTS server machines per LAN in a cell. Remember, you cannot configure a DTS server on a machine where a DTS client is already configured; you first have to unconfigure the DTS client and then configure the DTS server.

This DTS server configuration could also have been done via the command line:

```
# mkdce dts_local
```

3.4.4 Configuring Multiple Servers at the Same Time

We configured all servers, one after the other, on the same platform. When using the mkdce command from the command line, you can specify all servers on the same machine at once:

```
# mkdce -n itsc.austin.ibm.com sec_srv cds_srv dts_local
```

3.4.5 Configuring an AIX DCE Client

As we have seen, when we configure a DCE core server, the client part is automatically configured. Other clients must be explicitly configured.

In DCE 1.3 and 2.1, we have the ability to split the configuration process; The part that requires write access to CDS and the security server can be done centrally by the cell administrator. In fact this step prepares the server machine(s) to accept new clients machine in the cell.

The system administrator of a client machine need not know cell_admin's password to configure his machine into the DCE cell.

The split configuration feature also includes split unconfiguration. This actually enables large-scale, central DCE administration.

We propose two methods here. You can choose the method you want:

- Full configuration

  This method performs all configuration steps only on the client machine, which requires cell_admin's password. This means the DCE administrator has to do everything or give away the password. As long as all machines are in the same (trusted) LAN, the cell_admin can remotely log in to each client and perform the configuration.

- Split configuration

  This method is very helpful. You will not be constrained with space and time. The cell_admin can preconfigure DCE client machines. Each DCE client machine can be configured simply by a local system administrator at their convenience without knowing cell_admin's password.

Finally, we discuss our experiences with the split configuration.

3.4.5.1 Full Configuration Method

We assume you are logged in to the machine where you will configure DCE clients. Call SMIT and follow the indicated path, or call SMIT with the fastpath name:

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Clients
         -> 1 full configuration for this machine
            (fastpath = mkdceclient)
```

```
                        Full DCE/DFS Client Configuration

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                [Entry Fields]
* CELL name                                     [/.../itsc.austin.ibm.com]
* CLIENTS to configure                          [rpc sec_cl cds_cl dts_cl]
* SECURITY Server                               [ev1]
  CDS Server (If in a separate network)         [ev1]
* Cell ADMINISTRATOR's account                  [cell_admin]
* LAN PROFILE                                   [/.:/lan-profile]
  Client Machine DCE_HOSTNAME                   [dce_ev2]
     The following fields are used
     ONLY if a DFS client is configured
* DFS CACHE on disk or memory?                  [disk]                  +
* DFS cache SIZE (in kilobytes)                 [10000]
* DFS cache DIRECTORY (if on disk)              [/var/dce/adm/dfs/cache]
```

```
Password for DCE account cell_admin:

Configuring RPC Endpoint Mapper (rpc)...
RPC Endpoint Mapper (rpc) configured successfully

Configuring Security Client (sec_cl)...
Password must be changed!
Security Client (sec_cl) configured successfully

Password must be changed!
Configuring CDS Clerk (cds_cl)...
Waiting (up to 2 minutes) for cdsadv to find a CDS server.
Found a CDS server.
           Modifying acls on hosts/dce_ev2
           Modifying acls on hosts/dce_ev2/self
           Modifying acls on hosts/dce_ev2/cds-clerk
           Modifying acls on hosts/dce_ev2/profile
           Modifying acls on /.:/lan-profile
CDS Clerk (cds_cl) configured successfully
Configuring DTS Clerk (dts_cl)...
DTS Clerk (dts_cl) configured successfully

Current state of DCE configuration:
cds_cl       COMPLETE   CDS Clerk
dts_cl       COMPLETE   DTS Clerk
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
          Press Enter to continue
```

This following command has the same effect:

```
# mkdce -n itsc.austin.ibm.com -h dce_ev2 -s ev1 all_cl
```

At this point, all DCE core clients are configured. See Chapter 4, “Implementing DFS” on page 75 to understand how to configure a DFS server machine and a DFS client machine.

3.4.5.2 Split Configuration Method

With this type of configuration, we have to consider two steps:

- What is to be done by cell_admin?
- What is to be done by a local system administrator?

Administrator Part: The following are the tasks that need to be performed by the cell_admin from any machine already configured in the DCE cell. We assume we are on machine *ev1* and want to preconfigure the DCE client machine, *ev2*:

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Clients
         -> 3 admin only configuration for another machine
               (fastpath = mkdceclient)
```

```
                    Administrator DCE Client Configuration

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                          [Entry Fields]
* CLIENTS to configure                    [sec_cl cds_cl] +
* Cell ADMINISTRATOR's account            [cell_admin]
  Client Machine DCE_HOSTNAME             [dce_ev2]
* Client Machine IDENTIFIER               [ev2]
* LAN PROFILE                             [/.../itsc.austin.ibm.com/lan-profile]
```

```
Enter password for DCE account cell_admin:
Configuring Security Client (sec_cl) for dce_host dce_ev2 on
  machine ev2 ...
Password must be changed!
Completed admin configuration of Security Client (sec_cl) for
  dce_host dce_ev2 on machine ev2
Configuring Security Client (sec_cl) for dce_host dce_ev2 on
  machine ev2 ...
Completed admin configuration of Security Client (sec_cl) for
  dce_host dce_ev2 on machine ev2
Configuring CDS Clerk (cds_cl) for dce_host dce_ev2 on
  machine ev2 ...
           Modifying acls on hosts/dce_ev2
           Modifying acls on hosts/dce_ev2/self
           Modifying acls on hosts/dce_ev2/cds-clerk
           Modifying acls on hosts/dce_ev2/profile
           Modifying acls on /.:/lan-profile

Completed admin configuration of CDS Clerk (cds_cl) for
  dce_host dce_ev2 on machine ev2

Cell administrator's portion of client configuration has completed
```

```
successfully. Root administrator for ev2 should now
complete the client configuration on that machine.
        Press Enter to continue
```

You must specify two machine names, which was introduced with DCE 1.3:

```
  ...
  Client Machine DCE_HOSTNAME            [dce_ev2]
* Client Machine IDENTIFIER              [ev2]
  ...
```

The *DCE_HOSTNAME* is the name under which the machine is known in DCE. It is used for the machine principal name and the CDS entries. The *IDENTIFIER* is the TCP/IP hostname. You can use the same name for both entries. If the DCE hostname is not specified, the TCP/IP hostname is used. Pay attention to the output display of the command, and notice what DCE hostname is generated to make sure you use the same when you configure the client part. However, we recommend always explicitly specifying both names (`-h` flag and `-i` flag) to avoid problems. See also 7.1.1.2, “Split Configuration” on page 242.

Only sec_cl and cds_cl can be preconfigured with this method. Neither dts_cl (for DTS client) nor dfs_cl (for DFS client) is proposed in the menu, but it is not a problem. They can be configured on the DCE client machine by the local system administrator without having to specify cell_admin’s password.

The same can be achieved with following command (the new flags are highlighted):

```
# mkdce -o admin -h dce_ev2 -i ev2 sec_cl cds_cl
```

At this point, the task of cell_admin is finished. The local system administrator on the DCE client machine can configure their machine at their own pace.

Local Part: The following shows how the local system administrator needs to configure their machine as a DCE client. Call SMIT and follow the indicated path, or call SMIT with the fastpath name:

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Clients
         -> 2 local only configuration for this machine
            (fastpath = mkdceclient)
```

```
                        Local DCE/DFS Client Configuration

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                [Entry Fields]
* CELL name                                     [/.../itsc.austin.ibm.com]
* CLIENTS to configure                          [rpc sec_cl cds_cl dts_cl]
* SECURITY Server                               [ev1]
  CDS Server (If in a separate network)         [ev1]
* Client Machine DCE_HOSTNAME                   [dce_ev2]
     The following fields are used
     ONLY if a DFS client is configured
* DFS CACHE on disk or memory?                  [disk] +
* DFS cache SIZE (in kilobytes)                 [10000]
* DFS cache DIRECTORY (if on disk)              [/opt/dcelocal/var/adm/dfs/cache]
```

On this screen, you can select all clients. After pressing **Enter,** the system displays:

```
Configuring RPC Endpoint Mapper (rpc)...
RPC Endpoint Mapper (rpc) configured successfully

Configuring Security Client (sec_cl)...
Security Client (sec_cl) configured successfully on the
  local machine

Configuring CDS Clerk (cds_cl)...
CDS Clerk (cds_cl) configured successfully on the
  local machine

Configuring DTS Clerk (dts_cl)...
DTS Clerk (dts_cl) configured successfully on the
  local machine

Current state of DCE configuration:
cds_cl       COMPLETE   CDS Clerk
dts_cl       COMPLETE   DTS Clerk
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
          Press Enter to continue
```

At this point, all clients are configured. You do not have to provide cell_admin's password to configure the DCE client machine.

The same can be achieved with the following command (the new flags are highlighted):

```
# mkdce -o local -n itsc.austin.ibm.com -s ev1 -h dce_ev2 all_cl
```

3.4.5.3 Experience with Split Configuration

This feature is easy to use and understand. It is very useful to administer DCE client machine configurations in a large DCE cell. Owners of client workstations can request DCE preconfiguration from the DCE administrator who does the admin part and lets the requester know what was defined. The requester can then configure their own workstation into the cell. The procedure could also be automated in a large DCE cell.

When we used the new feature, we experienced one error situation. We did not specify the DCE hostname. The server part installation was as follows:

```
# mkdce -o admin -i ev8 sec_cl cds_cl
Enter password for DCE account cell_admin:
Configuring Security Client (sec_cl) for dce_host ev8.itsc.austin.ibm.com on
  machine ev8.itsc.austin.ibm.com ...
Password must be changed!
Completed admin configuration of Security Client (sec_cl) for
  dce_host ev8.itsc.austin.ibm.com on machine ev8.itsc.austin.ibm.com

Configuring Security Client (sec_cl) for dce_host ev8.itsc.austin.ibm.com on
  machine ev8.itsc.austin.ibm.com ...
Completed admin configuration of Security Client (sec_cl) for
  dce_host ev8.itsc.austin.ibm.com on machine ev8.itsc.austin.ibm.com

Configuring CDS Clerk (cds_cl) for dce_host ev8.itsc.austin.ibm.com on
  machine ev8.itsc.austin.ibm.com ...

          Modifying acls on hosts/ev8.itsc.austin.ibm.com
          Modifying acls on hosts/ev8.itsc.austin.ibm.com/self
          Modifying acls on hosts/ev8.itsc.austin.ibm.com/cds-clerk
          Modifying acls on hosts/ev8.itsc.austin.ibm.com/profile
          Modifying acls on /.:/lan-profile

Completed admin configuration of CDS Clerk (cds_cl) for
  dce_host ev8.itsc.austin.ibm.com on machine ev8.itsc.austin.ibm.com

Cell administrator's portion of client configuration has completed
  successfully. Root administrator for ev8.itsc.austin.ibm.com should now
  complete the client configuration on that machine.
```

Nothing wrong is reported and everything seems correct, but the client will not install correctly. Since the DCE hostname was not specified, the TCP/IP name was taken which resolves into a full domain name.

The local part of the client installation on *ev8* then failed with an invalid password:

```
# mkdce -n /.../itsc.austin.ibm.com -s ev7 -o local sec_cl cds_cl
Configuring RPC Endpoint Mapper (rpc)...
RPC Endpoint Mapper (rpc) configured successfully

Configuring Security Client (sec_cl)...
Sorry. Password Validation Failure. - Invalid password (dce / sec)
Cannot authenticate as DCE user hosts/ev8/self
Before you reconfigure, your cell administrator must reset
  the password for DCE user hosts/ev8/self.
Current state of DCE configuration:
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       PARTIAL    Security Client
```

The DCE client hostname was omitted (-h ev8). Here, the configuration procedure generated *ev8* for the DCE hostname and tried to authenticate as principal hosts/ev8/self which does not exist. The local hostname was used, which had been set without the domain name.

We recommend to always specify the DCE hostname for both sides of the split configuration even if it is the same as the TCP/IP hostname.

3.5 Installing and Preparing for DCE Configuration on OS/2 Warp

The following steps are required for a successful installation of DCE on OS/2 Warp.

1. Installing the DCE code
2. Verifying the MPTS installation and customization
3. Checking network name resolution
4. Checking network routing
5. Checking the network interfaces
6. Synchronizing the system clocks

In this scenario the OS/2 Warp machine will only be used as a client.

3.5.1 Installing the DCE Code

The DCE for OS/2 Warp version that we used was not a release level. It was built on 01/21/96, and it differs quite a lot from the original DCE Warp Beta release distributed in 1995.

The installation of the DCE code is very simple and depends on the medium on which the code is distributed.

- If the code is delivered on a CD-ROM, simply point to the CD-ROM by its drive letter.
- If the code is available on an NFS server, it will be necessary to set up TCP/IP in order to be able to mount the directory with the DCE installation code and point to the correct directory.
- An equivalent action is required if the code is available on a LAN Server server.

While pointing to the directory with the installable code, enter the command `install` and follow the self-explanatory Presentation Manager (PM) panels. It is best to accept the update of the *CONFIG.SYS* file; otherwise it has to be done manually afterwards. Finally, the procedure will present you a screen with component selections.

Depending on the intentions for this platform, selections have to be made. We decided to install the code for all possible components (DFS client included). This installation is only an unpacking of the code; NO configuration is done yet. The total package (without samples and books) requires about 50 MB of disk space. This may become smaller for the generally available code.

Finally, after the selections for the target drives, the **install** push button activates the copy/download of the DCE package components. After the completion of the download, it will be necessary to boot the system.

3.5.2 Verifying the MPTS Installation and Customization

The IBM DCE for OS/2 Warp supports TCP/IP and NetBIOS protocols. We use TCP/IP to be able to communicate with the AIX machines. However, you could choose NetBIOS between two OS/2 machines. The DCE protocol sequences for NetBIOS are ncadg_nb_dgram (connectionless) and ncacn_nb_stream (connection-oriented).

Before you install DCE, the network must be properly configured and working:

1. Install MPTS (Multi-Protocol Transport Service) for OS/2 Warp
2. Configure your network adapter and protocol stacks in MPTS
3. Configure TCP/IP

The MPTS installation is done, for example, from diskettes with the `A:INSTALL` command. This installs an *MPTS* icon on the desktop. Double-click this icon to configure MPTS.

In the *Configure* window (Figure 12 below), you need to select the **LAN adapters and protocols** (LAPS) to configure the network adapter and protocol drivers you use. Be sure to select the **TCP/IP** protocol. This is done in the *LAPS Configuration* window. Back in the *Configure* window, you must select the **Socket MPTS Transport Access**. Again, be sure to select **TCP/IP**. Then, click on **Configure** and eventually on the **Exit** buttons.

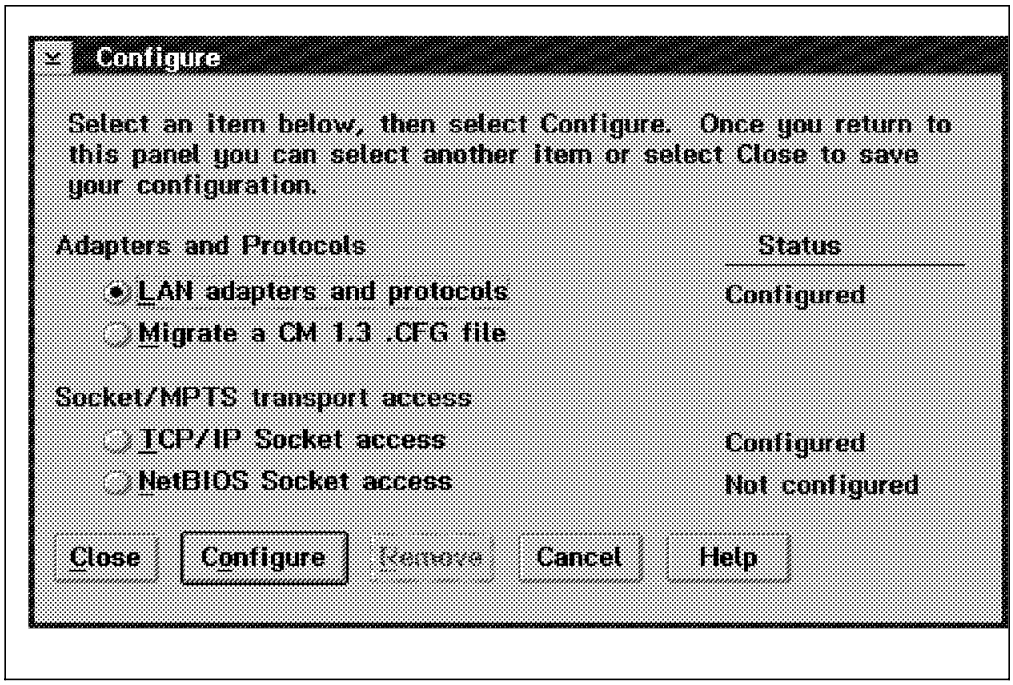


Figure 12. MPTS Configure Window

After executing this task, another system boot will be required to complete the MPTS update and to *unlock* locked files.

To configure TCP/IP, double-click on the **TCP/IP** folder and on the **TCP/IP Configuration** icon within that folder. This brings up the *TCP/IP Configuration* window with a notebook-like interface to fill in the different configuration parameters. On the *Network* page, fill in the IP address and the subnet mask. On the *Routing* page, fill in at least a default route by pushing the **Insert Before** button. On the *Services* page one, fill in the hostname (put *EV5*), the domain name and domain name server address. Then, exit the program and test the network, for example, with the `ping` command.

3.5.3 Checking Network Name Resolution

During the configuration of the OS/2 Warp machine, the platform will become part of the cell *itsc.austin.ibm.com* with master security and CDS servers on host *ev1*. Assuming that we are going to configure *EV5* now, we must have a correct forward and reverse name resolution between *EV5* and *ev1*.

Forward Resolution: Use the `hostname` and `host` commands to check out the forward resolution of symbolic names to all involved systems:

```
[<ev5>-C:]hostname
ev5.itsc.austin.ibm.com

[<ev5>-C:\]host ev1
ev1.itsc.austin.ibm.com is 9.3.1.68

[<ev5>-C:\]host ev3
ev3.itsc.austin.ibm.com is 9.3.1.122
```

Reverse Resolution: Using an IP address as apposed to a symbolic name allows us to check the reverse resolution.

```
[<ev5>-C:]host 9.3.1.68
9.3.1.68 is ev1.itsc.austin.ibm.com

[<ev5>-C:\]host 9.3.1.122
9.3.1.122 is ev3.itsc.austin.ibm.com

[<ev5>-C:\]host 9.3.1.124
9.3.1.124 is ev5.itsc.austin.ibm.com
```

3.5.4 Checking Network Routing

Before configuring host *EV5*, it is also required to verify the connectivity with the master hosts of the cell and the correct setting of IP routes.

Checking Connectivity to ev1: This can be done in a very simple way with the `ping` command.

```
.[<ev6>-C:]ping ev1
PING ev1.itsc.austin.ibm.com: 56 data bytes
64 bytes from 9.3.1.68: icmp_seq=0. time=0. ms
64 bytes from 9.3.1.68: icmp_seq=1. time=0. ms
64 bytes from 9.3.1.68: icmp_seq=2. time=0. ms
64 bytes from 9.3.1.68: icmp_seq=3. time=0. ms
```

We can also use the netstat command to verify *routes* and *interfaces*. We must make sure we have at least a default route set.

```
[<ev6>-C:]netstat -r

    destination         router    refcnt        use flags     snmp  intrf
                                                              metric
        default        9.3.1.74       0          0    U       -1   lan0
        9.3.1.0       9.3.1.125       1        627    U       -1   lan0
```

3.5.5 Checking the Network Interfaces

Using the -a (address) option of the netstat command allows you to inspect the interfaces.

```
[<ev6>-C:]netstat -a
addr          9.3.1.124 interface 0 mask ffffff00 broadcast          9.3.1.255
```

OS/2 Warp platform *EV5* has one interface with address 9.3.1.124.

3.5.6 Synchronizing the System Clocks

Time in DCE is based on the Coordinated Universal Time (UTC). For the local time representation, a Time Difference Factor (TDF) is applied. The latter is determined by the TZ environment variable. Since we are in the Central Time zone of the U.S., we set this variable in the CONFIG.SYS file as follows:

```
SET TZ=cst6edt
```

Reboot the system for the correct environment to be in effect.

The clock of the OS/2 Warp system has to be verified before starting the configuration process. This can happen with the time command:

```
[<ev5>-C:]time
Current time is: 17:35:42.41
Enter the new time:                    < Enter NEW TIME
```

3.6 Configuring DCE Clients on OS/2 Warp

We intend to use Hosts *EV5* and *EV6* as DCE clients. After a correct installation of the DCE software and going through all the preparation steps as outlined in 3.5, “Installing and Preparing for DCE Configuration on OS/2 Warp” on page 53, we are ready to configure DCE on the OS/2 Warp platforms. After discussing the use of the GUI, we will show how the same configuration can be achieved with the command line interface (CLI).

3.6.1 DCE Client Configuration Using the GUI

During the installation of the DCE Software, a DCE icon has been placed on the desktop. Opening this icon shows a submenu, which among other options presents a *Configure DCE* icon, as shown in Figure 13.

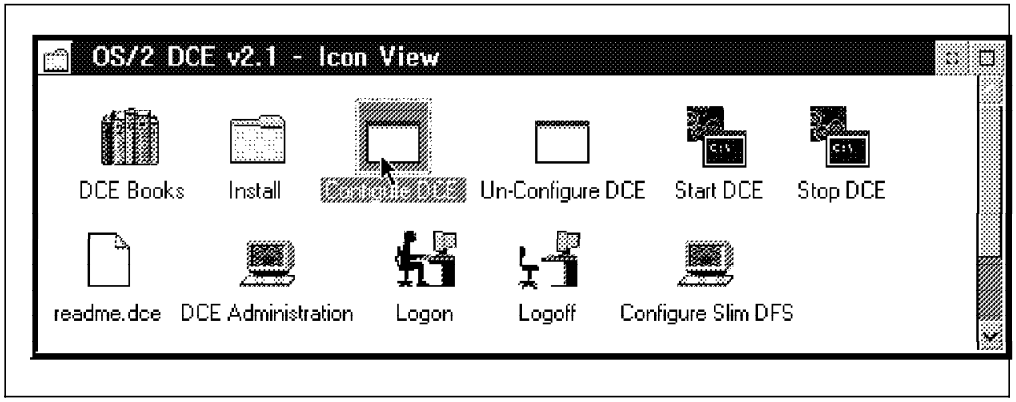


Figure 13. DCE Menu Window

Double-clicking on this icon activates the OS/2 DCE configuration. A similar result can be obtained by starting the program `optdcelocalbincfgdceg.exe`.

The first screen to be presented is the IBM logo, as shown in Figure 14.

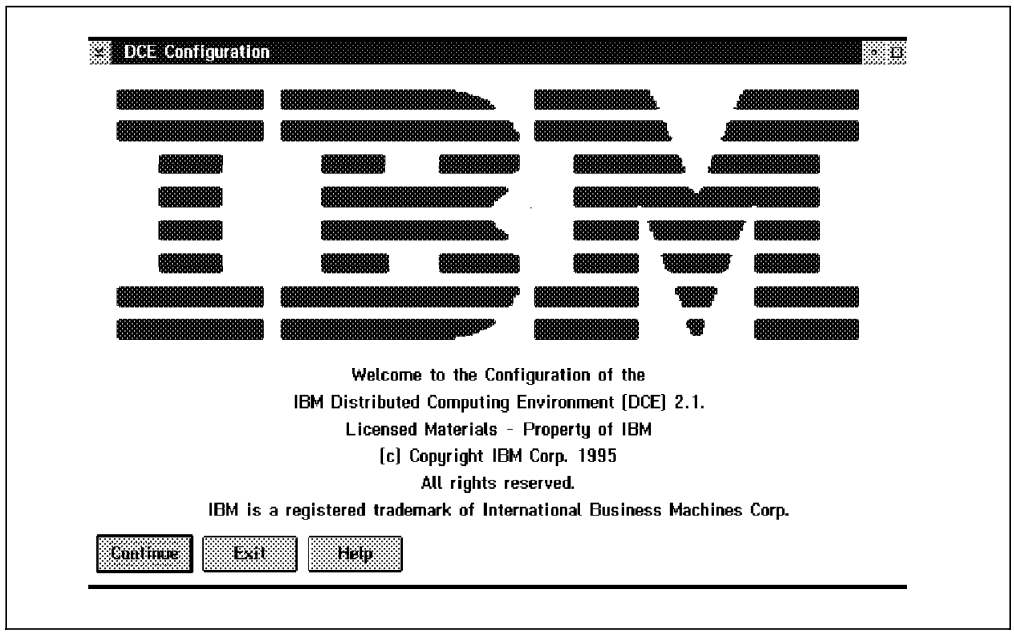


Figure 14. Configuration Logo Window

On this screen, we click on the **Continue** button, and the system starts to query its current configuration. After a short time, the following selection panel is shown.

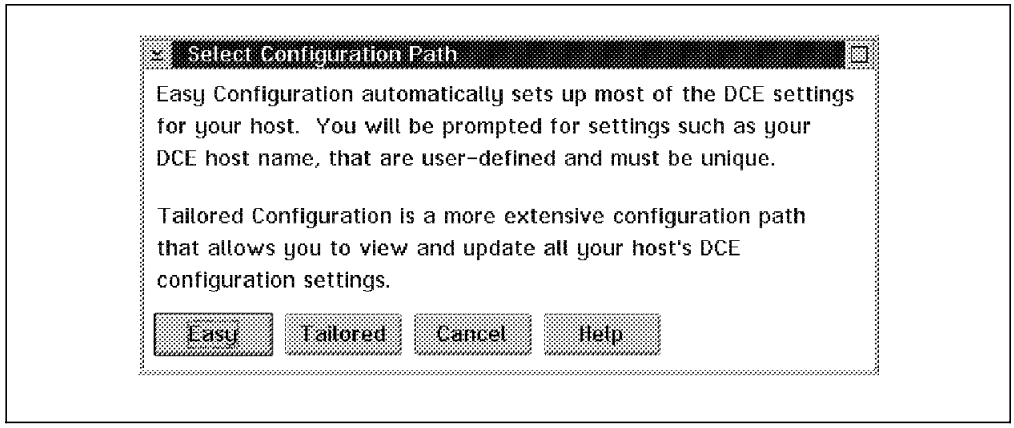


Figure 15. Select Configuration Path Window

We have the possibility to select different configuration paths. The tailored path gives you a notebook-like interface with a set of pages dedicated to each DCE component. In general, the *Easy* path will do it, and that's the one we select in Figure 15 above.

This will bring up the *Specify Configuration Response File Names* window as shown in Figure 16 on page 58.

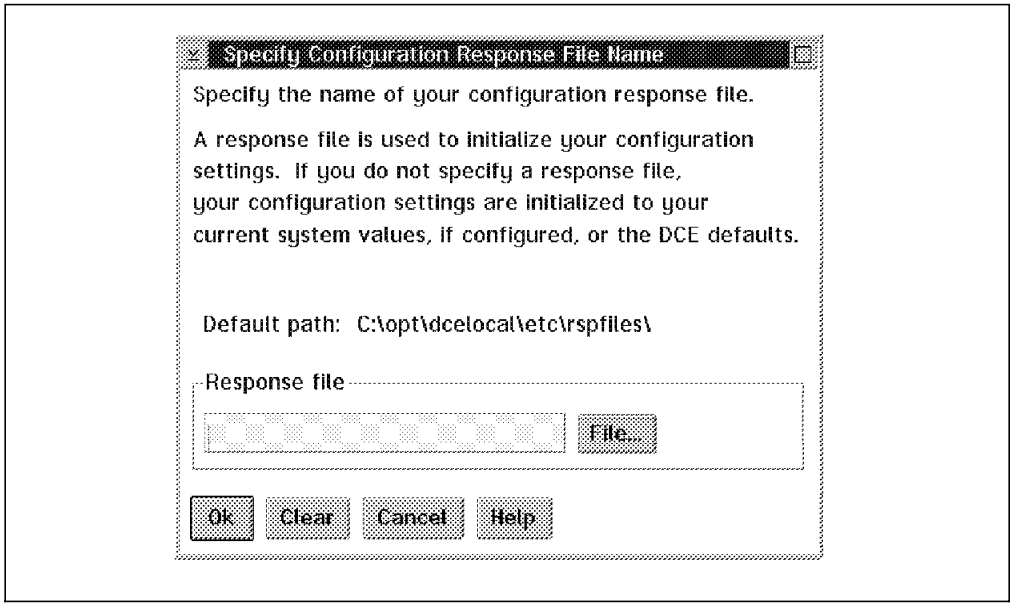


Figure 16. Specify Configuration Response File Name Window

There is the opportunity to save a response file so that you can store different configurations and revert back to them when needed. We did not use a response file and therefore left the input area empty. Then click on the **OK** button, which will bring up the *Specify Setup Parameters* window shown in Figure 17 below.

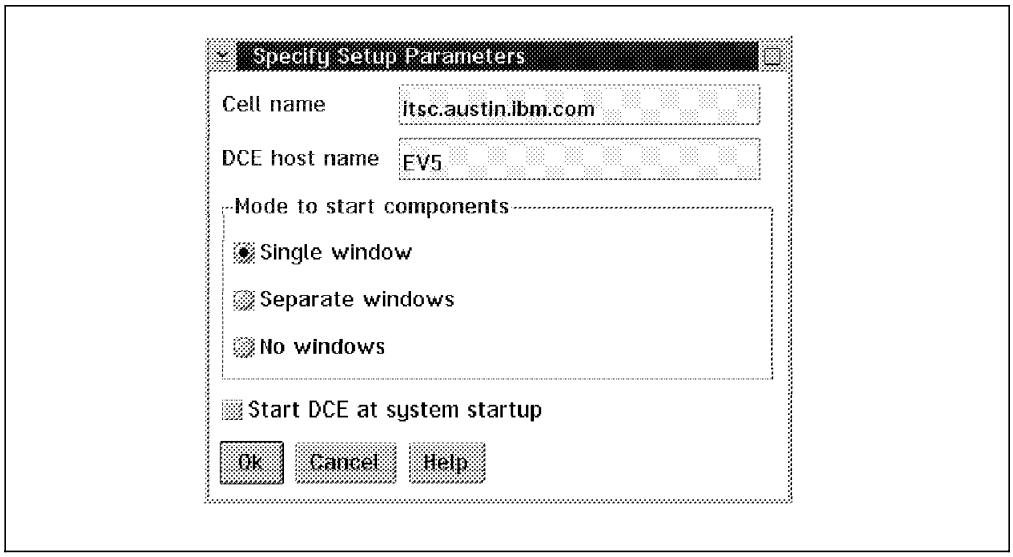


Figure 17. Specify Setup Parameters Window

Enter the cell name and the DCE host name. In general, take the TCP/IP name for the DCE hostname. The other options indicate *when* and *how*, the configured components will be started. Then click on the **OK** button.

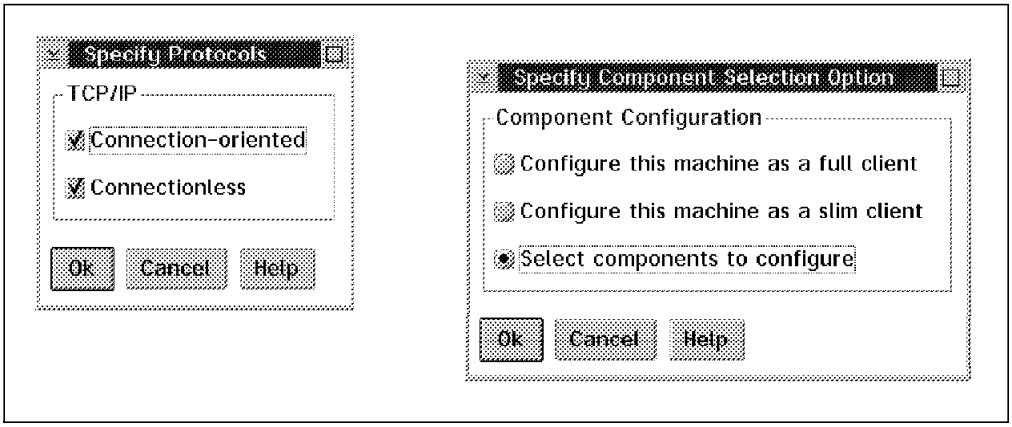


Figure 18. Specify Protocols and Component Selection Option Window

In the upcoming *Specify Protocols* window we select both TCP/IP protocols, and in the *Specify Component Selection Option* window (Figure 18), we will select **Select components to configure**. This option gives us full control of all steps. Then click on the **OK** button to proceed to the *Select Security Components* window shown in Figure 19 below.

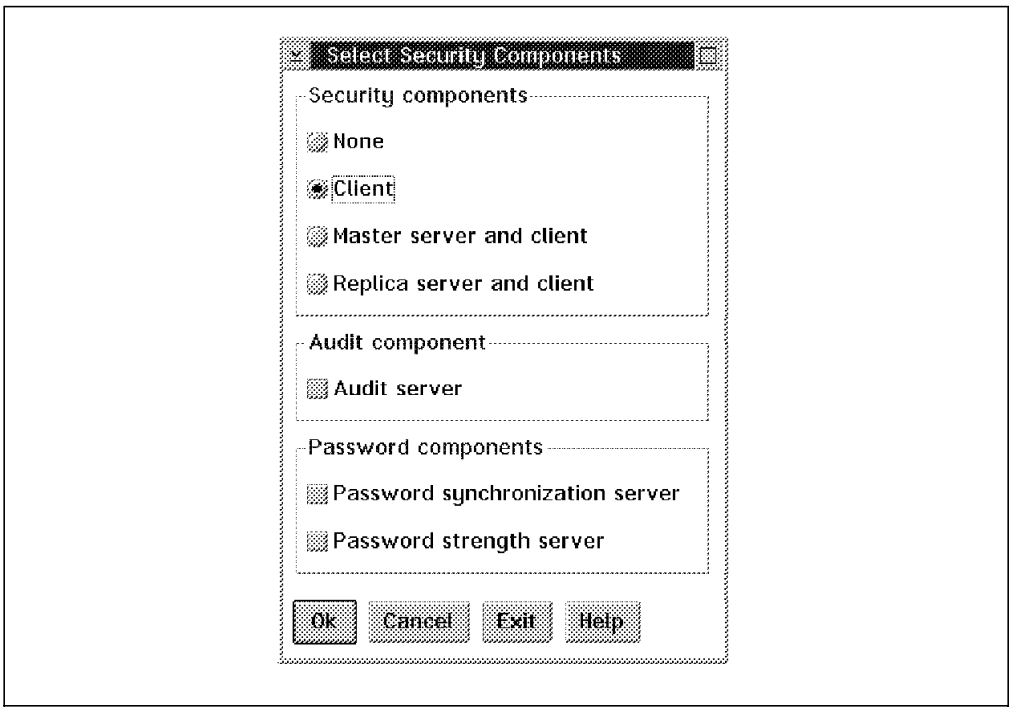


Figure 19. Select Security Components Window

Select the DCE security **Client,** and click on the **OK** button.

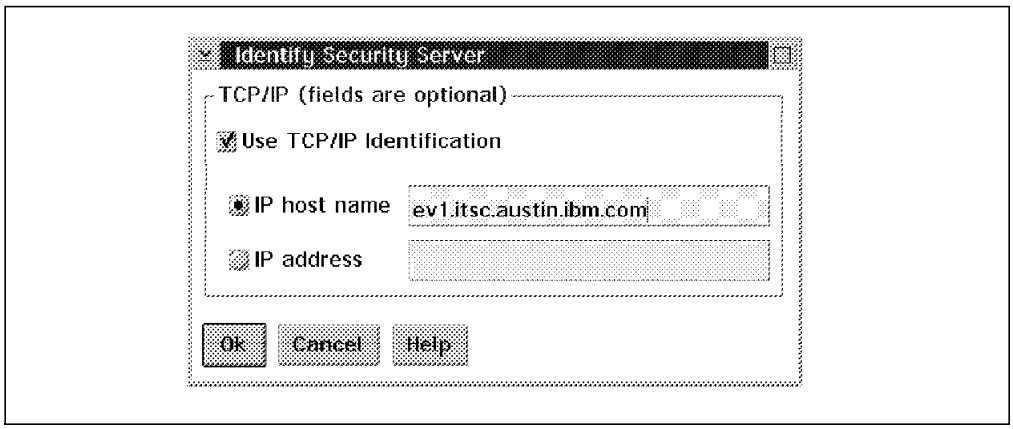


Figure 20. Identify Security Server Window

Identify the security server via hostname or IP address. For this, *ev1* would be sufficient if the network name resolution is set up correctly by using either the *Domain Name Service* or the *hosts* file. We have used the fully qualified host name in the *Identify Security Server* window in Figure 20.

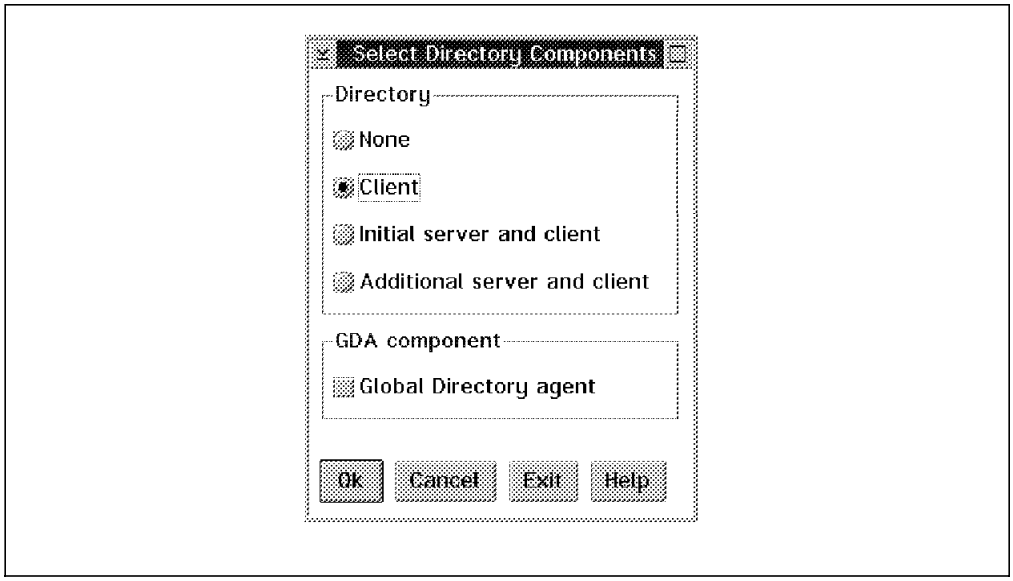


Figure 21. Select Directory Components Window

Next, select the CDS **Client** component in the upcoming *Select Directory Components* window in Figure 21. Currently we are not interested in the *Global Directory Agent* and therefore have not selected it.

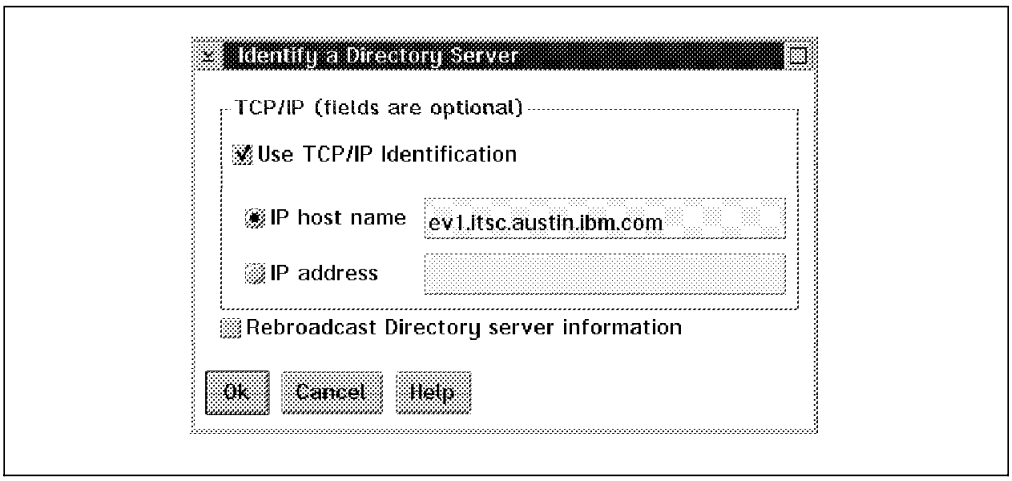


Figure 22. Identify Directory Server Window

In Figure 22 we need to specify the CDS server configured previously. We use the option to fill in the IP host name of *ev1*.

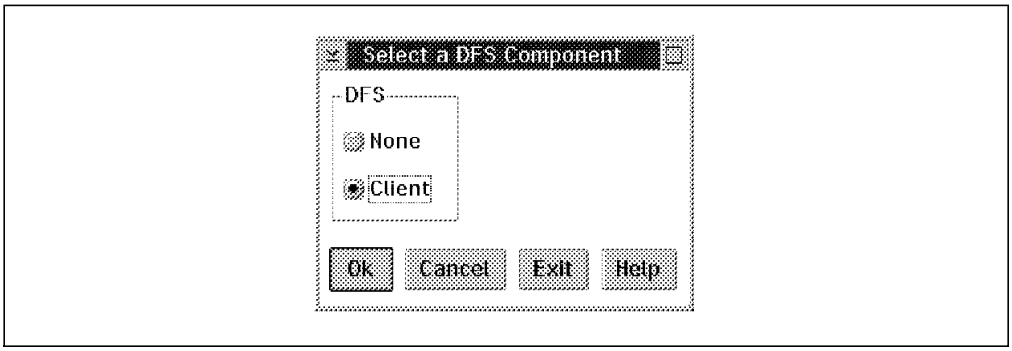


Figure 23. Select DFS Components Window

Although the DFS server has not yet been configured, for the ease of the preparation, we already request a DFS client. This is done in the *Select DFS Components* Window as shown in Figure 23. It will configure a *dfsd* daemon and will add a driver into the *config.sys* file. After start-up, during system boot, the driver will act as an extension of the Warp kernel.

The *Select an Event Management* window shown in Figure 24 allows us to specify whether or not this system will forward events to an Simple Network Management Protocol (SNMP) manager.

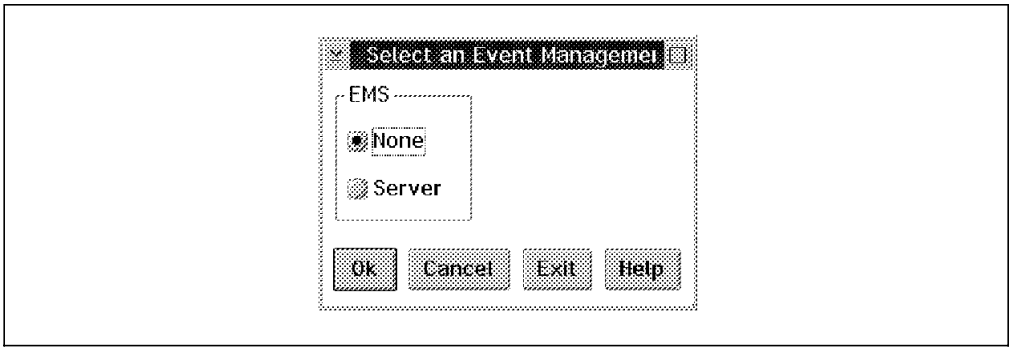


Figure 24. Select an Event Management Window

We chose not to use event management for this configuration. After proceeding with the **OK** button, we get to the DTS configuration.

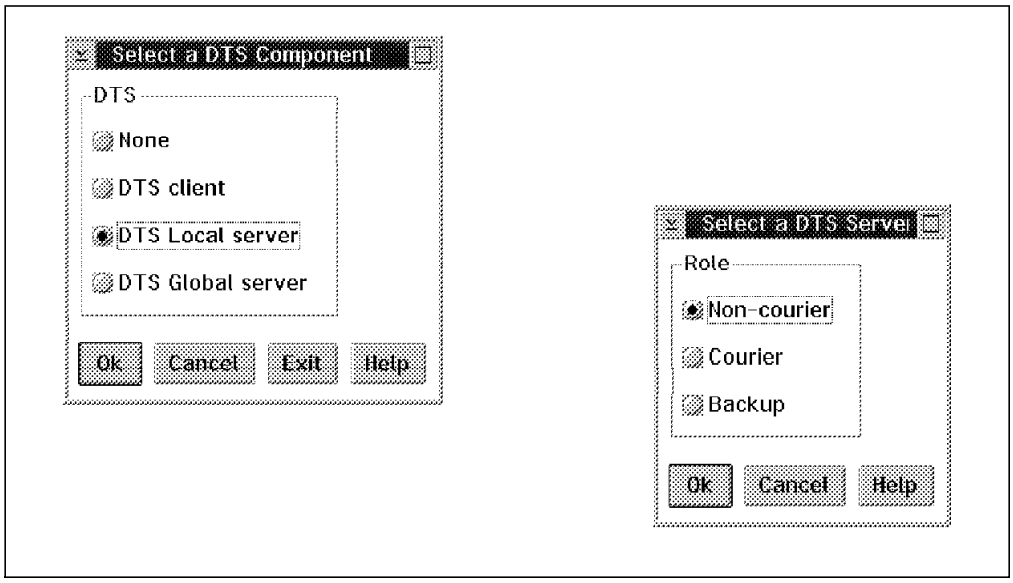


Figure 25. Select DTS Component and Server Type Window

We have the possibility to configure a DTS client or server. We selected a **DTS Local server** (in the upper-left window of Figure 25), with a **Non-courier** role (in the lower-right window). A *local DTS non-courier* server only has a responsibility in the LAN.

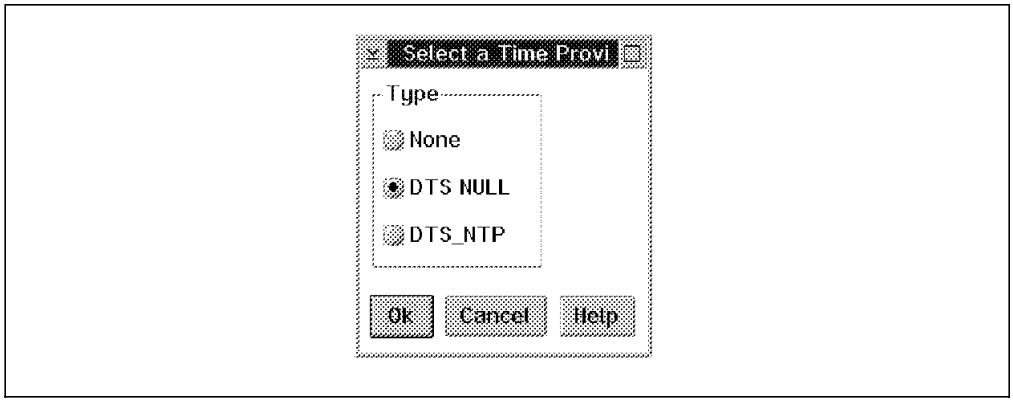


Figure 26. Select a Time Provider Window

To use the OS/2 internal clock, we selected the **DTS NULL** time provider option in Figure 26.

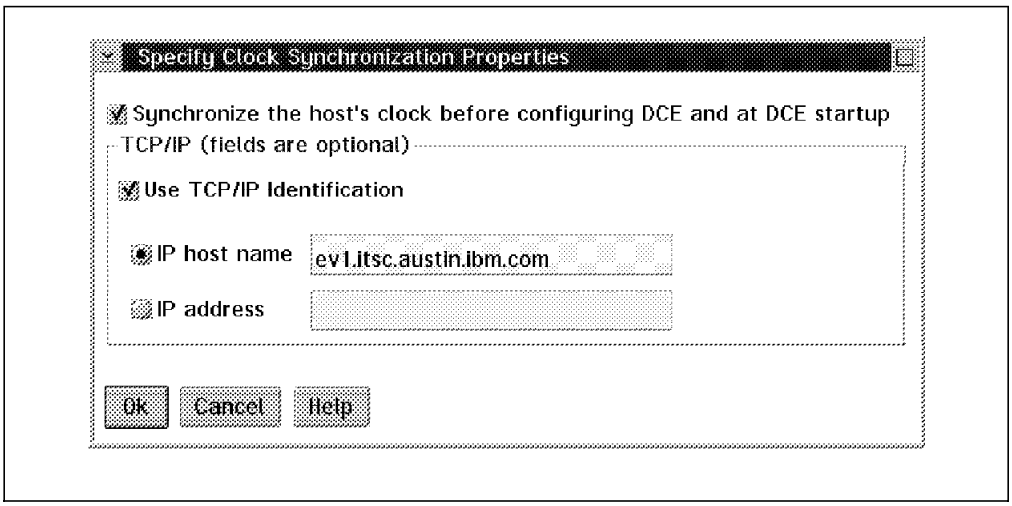


Figure 27. Specify Clock Synchronization Window

As mentioned in 3.5.6, “Synchronizing the System Clocks” on page 56, synchronization of the clocks is very important in a DCE cell. A time drift of more than five minutes will prevent the platform from configuring. You can import the time of your security server with the panel shown in Figure 27.

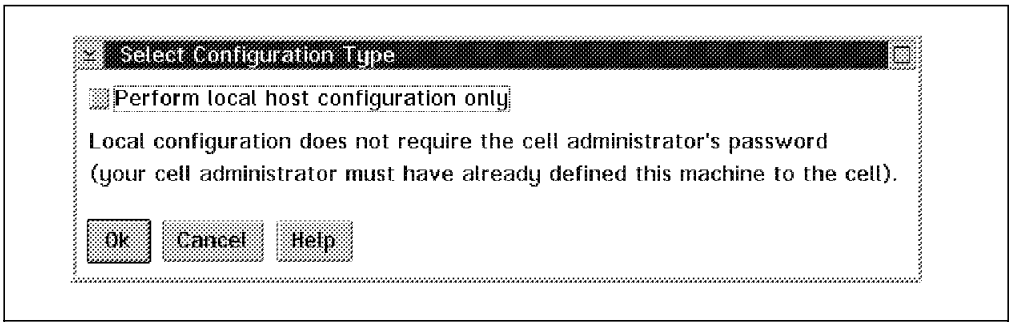


Figure 28. Select Configuration Type Window

Here we have the opportunity to specify a *full* configuration or a *local* configuration. The local requires an anticipated administrative configuration by cell_admin. We go with the full configuration because *EV5* has not been preconfigured into DCE.

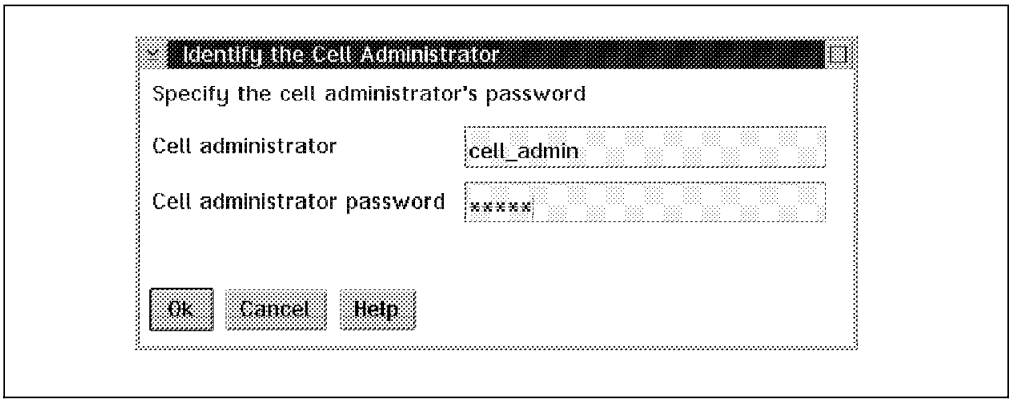


Figure 29. Identify the Cell Administrator Window

Because we selected a *full* configuration, we are required to provide the cell administrator’s name and password, as shown in Figure 29 above. Now we are

ready to execute the configuration with the parameters we have set. Clicking the **OK** button brings up the *Run Configuration* window.

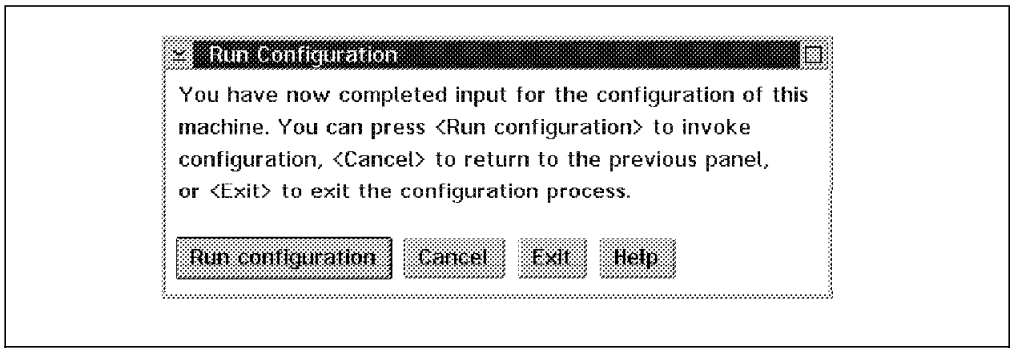


Figure 30. Run Configuration Window

Clicking on **Run configuration,** shown in Figure 30, will start the configuration process.

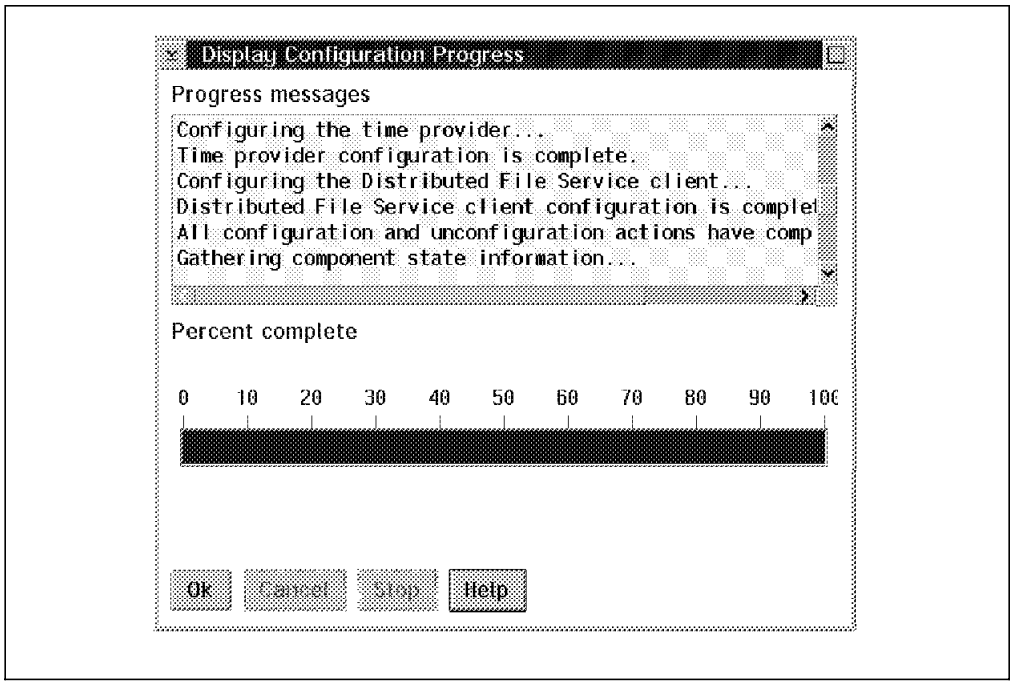


Figure 31. Display Configuration Progress Window

Finally, the configuration started and after a few minutes, the task result was presented as completed successfully. This is shown in Figure 31.

3.6.2 DCE Client Configuration Using the Command Line Interface

In order to achieve the same DCE and DFS client configuration as with the GUI above, issue the following command:

```
[C:] cfgdce cds_c sec_c dts_l dfs_c -o full -h EV5 -c tcpip -n itsc.austin.
ibm.com -s h.ev1 -t h.ev1 -S courier -T DTSnull
Gathering current configuration information...
0x1131505A: The cell administrator password is required for configuration.
0x1131505C:
Type the cell administrator password:
```

```
Configuring DCE...
Verifying data for the configuration or unconfiguration request.
All data needed for the request has been verified.
Configuration of DCE Host, EV5, will now begin.
Configuring the Security client...
Security client configuration is complete.
Configuring the Directory client...
Directory client configuration is complete.
Configuring the Distributed Time Service...
Distributed Time Service configuration is complete.
Configuring the Distributed File Service client...
Distributed File Service client configuration is complete.
Configuring the time provider...
Time provider configuration is complete.
Gathering component state information...

                DCE Component Summary for Host: EV5
      Component                    Configuration State         Running State
Security client                        Configured                  Running
Directory client                       Configured                  Running
DTS Local server                       Configured                  Running
DFS client                             Configured                  Running

The DCE component summary is complete.
Configuration has completed.
```

The cfgdce -? command displays help information.

3.7 Configuring Core Server Replica on AIX

In this part of the chapter we will complete our cell design by installing replicas. The reason for installing replicas can be manifold, especially in cells, which encompass WAN connections. However, in all cells it is also a matter of reliability and safety.

DCE core services that can be replicated are the CDS server, the security server and the DTS server. For the DTS server, the term *replication* is not appropriate; we would rather use the term *redundancy*.

Replication increases availability and load-balancing, but you should know that DCE replication has its limitations; replicated resources are *read-only*. Write access might be unavailable at times when a primary server fails.

In this section we discuss:

- Configuring a CDS replication server
- Configuring a security replication server

3.7.1 Replicating a CDS Server

The following are the steps to follow on the machine where you want to add another CDS server. Our example uses *ev3*. The DCE version installed on *ev3* is DCE 1.3 on AIX 3.2.5. We start the additional configuration with SMIT:

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Servers
         -> CDS (Cell Directory Service) Server
            -> 2 additional
               (fastpath = mkcdssrv)
```

```
                        CDS (Cell Directory Service) Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* CELL name                                           [/.../itsc.austin.ibm.com>
* SECURITY Server                                     [ev1]
  Initial CDS Server (If in a separate network)      []
* Cell ADMINISTRATOR's account                       [cell_admin]
* LAN PROFILE                                        [/.:/lan-profile]
  Machine's DCE HOSTNAME                             [ev3]
```

Selecting **Enter=Do** starts the configuration of the additional CDS server with the following progress list:

```
Enter password for DCE account cell_admin:

Password must be changed!
Configuring Additional CDS Server (cds_second)...
           Modifying acls on ev3_ch
Additional CDS Server (cds_second) configured successfully

Current state of DCE configuration:
cds_cl       COMPLETE   CDS Clerk
cds_second   COMPLETE   Additional CDS Server
dts_cl       COMPLETE   DTS Clerk
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
          Press Enter to continue
```

The previous display shows that we now have an additional CDS server on this machine (*ev3*). The component is called *cds_second*.

Since *ev3* already was a DCE client, the following command would have had the same effect:

```
# mkdce cds_second
```

At this point, even if you have configured the machine to become a CDS replication server, you have not achieved any replication of CDS data, with the exception of the cell root (/.:) directory and its content (the *leaf objects*).

You have to manually replicate (sub)directories. The word *replica* has a special meaning with respect to the CDS; the first instance of a (sub)directory is already called a *replica,* and each further copy of this directory object is another replica. The read/write copy is called the master replica. The replication of the CDS is distributed, which means that the master replicas of different directories may be in several distinct *clearinghouses*. This opens a way for complex setups.

We decided to keep this replication straightforward and to replicate the *entire* CDS master structure into the new clearinghouse on *ev3*. A shell script has been written to fulfill this task. This script carries the name copy_ch.

```
ev1::/-> copy_ch -h
Usage: copy_ch [-m] -s <source CH> -t <target CH> [-h]

Version 2.1

[-m]                          Will define master replicas on the target
-s   <Source clearinghouse>   Can contain master and/or read-only replicas
-t   <Target clearinghouse>   Will contain R/O replicas unless -m is given
[-h]                          help

Example:

   copy_ch -m -s ev7_ch -t new_ch

   This example creates all replicas of /.:/ev7_ch in /.:/new_ch, no
   matter whether they are master or read-only replicas.  When you use
   the -m flag, every master replica found in /.:/ev7_ch is established
   as a master replica in the target clearinghouse (/.:/new_ch), thereby
   relocating the master role.

   There may be more clearinghouses defined in the cell. This command
   correctly redefines the replica set of every directory of which
   /.:/ev7_ch has an instance.  The target clearinghouse does not need
   to be empty; therefore the command can also be used to merge the
   source clearinghouse into the target clearinghouse.
```

The copy_ch command basically performs three steps, using the dcecp command now offered by DCE 2.1. It uses Korn shell and some UNIX tools; therefore it only runs on AIX DCE 2.1 machines. However, it can copy or move clearinghouses between any platform. It could be developed completely in the Tcl language, which would provide the advantage of enabling *portable* code for all platforms. The three steps are:

1. The script first collects the names of all (sub)directories in the original clearinghouse and puts them in a table.
2. For each existing (sub)directory, a create replica is done into the new clearinghouse.
3. For each (sub)directory (or replica), the replica set is redefined, no matter how many clearinghouses there are. This step is achieved with the cdscp set directory to new epoch master command, which also implicitly invokes a skulking (update of the read-onlys).

This script is on the diskette accompanying this redbook. It is well documented, and details can be seen in the script file.

3.7.2 Replicating the Security Server

Security servers have to be replicated on a system with the same or higher DCE software level. This is something that we found out by trying first to replicate on the *ev3* system with AIX 3.2.5 and DCE1.3. By using the dcecp command, we can find out that the currently active DCE registry on *ev1* is on level 1.1. A replication on a DCE 1.3 would imply a downgrade of the registry, and this is not possible.

```
dcecp> registry show
{deftktlife +0-10:00:00.000I-----}
{hidepwd yes}
{maxuid 32767}
{mingid 100}
{minorgid 100}
{mintktlife +0-00:05:00.000I-----}
{minuid 100}
{version secd.dce.1.1}
```

Replication of the security server is really making a new copy of the existing security databases. We will build this new security server on the *ev4* system with AIX4.1.4 and DCE 2.1.

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Servers
         -> SECURITY Server
            -> 2 secondary
               (fastpath = mkdcesecsrv)
```

```
                                SECURITY Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* CELL name                                          [/.../itsc.austin.ibm.com>
* Cell ADMINISTRATOR's account                       [cell_admin]
* REPLICA name                                       [ev4]
* SECURITY Server                                    [ev1]
  CDS Server (If in a separate network)              []
* LAN PROFILE                                        [/.:/lan-profile]
  Machine's DCE HOSTNAME                             [ev4]
```

Selecting **Enter=Do** starts the configuration of the additional security server with the following progress list:

```
Enter password for DCE account cell_admin:

Password must be changed!

Configuring Security Server (sec_srv)...

          Modifying acls on /.:/sec/replist
          Modifying acls on /.:/subsys/dce/sec
          Modifying acls on /.:/sec
          Modifying acls on /.:
          Modifying acls on /.:/cell-profile
Security Server (sec_srv) configured successfully

Current state of DCE configuration:
cds_cl       COMPLETE   CDS Clerk
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
sec_srv      COMPLETE   Security Server (Replica)
         Press Enter to continue
```

This following command has the same effect:

```
# mkdce -R -r ev4 -n itsc.austin.ibm.com -s ev1 sec_srv
```

The security server on this machine is marked (Replica). If the security master server goes down, you can continue to log in to the cell, but it may take a long time. Access to the security servers is randomly selected, and some calls are directed toward the unavailable server because its bindings are still exported or cached in CDS. In this case, a client will experience a time-out before it looks for an alternative server to get tickets from.

3.8 Configuring Server Replica on OS/2 Warp

CDS and registry (security) replica can also be put on an OS/2 Warp platform. This is what we describe in this section.

3.8.1 Replicating the CDS Server on OS/2 WARP

We will configure *EV5* for a CDS secondary. Remember that configuring a *cds-second* does not duplicate any data. This has to be done later on.

To describe this configuration, we will restart from the *Select Configuration Path* window as shown Figure 15 on page 57 and select the **Tailored** path. This brings us up to the following PM screen.

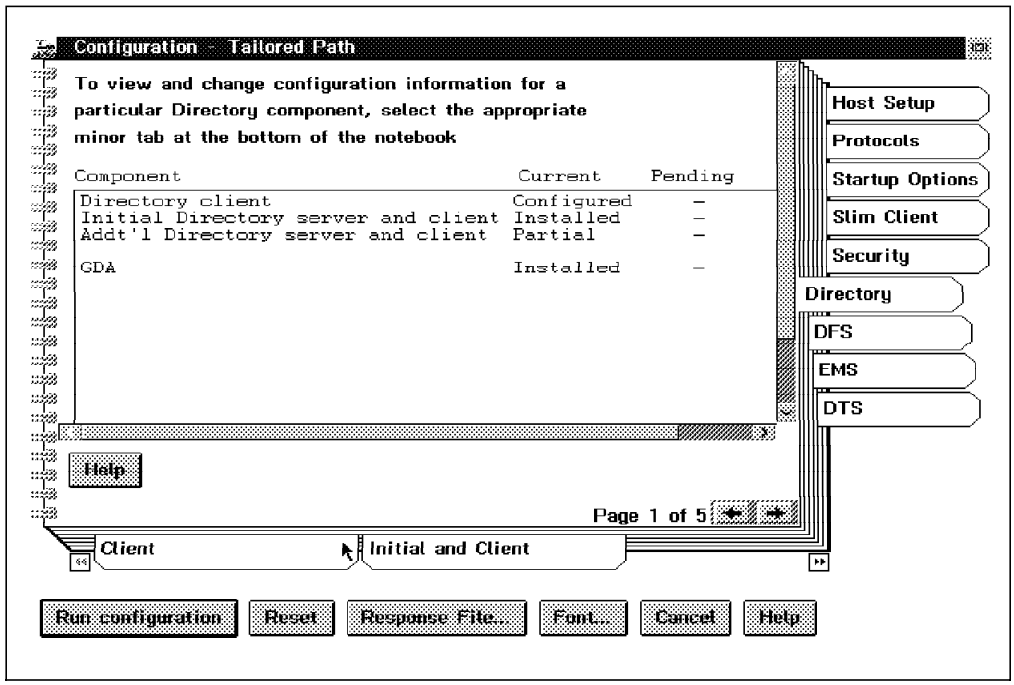


Figure 32. Tailored Path Window

The tailored path presents us a notebook look, as shown in Figure 32. The major tabs on the right allow for scrolling through the component groups. Within a group, we can scroll through the detail panels using the tabs at the bottom.

The window is already positioned on the CDS component; it shows the current status of CDS functions on this platform. With the minor tabs, we page to the *Additional and Client CDS* page in the notebook.

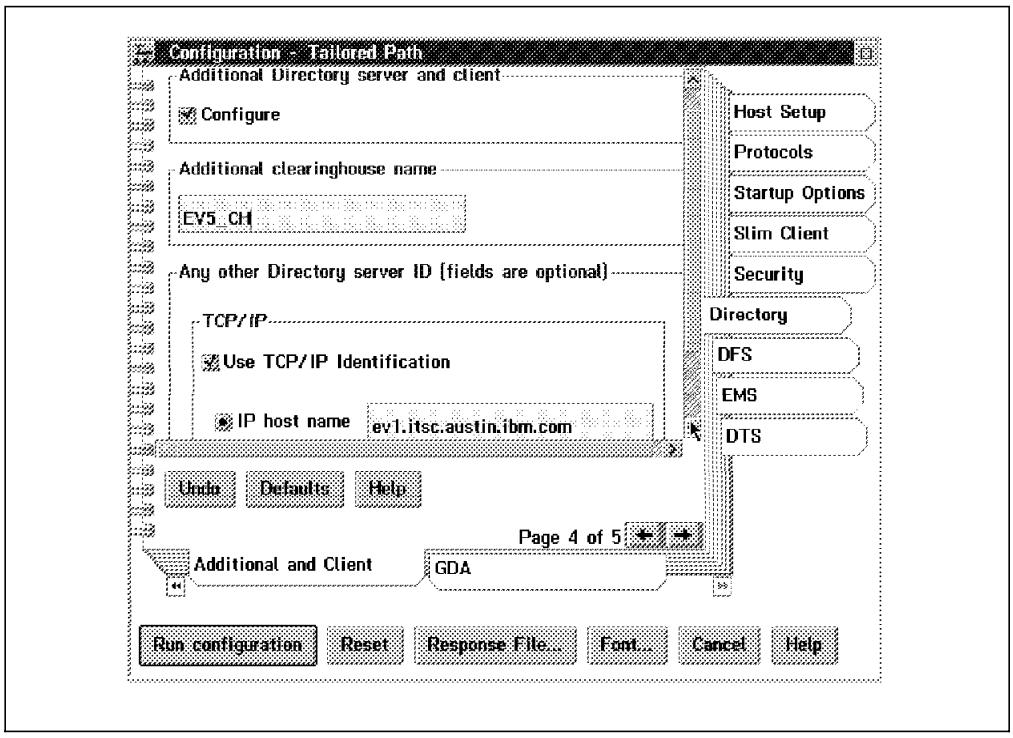


Figure 33. Configuration - Tailored Path Window

We fill in the required data in Figure 32 on page 69, and the **Run configuration** button will start the work. Upon successful completion, the additional CDS will be configured.

3.9 Summary

All DCE core services in our cell have now been configured, and we are ready to use this environment for applications. It is easy to find out which components have been configured on each platform

3.9.1 Inventory on the AIX Platform

On an AIX platform, type the following command:

```
ev1::/-> lsdce
Current state of DCE configuration:
cds_cl       COMPLETE   CDS Clerk
cds_srv      COMPLETE   Initial CDS Server
dts_local    COMPLETE   Local DTS Server
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
sec_srv      COMPLETE   Security Server (Master)
```

3.9.2 Inventory on the OS/2 Warp Platform

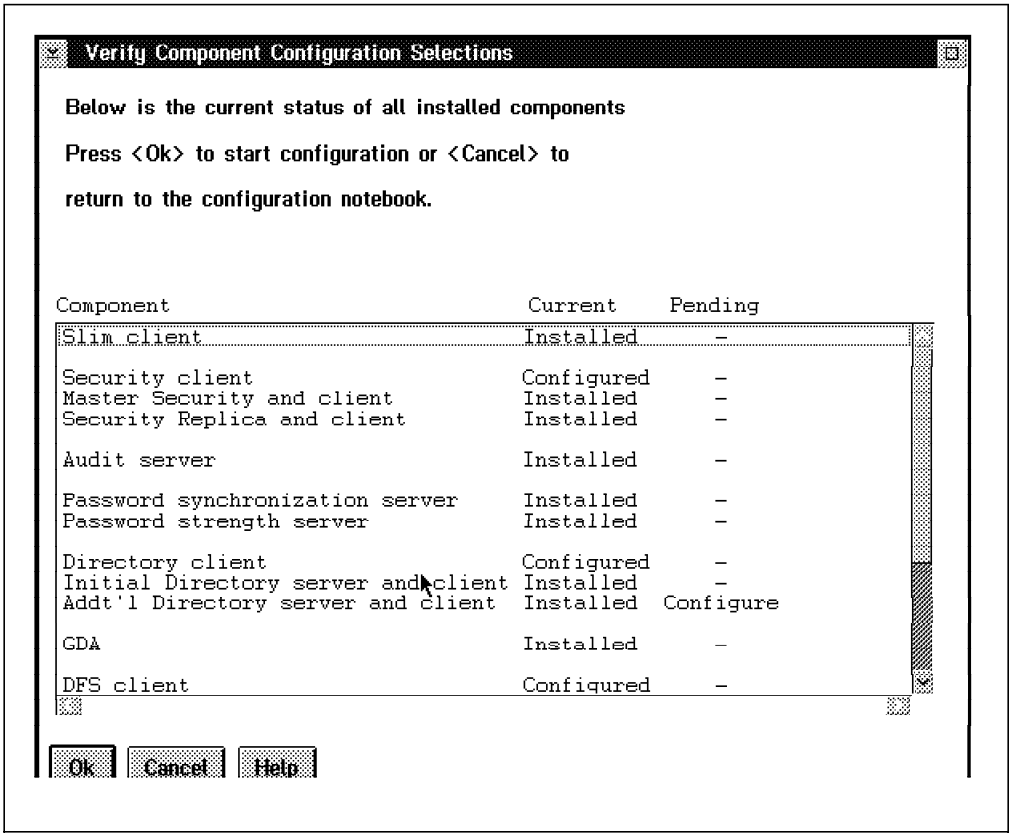


Figure 34. Verify Component Configuration Selections Window

Within the tailored *Configuration Path,* it is possible to get an overview of the configuration status of the OS/2 Warp machine. The panel shown in Figure 34 is presented just before beginning the configuration execution. It clearly shows all *Installed*, *Configured* and *Pending* components. This can also obtained component by component. The showcfg.exe command presents the same information.

Another way to find out the configuration status of an OS/2 Warp platform is to use the following server cat command:

```
dcecp> server cat /.:/hosts/EV5
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/time_provider
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/cdsadv
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/dfsd
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/dtsd
```

From this, we find out what components have been configured on *EV5*.

3.9.3 Cell Configuration and Status Information

The dcecp shell in general will be extremely valuable to verify and interrogate the cell status. For instance, look at the following commands:

- Displaying the cell status information:

```
dcecp> cell show
{secservers
 /.../itsc.austin.ibm.com/subsys/dce/sec/master
 /.../itsc.austin.ibm.com/subsys/dce/sec/ev4}
{cdsservers
 }
{dtsservers
 /.../itsc.austin.ibm.com/hosts/ev1/dts-entity
 /.../itsc.austin.ibm.com/hosts/EV5/dts-entity
 /.../itsc.austin.ibm.com/hosts/EV6/dts-entity}
{hosts
 /.../itsc.austin.ibm.com/hosts/EV5
 /.../itsc.austin.ibm.com/hosts/EV6
 /.../itsc.austin.ibm.com/hosts/ev1
 /.../itsc.austin.ibm.com/hosts/ev2
 /.../itsc.austin.ibm.com/hosts/ev3
 /.../itsc.austin.ibm.com/hosts/ev4}
```

- Listing all hosts that are part of the cell:

```
dcecp> host cat
/.../itsc.austin.ibm.com/hosts/EV5
/.../itsc.austin.ibm.com/hosts/EV6
/.../itsc.austin.ibm.com/hosts/ev1
/.../itsc.austin.ibm.com/hosts/ev2
/.../itsc.austin.ibm.com/hosts/ev3
/.../itsc.austin.ibm.com/hosts/ev4
```

- Listing the status of the CDS replication:

  We also developed a TCL script that lists the complete status of the CDS replication. It works on AIX and OS/2:

```
[C:] dcecp show_cds
==> List of Clearinghouses
/.../itsc.austin.ibm.com/ev1_ch
/.../itsc.austin.ibm.com/ev3_ch
   number of clearinghouses = 2
==> Analyze clearinghouse /.../itsc.austin.ibm.com/ev1_ch
==> Analyze clearinghouse /.../itsc.austin.ibm.com/ev3_ch
 scan of ClearingHouses terminated total replicas 24
 cell  /.../itsc.austin.ibm.com cellname_length = 25
 directory /.:/
ev1_ch { Master}
ev3_ch { ReadOnly}
 directory /.:/hosts
ev1_ch { Master}
ev3_ch { ReadOnly}
 directory /.:/hosts/EV5
ev1_ch { Master}
ev3_ch { ReadOnly}
 directory /.:/hosts/EV6
ev1_ch { Master}
ev3_ch { ReadOnly}
 directory /.:/hosts/ev1
ev1_ch { Master}
ev3_ch { ReadOnly}
 directory /.:/hosts/ev2
ev1_ch { Master}
ev3_ch { ReadOnly}
```

```
 directory /.:/hosts/ev3
ev1_ch { Master}
ev3_ch { ReadOnly}
 directory /.:/hosts/ev4
ev1_ch { Master}
ev3_ch { ReadOnly}
 directory /.:/subsys
ev1_ch { Master}
ev3_ch { ReadOnly}
 directory /.:/subsys/dce
ev1_ch { Master}
ev3_ch { ReadOnly}
 directory /.:/subsys/dce/dfs
ev1_ch { Master}
ev3_ch { ReadOnly}
 directory /.:/subsys/dce/sec
ev1_ch { Master}
ev3_ch { ReadOnly}
 number of Clearinghouse directories 12
```

3.9.4 Summary of Daemons and Processes

Let's give a brief overview of the required components on each server. The following table shows what daemons and processes are required in order to enable a particular function.

Machine Role	Function Performed	Processes	Remarks
All machines	EndPoint mapper sec_cl (in DCEV2.1) Host Management	dced (sec_cl)	Will play an important role in current and future DCE management
All clients, also DTS	CDS advertiser CDS clerk function Time clerk	+ cdsadv + cds_clerk + dtsd -c	CDS clerk is started via cdsadv
Security server	Security daemon	+ secd	Master and Secondary
CDS server	CDS daemon	+ cdsd	Also on all replica servers
Time server	Time distributor	+ dtsd -s	dtsd can have client or server role

Figure 35. List of Processes and Daemons for DCE Machine Roles

Chapter 4. Implementing DFS

The Distributed File System (DFS) is one of the first real DCE applications. It is not considered a core service. See 1.2.7, “Distributed File System” on page 11 in this publication for a short description on DFS. The redbook *The Distributed File System (DFS) for AIX/6000,* GG24-4255, has more details on DFS.

In this chapter we describe the configuration of DFS in an already configured DCE cell composed of several distinct hardware and software platforms. We extend scenario 1 configured in Chapter 3, “Implementing DCE Cells” on page 37, with detailed configuration instructions for DFS. We also discuss experiences with fileset replication, a feature which was not available at the time the redbook titled *The Distributed File System (DFS) for AIX/6000* was created.

In Chapter 5, “Implementing Various LAN/WAN Scenarios” on page 105, we give short-path instructions to quickly install different scenarios. Besides giving step-by-step installation instructions for different network topologies, we also discuss performance and availability issues in that chapter.

4.1 Overview and Cell Layout

The following DFS and AIX levels are used in the configuration:

- DFS 2.1 for AIX 4.1.4 as DFS servers and clients
- DFS 1.3 for AIX 3.2.5 as DFS server and clients
- DFS (BETA 21/01/96) for OS/2 Warp (OS/2 V3) as DFS clients only

Before starting, let’s again have a closer look at the objective that we like to achieve and the role that the machines of the DCE cell will play in the DFS setup. We recall that the *DFS* is a distributed application, with different roles attributed to particular machines, as shown in Figure 36 below.

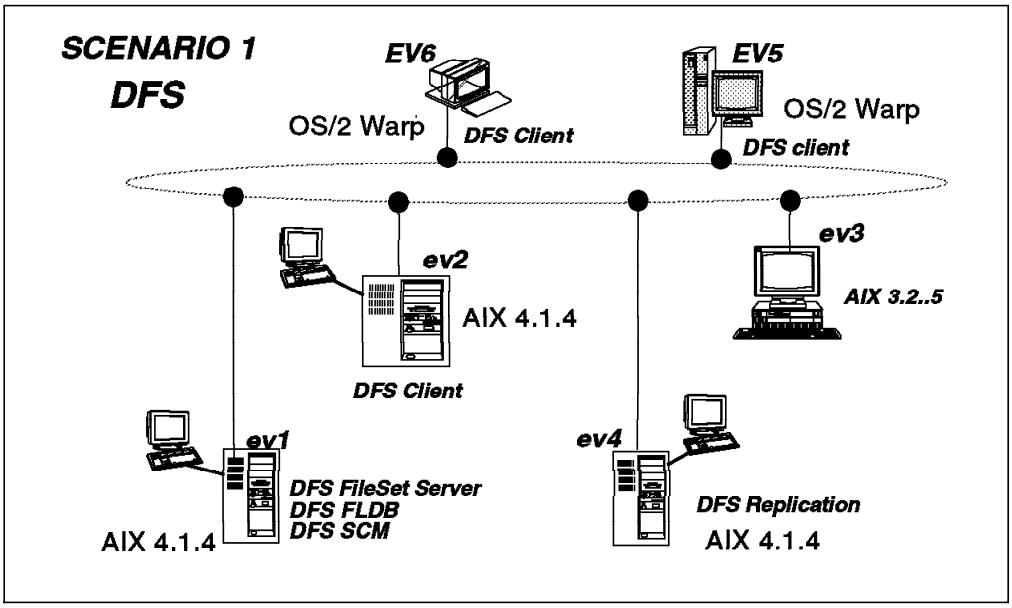


Figure 36. DFS Implementation Within Scenario 1

According to the tasks that have to be executed to achieve our cell objective, we discuss the following topics:

- Configuring DFS server roles on AIX
  - Configuring the DFS SCM role on *ev1*
  - Configuring the DFS FLDB role on *ev1*
  - Configuring a fileset server role on *ev1*
  - Preparing the DFS root fileset
  - Adding an additional Fileset
- Configuring a DFS client on OS/2 Warp
- Replicating DFS filesets under AIX
  - Configuring a replication server on *ev1*
  - Configuring a fileset replication server on *ev4*
  - Replicating the *root.dfs* directory on *ev1*
  - Replicating the fileset warp001 on *ev4*

4.2 Configuring a DFS Server

This section contains the basic steps to configure a base DFS server environment. For more detailed information, see *The Distributed File System (DFS) for AIX/6000* redbook.

Before starting to configure the base DFS server environment, make sure that all DCE core services are configured and the Enhanced DFS product is installed. If not, install it according to 3.3, “Installing the DCE Code on AIX” on page 42 or 3.5, “Installing and Preparing for DCE Configuration on OS/2 Warp” on page 53.

A DFS fileset server actually serving data is only one of the many machines that are part of the DFS server complex. Before we are able to export some files, an environment has to be prepared that is composed of three machine roles:

1. System Control Machine (SCM)
2. Fileset Location Data Base Machine (FLDB)
3. Fileset Server Machine(s)

After setting up these roles, one can start with the building of the real *”file server”* function. This function will be supported by one or several machines, with or without replication. Configuration of the DFS filespace has to start with a *root* directory. The first *file server* to be added to the DFS set has to include this directory.

Below is a short summary description of the different DFS components (roles) to be configured:

- **System Control Machine**

  The *System Control Machine* (SCM) controls various lists of users and groups that can perform administrative functions on the different types of DFS servers. The SCM houses the master copy of these lists that are distributed to the various fileset servers. A DFS domain is the set of machines that is controlled by one SCM and respectively by the same lists of

DFS administrators. There can be multiple domains in a DFS server complex.

- **Fileset Location DataBase Machine**

  A *fileset location database* machine stores the Fileset Location Database (FLDB). The purpose of the FLDB is to take a path name for a file that is located in the DFS namespace and determine the location of the file server that houses that file. The end result is that the user never has to know the physical location of a file. The user is only confronted with an hierarchical look of the total DFS filespace.

- **File Server Machine**

  A *file server* machine is used to store and export DFS LFS file systems or non-LFS file systems into the DFS namespace. In AIX/DCE, a non-LFS file system is a JFS (Journaled File System). AIX/DCE supports JFS and LFS file systems. New in this release (2.1) is the support for CD-ROM file systems. In our example, we will use LFS file systems. A fileset is a subtree of files and directories that is smaller or equal to a logical disk partition.

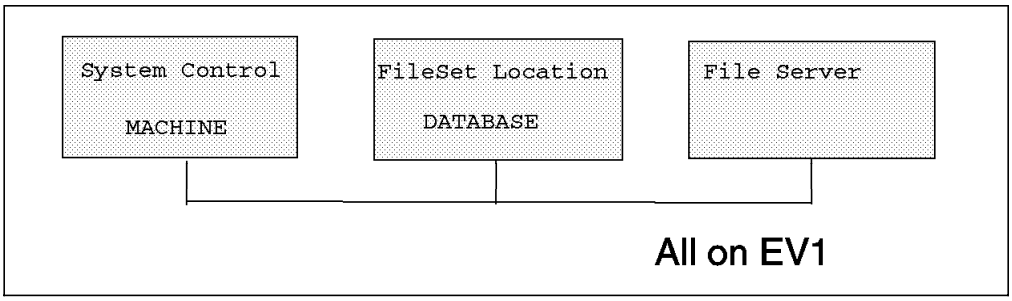


Figure 37. Minimum Basic DFS Machine Roles

These machine roles can be configured on different platforms or on the same platform. As shown in Figure 37, we will configure all these servers on *ev1*. After having configured these roles, we can export the *root.dfs* fileset and other filesets. Below are the steps to create such a DFS filespace.

If you plan to replicate a fileset, you can also replicate this fileset immediately after having created the read/write fileset, without creating mount points first. This would prevent any DFS clients from obtaining access to the read/write fileset via a regular mount point. See 4.4, “Replicating Filesets on AIX” on page 89 for a discussion on when to replicate filesets and when to create mount points.

4.2.1 Configuring a System Control Machine

We first start configuring the SCM role on *ev1*. Call SMIT and follow the indicated path, or call SMIT with the fastpath name:

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Servers
         -> DFS (Distributed File Service) System Control Machine
            (fastpath = mkdfsscm)
```

```
                          DFS System Control Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                      [Entry Fields]
* CELL name                                          [/.../itsc.austin.ibm.com>
* SECURITY Server                                    [ev1]
  CDS Server (If in a separate network)              []
* Cell ADMINISTRATOR's account                       [cell_admin]
* LAN PROFILE                                        [/.../itsc.austin.ibm.com>
```

The following progress report is produced:

```
Enter password for DCE account cell_admin:
Current state of DCE configuration:
cds_cl       COMPLETE   CDS Clerk
cds_srv      COMPLETE   Initial CDS Server
dts_local    COMPLETE   Local DTS Server
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
sec_srv      COMPLETE   Security Server (Master)
Enter password for DCE account cell_admin

Password must be changed!
Configuring DFS System Control Machine (dfs_scm)...
DFS System Control Machine (dfs_scm) configured successfully

Current state of DFS configuration:
dfs_scm      COMPLETE   DFS System Control Machine
```

The following command would have had the same effect:

```
# mkdfs dfs_scm
```

There are two processes always running on a DFS SCM role machine. A third process could also be present if a *Binary Distribution* role were configured.

- **upserver —** Responsible for the distribution of the administrative lists to all server machines in the domain.
- **bosserver —** Also called the *Basic Overseer Server* process; it is responsible for keeping the system outages at a minimum.
- **upclient** — For the retrieval of binary files from the *Binary Distribution Machine*. This is not discussed in this setup.

4.2.2 Configuring a Fileset Location Database Machine

The next machine to configure is the FLDB role. We put it also on *ev1*. Call SMIT and follow the indicated path, or call SMIT with the fastpath name:

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Servers
         -> DFS Fileset Database Machine
            (fastpath = mkdfsfldb)
```

```
                        DFS Fileset Database Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                      [Entry Fields]
  Additional GROUP to administer filesets on this    []
      machine
  DFS System CONTROL machine to get                  []
      administration lists from
  FREQUENCY to update administration lists (in sec)  []
  LOG file for administration list updates           []
* CELL name                                          [/.../itsc.austin.ibm.com]
* SECURITY Server                                    [ev1]
  CDS Server (If in a separate network)              []
* Cell ADMINISTRATOR's account                       [cell_admin]
* LAN PROFILE                                        [/.../itsc.austin.ibm.com]
```

The following progress report is produced:

```
Enter password for DCE account cell_admin:

Current state of DCE configuration:
cds_cl       COMPLETE   CDS Clerk
cds_srv      COMPLETE   Initial CDS Server
dts_local    COMPLETE   Local DTS Server
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
sec_srv      COMPLETE   Security Server (Master)

Enter password for DCE account cell_admin:

Password must be changed!
Configuring DFS Fileset Database Machine (dfs_fldb)...
Waiting (up to 5 minutes) for Fileset Database machines to elect
  a synchronization site...
       /.../itsc.austin.ibm.com/hosts/ev1 has been elected
synchronization site for the Fileset Location Database.
Waiting (up to 5 minutes) for the server entry for /.:/hosts/ev1
  to be added to the fileset location database.
Server entry for /.:/hosts/ev1 has been added
  to the fileset location database.
DFS Fileset Database Machine (dfs_fldb) configured successfully

Current state of DFS configuration:
dfs_fldb     COMPLETE   DFS Fileset Database Machine
dfs_scm      COMPLETE   DFS System Control Machine
```

The following command would have had the same effect:

```
# mkdfs dfs_fldb
```

Each fileset database (FLDB) machine runs the following processes:

- **flserver —** A *fileset location server* process in charge of tracking the locations of all filesets in a cell and recording changes in the FLDB.
- **upclient(1) —** A process to retrieve configuration files from the SCM machine.
- **bosserver —** A monitoring process for the required processes.
- **upclient(2) —** For the retrieval of binary files from the *binary distribution machine (BDM)*. The BDM is not discussed in this setup.

4.2.3 Configuring the DFS File Server Machine

The configuration of the first real file server role machine begins here. Call SMIT and follow the indicated path, or call SMIT with the fastpath name:

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Servers
         -> DFS File Server Machine
               (fastpath = mkdfssrv)
```

```
                          DFS File Server Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                        [Entry Fields]
  Additional GROUP to administer filesets on this     []
      machine
  Load LFS kernel extension?                          [yes] +
  DFS System CONTROL machine to get                   []
      administration lists from
  FREQUENCY to update administration lists (in sec)   []
  LOG file for administration list updates            []
* CELL name                                           [/.../itsc.austin.ibm.com]
* SECURITY Server                                     [ev1]
  CDS Server (If in a separate network)               []
* Cell ADMINISTRATOR's account                        [cell_admin]
* LAN PROFILE                                         [/.../itsc.austin.ibm.com]
```

The following progress report is produced:

```
Enter password for DCE account cell_admin:

Current state of DCE configuration:
cds_cl       COMPLETE   CDS Clerk
cds_srv      COMPLETE   Initial CDS Server
dts_local    COMPLETE   Local DTS Server
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
sec_srv      COMPLETE   Security Server (Master)

Enter password for DCE account cell_admin:
Password must be changed!
Configuring DFS File Server Machine (dfs_srv)...
DFS File Server Machine (dfs_srv) configured successfully

Current state of DFS configuration:
dfs_fldb     COMPLETE   DFS Fileset Database Machine
dfs_scm      COMPLETE   DFS System Control Machine
dfs_srv      COMPLETE   DFS File Server Machine
```

The following command would have had the same effect:

```
# mkdfs -e dfs_srv
```

The following processes run on a *file server machine:*

- **ftserver —** Accepting interoperation with the fts command suite.

- **upclient(1)** — Responsible for the contacts with the *upserver* of the SCM machine.
- **fxd** — Responsible for the initialization of the *file exporter*, which runs as an extension of the kernel on each file server machine. It provides the same services across the network as the operating system provides for a local disk.
- **dfsbind —** An interface to the CDS and Security Services of the DCE cell.
- **bosserver** — Supervises the required processes.
- **upclient(2)** — (Optionally) For the retrieval of binary files from the *binary distribution machine*. This is not discussed in this setup.

After the successful configuration of these DFS server (roles), we can start the build-up of the DFS file system. The file system starts with a *root* that can be addressed by several names. The DFS hierarchy starts from the ***fs junction*** point in the CDS. This junction is the root of the filespace hierarchy and can be addressed by several names:

`/.../itsc.austin.ibm.com/fs`	global absolute name
`/.:/fs`	cell relative name (only from within the cell)
`/:`	the special short name (only from within the cell)

4.2.4 Configuring a DFS Root Fileset

Creating a DCE/LFS fileset requires the following steps when the fileset is created on a new aggregate:

1. Making a logical volume (`mklv`)
2. Creating an aggregate (`newaggr`)
3. Creating the DFS root directory by:
   - Editing the /var/dce/dfs/dfstab (/opt/dcelocal/var/dfs/dfstab) file
   - Exporting the aggregate (`dfsexport`)
   - Creating the fileset (`fts create`)
   - Creating a mount point in the DFS namespace (`fts crmount`)

The root directory will be contained in the *root.dfs* fileset. This fileset is a real LFS fileset and is contained in an *aggregate*. It is advisable to reserve this aggregate exclusively for the root.dfs fileset. Let's have a look into the details.

1. Create a logical volume:

   The aggregate does not need to be very large. Creating a logical volume with one block (4 MB) is sufficient. We suggest you do not use the root directory /: to store files and data. This directory should only be used to hold subdirectories or mount points for other filesets.

   Create a logical volume with the following command:

   ```
   # mklv -t lfs -y lfsroot rootvg 1
   ```

2. Create an aggregate

   The new logical volume has to be initialized as an *aggregate*. An *aggregate* is able to contain several filesets, while a basic logical volume can only house one JFS (non-LFS) fileset. So, make the /dev/lfsroot logical volume an LFS logical volume:

```
# newaggr -aggregate /dev/lfsroot -block 8192 -frag 1024 -overwrite
*** Using default initialempty value of 1.
*** Using default number of (8192-byte) blocks: 511
*** Defaulting to 50 log blocks    (maximum of 5 concurrent transactions).
/dev/rlfsroot: Already marked as non-BSD.
Done.  /dev/rlfsroot is now an Episode aggregate.
```

The command newaggr is only available to be used on a logical volume. It is used to prepare a logical volume for holding LFS fileset(s) as opposed to a single JFS.

3. Create the DFS root directory

   Using standard OSF commands, this step would require creating the /opt/dcelocal/var/dfs/dfstab file, exporting the aggregate /dev/lfsroot, creating the root.dfs fileset, and mounting it at the DFS junction. This can all be achieved with the IBM-only mkdfslfs command as follows:

   ```
   # mkdfslfs -r -d /dev/lfsroot -n lfsroot
   ```

   The -r flag specifies that the *root.dfs* fileset be created. Below you find the output of this command:

   ```
           readWrite   ID 0,,1  valid
           readOnly    ID 0,,2  invalid
           backup      ID 0,,3  invalid
   number of sites: 1
   server        flags     aggr     siteAge  principal    owner
   ev1           RW        lfsroot  0:00:00  hosts/ev1    <nil>
   Fileset 0,,1 created on aggregate lfsroot of /.:/hosts/ev1
   ```

   At this point, the DFS root directory /: is available. The name root.dfs was automatically assigned to the root fileset (that is what the -r flag did), and three file ID numbers were reserved, one for each replica type that can exist for this fileset.

   Before being able to access the DFS namespace, you have to configure a DFS client.

4.2.5 Configuring a DFS Client

Before configuring a DFS client you should verify the DCE core services are configured properly. This task is the same on every machine that will house DFS clients.

Make sure you have followed all the preparation steps outlined in 3.2, “Preparing for DCE Configuration on AIX” on page 38. The separate file system for the DFS cache has to be mounted before you begin. If you want to choose a cache size other than 10 MB (default), you should call SMIT. Otherwise, type the following command (you will not need cell_admin’s password):

```
# mkdfs dfs_cl
Configuring DFS Client Machine (dfs_cl)...
dfsd: start sweeping disk cache files ....
dfsd: All DFS daemons started.
DFS Client Machine (dfs_cl) configured successfully

Current state of DFS configuration:
dfs_cl       COMPLETE   DFS Client Machine
dfs_fldb     COMPLETE   DFS Fileset Database Machine
dfs_scm      COMPLETE   DFS System Control Machine
dfs_srv      COMPLETE   DFS File Server Machine
```

4.2.6 Testing Access to the DFS Root Fileset

You have to wait some time before being able to access the DFS filespace because at each restart, the DFS file server has to reestablish the state of its tokens with its DFS clients. This happens even if it is the first time you configure DFS and there are no clients to wait for. When you type the following command for the first time:

```
ev1::/-> cd /:
/bin/ksh: /:: Permission denied.
```

You get the permission denied message. Look at the output of the `ls -l /:` command:

```
ev1::/-> ls -l /:
lrwxrwxrwx   1 root system 27 Feb 12 11:17 /: -> /.../itsc.austin.ibm.com/fs
```

This output looks good, but it does not show the effective permissions. Look at the ACLs for the root directory:

```
ev1::/-> dcecp -c acl show /:
{user_obj rwxcid}
{group_obj ------}
{other_obj ------}
ev1::/->
```

See 4.2.7, “Fixing Access Permissions (ACLs)” on page 84, for explanations on initial ACL settings and how to set correct ACLs. In order to test DFS access, we set read permission for everybody on the root directory. It might be necessary to dce_login again if the previous dce_login occurred before the machine was configured for DFS. Then enter the following command:

```
ev1::/-> chmod a+rx /:
```

After a successful completion of the above modification, you can execute the following commands to check the status of the file system:

1. Enter the DFS root directory:

   ```
   ev1::/-> cd /:
   ev1::/-> pwd
   /:
   ```

2. List the contents of the *dfstab* file:

   ```
   ev1::/-> cat /opt/dcelocal/var/dfs/dfstab
   #blkdev              aggname                  aggtype  aggid [UFS fsid]
   /dev/lfsrootv        lfsroot                  lfs      1
   ```

3. List the aggregates that are exported on this machine:

   ```
   ev1::/-> dfsexport
   Aggregate Device     Name                Type    Aggr ID    Non-LFS fileset ID
   /dev/lfsroot         lfsroot             lfs     1
   ```

4. List the FLDB:

   ```
   ev1::/-> fts lsfldb
   root.dfs
           readWrite   ID 0,,1  valid
           readOnly    ID 0,,2  invalid
           backup      ID 0,,3  invalid
   number of sites: 1
   server        flags     aggr     siteAge  principal    owner
   ev1           RW        lfsroot  0:00:00  hosts/ev1    <nil>
   ```

```
---------------
Total FLDB entries that were successfully enumerated: 1 (0 failed; 0 wrong aggrtype)
```

4.2.7 Fixing Access Permissions (ACLs)

When a DCE LFS fileset is created and mounted into the DFS file space, the owning UID is the DCE principal associated with the 0 UID, which is usually the DCE root principal. The owning group is the DCE group associated with the GID of 0, which is usually the system group. The default permissions are 700. If you are DCE-authenticated as a member of the subsys/dce/dfsadmin group, you can change these permissions by either using the chmod command or acl_edit or dcecp acl. The cell_admin DCE principal is normally in the *dfsadmin* group. When a fileset is created, it is important that the ownership and permissions are set up according to how the fileset will be used.

Before we can create the mount point for another fileset, it is necessary to adapt the permissions of the parent DFS object (here the /: directory). The mounting of the new fileset will otherwise be denied. Normally, DFS objects are protected by access control lists (ACL), but when a fileset is first created in the DFS filespace, the permissions for the mount point directory default to the AIX mode permissions 700, with an owner of UID 0. An ACL for this directory does not really exist at this time. The default AIX authorizations set for the root directory /: are, in general, not sufficient for allowing new *subdirectory inserts*.

```
ev1::/-> pwd
/:
ev1::/-> ls -al
total 3
drwxr-xr-x   2 root     system       256 Jun 01 16:56 .
dr-xr-xr-x   3 root     system      2048 May 12 1989  ..
ev1::/->
ev1::/-> touch testfile
touch testfile: The file access permissions do not allow the specified action.
```

Remember, in 4.2.6, “Testing Access to the DFS Root Fileset” on page 83, we had to fix the AIX permissions (to 755) to be able to list the contents of the DFS root directory and to change into it. Write permission is not sufficient for inserting a new directory and/or file; we need insert permission, which can be granted by changing the ACL.

The owner of the DFS root (/:) directory is user *root*, belonging to group *system*. We logged in for UNIX as this user, but this user is not authenticated for DCE. We logged in to DCE as *cell_admin*. Several solutions are possible to adapt the authorizations so that we can insert and work on the /: directory.

Let’s first show the different ACLs for the /: directory.

```
ev1::/-> dcecp
dcecp> acl show /:
{user_obj  rwxcid}
{group_obj r-x---}
{other_obj r-x---}

dcecp> acl show -io /:
{user_obj  rwxc--}
{group_obj rwx---}
{other_obj rwx---}
```

```
dcecp> acl show -ic /:
{user_obj  rwxcid}
{group_obj rwx-id}
{other_obj rwx-id}
```

When using `acl modify` of dcecp or `acl_edit`, an ACL will be created from the current setting of the AIX file permissions. To allow for the creation of new subdirectories in /:, at least insert authority is required. All modifications are done with an `acl modify` subcommand of dcecp.

```
dcecp> acl modify /: -add {user cell_admin rwxcid}
dcecp> acl show /:
{mask_obj  r-x---}
{user_obj  rwxcid}
{user cell_admin rwxcid  effective r-x---}
{group_obj r-x---}
{other_obj r-x---}
```

The *mask_obj* is created from the existing UNIX permission bits for *group*. It limits *cell_admin*'s permission; so, we also have to adapt the *mask_obj* to get the required authorizations for *cell_admin*. You need to decide how many privileges you want to give *cell_admin*; it does not need to get the full permission set granted. In the following example, we open up the *mask_obj* completely.

```
dcecp> acl modify /: -change {mask_obj rwxcid}
dcecp> acl show /:
{mask_obj rwxcid}
{user_obj rwxCid}
{user cell_admin rwxcid}
{group_obj r-x---}
{other_obj r-x---}
```

If you want *cell_admin* to inherit ACLs on files and directories created within the root fileset, you can now create an entry for the *initial object creation* and the *initial container creation* ACLs. Run the following commands:

```
dcecp> acl modify -io /: -add {user cell_admin rwxcid}
dcecp> acl modify -io /: -change {mask_obj rwxcid}
dcecp> acl modify -ic /: -add {user cell_admin rwxcid}
dcecp> acl modify -ic /: -change {mask_obj rwxcid}
```

Note: The inheritance is limited to the fileset boundaries. For filesets mounted underneath /:, the same permission problems will arise again. Note also that the `umask` setting may further limit the initial ACL settings. If *cell_admin* creates a file, /:/testfile, it will not have write permission for *group* and *others*, unless you had changed the `umask` for the user with UID 100.

At this point, some AIX users are allowed to do what is required. To find the permissions you will be granted, just determine who you are, to which group you belong, and who has the ownership of the object. If you are not affiliated with the owner or the primary group, look for the *other* authorizations.

```
ev1::/-> whoami
root

ev1::/-> pwd
/:
ev1::/-> ls -las
total 5
   1 drwxrwxrwx   4 root     system       384 Jan 30 11:37 .
```

```
0 dr-xr-xr-x   3 root     system      2048 May 12 1989  ..
```

With this information, we see that the *user_obj* is *root* (actually a user with user ID 0), and the *group_obj* is *system* (actually a group with group ID 0). However, when logged in to DCE as *cell_admin*, we have a user ID of 100, which is usually assigned to AIX user *guest*. So, we are not associated with the *user_obj* and need a separate *user* entry if we want to give cell_admin full permissions.

With this new setting, we should have all required permissions for mounting the new fileset.

4.2.8 Adding Another Fileset

We can add many filesets within a single LFS aggregate. However, we suggest not adding another fileset to the lfsroot aggregate. We suggest creating another aggregate for other filesets. Adding other filesets can be done on every DFS server machine.

The DFS setup that we want to achieve here is very simple, and it reflects the fact that the DFS system will be used by both AIX and OS/2 clients. Figure 38 shows an example of a file subtree that will only be used by OS/2 clients (below mount point *warp001*) and another one that contains all the home directories for users. The /home directory is an ordinary directory within the *root.dfs* fileset and contains a subdirectory for each user, each of which is a mount point for a separate fileset (shown is fileset *brice.ft* mounted to /:/home/brice). The home directories will be accessed by AIX and OS/2 users.

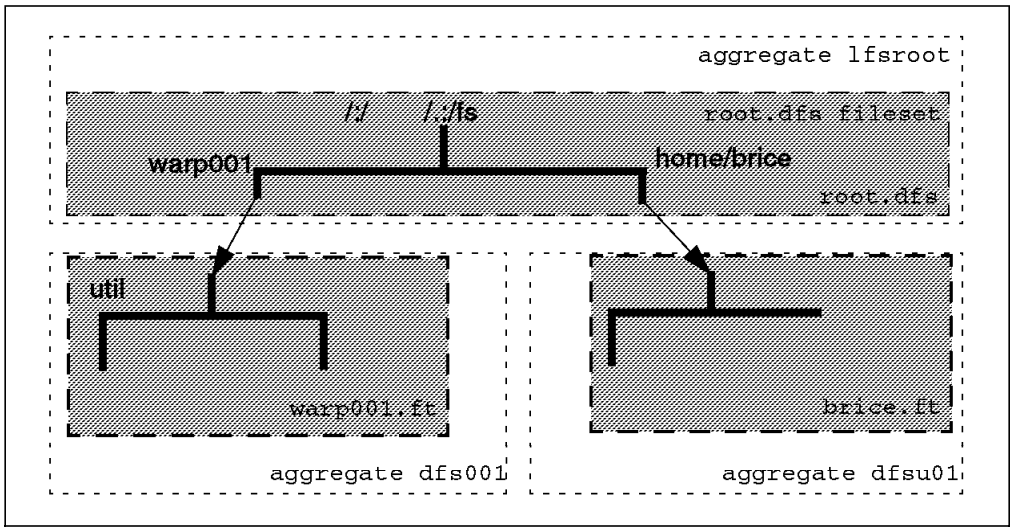


Figure 38. Simple Fileset Splitup for AIX and OS/2 Warp

Those two filesets may reside in the same or in different aggregates. In the rest of this section, we want to explain how *warp001* is created. For that, we want to add another fileset, *warp001.ft*:

1. Create a logical volume, *dfs001,* with five blocks:

   ```
   ev1::/-> mklv -t lfs -y dfs001 rootvg 5
   ```

   You can list the logical volumes:

```
ev1::/-> lsvg -l rootvg
rootvg:
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
........
lfsroot             lfs        1     1     1    open/syncd    N/A
dfs001              lfs        5     5     1    open/syncd    N/A
```

2. Create a new aggregate on /dev/dfs001:

   ```
   ev1::/-> newaggr -aggregate /dev/dfs001 -bl 8192 -fr 1024
   ```

   ```
   *** Using default initialempty value of 1.
   *** Using default number of (8192-byte) blocks: 2559
   *** Defaulting to 50 log blocks  (maximum of 5 concurrent transactions).
   /dev/rdfs001: Marked as not a BSD file system any more.
   Done.  /dev/rdfs001 is now an Episode aggregate.
   ```

   If you have some problem when creating the aggregate, try to use the -overwrite option as shown here:

   ```
   ev1::/-> newaggr -aggregate /dev/dfs001 -bl 8192 -fr 1024 -overwrite
   ```

3. Export the aggregate:

   ```
   ev1::/-> mkdfslfs -d /dev/dfs001 -n dfs001
   ```

   At this point, we created two aggregates. Aggregates are the containers for the LFS filesets, and as explained earlier on, several filesets can be contained in one aggregate. Aggregates do not have a mount point. A mount point can be specified only for a fileset contained in an aggregate.

   - The lfsroot aggregate contains the *root.dfs* fileset, and of course the root directory (/:).
   - The *dfs001* aggregate has to contain a *warp001.ft* fileset mounted on /:/warp001.

4. Create a fileset with a mount point:

   ```
   ev1::/-> mkdfslfs -f warp001.ft -m /:/warp001 -n dfs001
   ```

   ```
           readWrite   ID 0,,4  valid
           readOnly    ID 0,,5  invalid
           backup      ID 0,,6  invalid
   number of sites: 1
   server           flags     aggr   siteAge principal      owner
   ev1                 RW       dfs001 0:00:00 hosts/ev1      <nil>
   Fileset 0,,4 created on aggregate dfs001 of /.:/hosts/ev1
   ```

   If you forgot to log in to DCE and the following message appears,

   ```
   fts crmount: error making mount point for /:/warp001: Permission denied
   Cannot create mount point /:/warp001 in DFS file space.
   ```

   you should log in as cell_admin, and enter the following command:

   ```
   ev1::/-> fts crmount -dir /:/warp001 -fileset warp001.ft
   ```

   At this point, you should have two filesets available. If you issue the fts lsfldb command, you should receive the following output:

   ```
   ev1::/-> fts lsfldb
   root.dfs
           readWrite   ID 0,,1  valid
           readOnly    ID 0,,2  invalid
           backup      ID 0,,3  invalid
   number of sites: 1
   ```

```
  Release repl: maxAge=2:00:00; failAge=1d0:00:00; reclaimWait=18:00:00
   server           flags     aggr   siteAge principal      owner
ev1.itsc.austin.ibm RW,RO    lfsroot 0:00:00 hosts/ev1      <nil>
warp001.ft
        readWrite   ID 0,,4  valid
        readOnly    ID 0,,5  invalid
        backup      ID 0,,6  invalid
number of sites: 2
  Release repl: maxAge=2:00:00; failAge=1d0:00:00; reclaimWait=18:00:00
   server           flags     aggr   siteAge principal      owner
ev1.itsc.austin.ibm RW,RO    dfs001  0:00:00 hosts/ev1      <nil>
----------------------
Total FLDB entries that were successfully enumerated: 2
```

5. In order to be able to access the new fileset, set the appropriate ACLs as described in 4.2.7, “Fixing Access Permissions (ACLs)” on page 84. For the sake of simplicity, we just modify the UNIX permissions:

   ```
   # chmod a+rwx /:/warp001
   ```

4.3 A DFS Client on OS/2 Warp

Machine *EV5* and *EV6* will be used as DFS clients.

4.3.1 Preparing OS/2 Warp for DFS

A preliminary requirement for a DFS client on an OS/2 Warp platform is to be part of the DCE cell. This has already been done in 3.6, “Configuring DCE Clients on OS/2 Warp” on page 56. At the same time, we also selected the configuration of the DFS client. As a result, the following line has been added to the *CONFIG.SYS* file:

```
IFS=C:OPTDCELOCALBINDFSCLI.IFS
```

This entry points to an *Installable File System* driver that receives the requests from OS/2. This acts as a *kernel extension* of OS/2 Warp.

4.3.2 Starting Up the DFS Client

Before being able to use the DFS filespace from OS/2 Warp, the dfsd daemon has to be started:

```
EV5-> DFSD
dfsd:  The number of DFS client threads is 2 .
DFS Cache Manager (DFSD.EXE) registering with IFS driver.
dfsd: Junction Life TTL set to 143600
dfsd: Prefix Life TTL set to 143600
dfsd: NotFound Life TTL set to 6400
dfsd: OS/2-style Attribute support has NOT been activated
dfsd: OS/2 Extended Attribute support has NOT been activated
dfsd: start sweeping disk cache files ....
dfsd: All DFS daemons started.
dfsd: Initialization is continuing........

Attaching drive + to the DFS File System.
The DFS path prefix associated with this drive is  /:
Successfully Attached drive D.

The DCE DFS Daemon has initialized successfully.
```

The dfsd has been started, and the DFS filespace has been associated with drive letter **d:**. Two processes have to be active in OS/2 to enable DFS client support:

1. The dfsd process is responsible for the caching.

   The dfsd command initiates the DFS cache manager and starts the related daemons. A lot of parameters can be specified with this command, for example:

   - blocks (number of cache blocks)
   - mountdir (DFS subdirectory to become top-level directory for drive d:)

   Specifying values here overrides the defaults and the values that are stored in the *optdcelocaletccachinfo* file. This file can contain information like:

   - Initial cache settings
   - Time to Live settings (TTL) associated with the DFSBIND function
   - DFS drives to be auto attached at dfsd initialization

2. The dfsbind process is also started with dfsd. It is responsible for the contacts with CDS and Security.

All mounts from OS/2 are associated with a drive letter. After starting dfsd, additional mounts can be created with the dfsdrive command:

```
dfsdrive -attach e: /:/warp001
```

From here on, drive letter e: corresponds to */.:/fs/warp001* in the DFS filespace.

Note: NO replication has been done for any fileset.

4.4 Replicating Filesets on AIX

Before reading this section, we recommend you read the discussion about DFS replication in 7.2, “DFS Replication” on page 252:

- Why do we need to replicate a fileset?
- Which filesets to replicate?

More examples on how to set up DFS replication can be found in the scenario configuration instructions in chapter Chapter 5, “Implementing Various LAN/WAN Scenarios” on page 105. There we replicate the filesets before we create a mount point to prevent anyone from obtaining unwanted write access.

In this section we describe how to replicate a fileset which had been created before it was decided to replicate it. This task is based on Figure 36 on page 75. Therefore, we assume the machine *ev1* is housing the read/write root.dfs fileset.

However, there is not a right or a wrong way of setting up replication. You can configure the whole fileset tree with read/write filesets and regular mount points, populate the filesets, and then decide some time later on what to replicate. Or you can create the read-only replicas immediately after creating the read/write fileset, and then create the mount point(s), if you already know that you want to replicate the fileset.

If we use release replication (see 4.4.4.1, “Configuring Fileset Using Release Replication” on page 94) we must replicate the *root.dfs* fileset on the same machine that physically houses this *root.dfs* fileset before we can replicate any

other fileset. The *root.dfs* fileset is the fileset that houses the root directory (/:) of DFS. This section discusses in detail the following issues:

- Configuring the replication server on *ev1*
- Replicating the *root.dfs* fileset
- Replicating the *warp001.ft* fileset on *ev4*

4.4.1 Configuring and Starting a Replication Server on ev1

Before any replication can be done, each fileset server machine which intends to replicate some of its filesets must be configured with a *replication server* role. We will now configure such a replication function on *ev1*.

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Servers
         -> DFS Fileset Replication Server Machine
            (fastpath = mkdfsrepsrv)
```

```
                    DFS Fileset Replication Server Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry
Fields]
* Cell ADMINISTRATOR's account                       [cell_admin]
```

```
Enter password for DCE account cell_admin:

Password must be changed!
Configuring DFS Replicated Fileset Server Machine (dfs_repsrv)...
DFS Replicated Fileset Server Machine (dfs_repsrv) configured
successfully

Current state of DFS configuration:
dfs_cl       COMPLETE   DFS Client Machine
dfs_fldb     COMPLETE   DFS Fileset Database Machine
dfs_repsrv   COMPLETE   DFS Replicated Fileset Server Machine
dfs_scm      COMPLETE   DFS System Control Machine
dfs_srv      COMPLETE   DFS File Server Machine
          Press Enter to continue
```

The following command has the same effect:

```
# mkdfs dfs_repsrv
```

4.4.2 Replicating the Root Directory on ev1

A replication server role was configured above. Before being able to replicate any other fileset, the *root.dfs* must be replicated. The current *root.dfs* is mounted at the root directory (/:) via a *regular mount point*. The replication steps of the *root.dfs* that we describe will create another mount point for this fileset, a *read/write mount point*, which will be named */:/.rw*. The read/write fileset will be moved to /:/.rw and will only be accessible via this path. The former root directory (/:) will access the readonly fileset. So, at the end of this section, we will have the following picture:

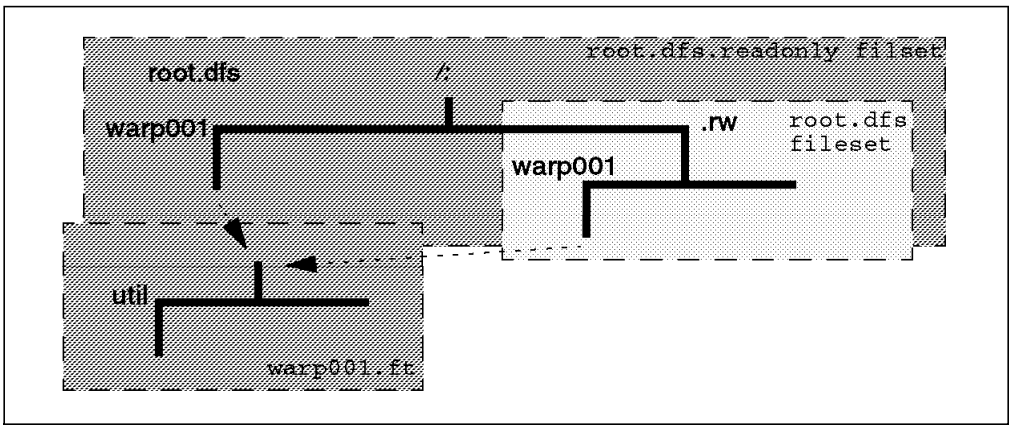


Figure 39. DFS Status After Replicating root.dfs

We assume that *root.dfs* is properly configured on an LFS fileset. See 4.2.4, "Configuring a DFS Root Fileset" on page 81 for information on how to configure the root.dfs.

The following are the steps to replicate the *root.dfs* fileset. We assume we are on *ev1* where the *root.dfs* fileset is located.

1. Create a read/write mount point for *root.dfs*:

   ```
   # fts crmount /:/.rw root.dfs -rw
   ```

2. Define the replication type for *root.dfs*:

   ```
   # fts setrepinfo -fileset root.dfs -rel
   fts setrepinfo: Using default value for maxage of 2:00:00
   fts setrepinfo: Using derived value for failage of 1d0:00:00
   fts setrepinfo: Using default value for reclaimwait of 18:00:00
   ```

   With this command, you apply the type of replication to the root.dfs fileset. The type of replication here is *release*.

3. Define the replication site (on the same machine):

   ```
   # fts addsite -fileset root.dfs -server /.:/hosts/ev1 -aggr lfsroot
   Added replication site /.:/hosts/ev1 lfsroot for fileset root.dfs
   ```

4. Create the read-only fileset and force replication from the read/write source:

   ```
   # fts release -fileset root.dfs
   Released fileset root.dfs successfully
   ```

5. Leave the DFS root directory; otherwise you are still connected to the read/write fileset of the /: directory:

   ```
   # cd
   ```

6. Force the local cache manager to refresh its knowledge about the fileset configuration:

   ```
   # cm checkfilesets
   ```

7. Check whether you can create a file in /: now (should be denied):

   ```
   # cd /:
   # touch testfile
   touch: 0652-046 Cannot create testfile.
   ```

8. You can create the *testfile* only via the read/write mount point:

```
# cd /:/.rw
# touch testfile
# ls
```

At this point, you can start to replicate other filesets. In the next sections, we will replicate the *warp001* fileset on *ev4*.

Verify the status of the fileset in the FLDB:

```
# fts lsfldb
```

```
root.dfs
        readWrite   ID 0,,1  valid
        readOnly    ID 0,,2 valid
        backup      ID 0,,3 invalid
number of sites: 1
  Release repl: maxAge=2:00:00; failAge=1d0:00:00; reclaimWait=18:00:00
   server           flags     aggr   siteAge principal      owner
ev1                 RW,RO    lfsroot 0:00:00 hosts/ev1      <nil>
----------------------
```

Notice that the root.dfs readOnly fileset version is now marked valid.

Note

When you plan to replicate filesets, we recommend replicating root.dfs immediately after having created it and before you create any other filesets. This prevents other DFS clients from getting unwanted access to the read/write fileset of root.dfs.

If a DFS client had connected to /: before you created the root.dfs replica, it continues to have write access. To stop read/write access, users of this DFS client must leave that fileset, and cm ckeckfilesets must be run on that node.

Each cache manager automatically refreshes its fileset information every hour, which is the equivalent function to the cm ckeckfilesets command. So, unwanted fileset access is usually automatically terminated after one hour, unless users are still in this directory.

4.4.3 Configuring File and Replication Servers on ev4

Since *ev4* is going to be a file server machine in our DCE/DFS scenario, it has to be configured with a fileset server. This is done in the usual way.

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Servers
         -> DFS File Server Machine
            (fastpath = mkdfssrv)
```

```
                          DFS File Server Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Additional GROUP to administer filesets on this machine[]
  Load LFS kernel extension?                           [yes] +
  DFS System CONTROL machine to get                    [/.:/hosts/ev1]
      administration lists from
  FREQUENCY to update administration lists (in seconds) []
  LOG file for administration list updates             []
* CELL name                                            [/.../itsc.austin.ibm.com]
* SECURITY Server                                      [ev1]
  CDS Server (If in a separate network)                []
* Cell ADMINISTRATOR's account                         [cell_admin]
* LAN PROFILE                                          [/.:/lan-profile]
```

Here is the progress list:

```
Enter password for DCE account cell_admin:

File /opt/dcelocal/var/dfs/dfstab already exists--
  check it for old entries after configuration is complete.
Password must be changed!
Configuring DFS File Server Machine (dfs_srv)...
Waiting (up to 5 minutes) for the server entry for /.:/hosts/ev4
  to be added to the fileset location database.
Server entry for /.:/hosts/ev4 has been added
  to the fileset location database.
DFS File Server Machine (dfs_srv) configured successfully

Current state of DFS configuration:
dfs_srv      COMPLETE   DFS File Server Machine
```

This command would have had the same effect:

```
# mkdfs -s /.:/hosts/ev1 -e dfs_srv
```

We also have to configure a *replication server* role on *ev4*:

```
# smitty dce
   -> Configure DCE/DFS
      -> Configure DCE/DFS Servers
         -> DFS Fileset Replication Server Machine
            (fastpath = mkdfsrepsrv)
```

```
                    DFS Fileset Replication Server Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
* Cell ADMINISTRATOR's account                     [cell_admin]
```

Here is the progress report:

```
Enter password for DCE account cell_admin:

Password must be changed!
Configuring DFS Replicated Fileset Server Machine (dfs_repsrv)...
DFS Replicated Fileset Server Machine (dfs_repsrv) configured successfully

Current state of DFS configuration:
dfs_repsrv   COMPLETE   DFS Replicated Fileset Server Machine
dfs_srv      COMPLETE   DFS File Server Machine
```

After this, we are ready to start the replication of the *warp001.ft* fileset on server *ev4.*

4.4.4 Setting Up Replication for Another Fileset

We assume we want to replicate the *warp001.ft* fileset on *ev4*.

For all types of replication (release or schedule), you must have previously installed and configured a DFS fileset server and a DFS replication server (repserver process) on the machine that is to serve as DFS fileset replication server (*ev4* in this case). See 4.4.3, “Configuring File and Replication Servers on ev4” on page 92.

We assume in this example, that *ev1* is the system control machine. According to Figure 39 on page 91, we assume we have a fileset called *warp001* that we want to replicate on another machine *ev4*. The warp001 fileset is physically housed on *ev1*. We assume also the regular mount point of the warp001 fileset is /:/warp001 before and after replication.

4.4.4.1 Configuring Fileset Using Release Replication

Actions have to be taken on both the source and the target platform. Replication of a fileset to another platform than the housing machine always first requires a replication on the local machine. This replication is not a full replication, but is merely a pointer-like cloning. This aspect is explained in detail in 7.2, “DFS Replication” on page 252.

1. On *ev1*, which physically houses the fileset:
   a. Set the replication parameters for release replication:

      ```
      # fts setrepinfo -fileset warp001.ft -rel
      ```

   b. Define the same machine as a replication site:

      ```
      # fts addsite -fileset warp001.ft -server /.:/hosts/ev1 -aggr dfs001
      ```

   c. Create the read-only fileset and force replication from the read/write source:

      ```
      # fts release -fileset warp001.ft
      ```

2. On a target machine (*ev4*):
   a. Create an aggregate large enough to house the fileset:

      Suppose the aggregate *dfs001* has five blocks of 4 MB. We have to provide at least that same size on the source machine.

      Create a logical volume as large as on *ev1*:

      ```
      # mklv -t lfs -y dfs001 rootvg 5
      ```

   b. Create an aggregate on /dev/dfs001:

```
# newaggr -aggreg /dev/dfs001 -bl 8192 -fr 1024 -overwrite
```

c. Export the aggregate:

```
# mkdfslfs -d /dev/dfs001 -n dfs001
```

d. Define the new replication site:

```
# fts addsite -fileset warp001.ft -server /.:/hosts/ev4 -aggr dfs001
```

e. Create the read-only fileset and force replication from the read/write source:

```
# fts release -fileset warp001.ft
Released fileset warp001.ft successfully
```

Since the /:/warp001 directory with the regular mount point existed before, we need not force an update of the /: parent directory at this point.

f. Configure a DFS client if it does not already exist:

```
# mkdce dfs_cl
```

g. Leave the /:/warp001 directory which might still be connected to the read/write fileset if you had accessed it from *ev4* before you created the first replica for warp001.ft.

```
# cd /:
```

h. In order to be able to access the new fileset, set the appropriate ACLs as described in 4.2.7, “Fixing Access Permissions (ACLs)” on page 84, if you did not do it right after creation of the fileset. For the sake of simplicity, we just modify the UNIX permissions:

```
# chmod a+rwx /:/.rw/warp001
# fts release -fileset warp001.ft
```

i. Force the local cache manager to read the new fileset information:

```
# cm checkfilesets
```

j. Check whether you can create a file in /:/warp001 now:

```
# cd /:/warp001
# touch testfile
touch: 0652-046 Cannot create testfile.
```

You are not able to create files in /:/warp001 because this path accesses the read-only fileset. You can access the read/write fileset via /:/.rw/warp001, or you can create a read/write mount point, /:/.warp001, if you do not plan to keep /:/.rw available for daily use.

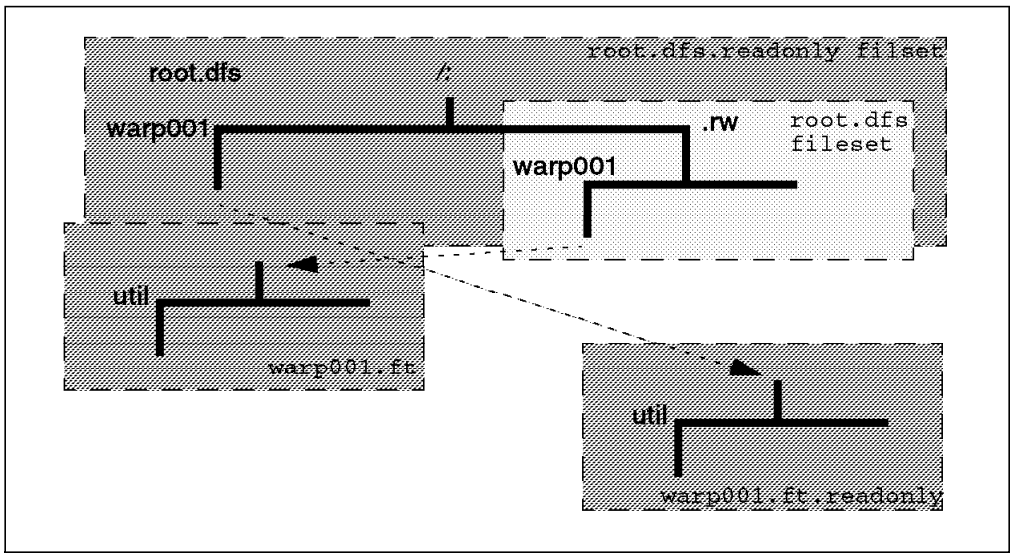


Figure 40. DFS Status After Replicating warp001.ft

To create the read/write mount point, follow these steps:

1. Create the mount point:

   ```
   # fts crmount /:/.rw/.warp001 warp001.ft -rw
   ```

2. Update the read-only copy of root.dfs to make the /:/.warp001 directory available:

   ```
   # fts rel root.dfs
   ```

3. Force the local cache manager to read the new fileset information:

   ```
   # cm checkfilesets
   ```

Now you can access the read/write fileset and create a file.

High availability of read-only filesets

If we create an additional DFS file server to increase the availability of a fileset, for instance of the warp001.ft.readonly fileset, we need to make sure that all filesets above it in the fileset hierarchy are also replicated on the same server. In other words, root.dfs should be replicated on *ev4* to make sure /:/warp001 can be accessed if *ev1* fails.

4.4.4.2 Configuring a Fileset Using Scheduled Replication

With this type of replication, you do not need to use the command `fts addsite` on the machine with the read/write fileset, but you need to use it on the target machine. However, you *may* use it on the source if you *want* to create a read-only copy on this primary machine.

1. On *ev1*, which physically houses the fileset:

   a. Set the replication parameters for scheduled replication:

      ```
      # fts setrepinfo -fileset warp001.ft -sched
      ```

2. On a target machine (ev4):

   These are the steps to be done on each target machine:

   a. Create an aggregate large enough to house the fileset:

Assume the aggregate *dfs001* has five blocks of 4 MB. Create a logical volume as large as on *ev1*:

```
# mklv -t lfs -y dfs001 rootvg 5
```

b. Create an aggregate on /dev/dfs001:

```
# newaggr -aggreg /dev/dfs001 -bl 8192 -fr 1024 -overwrite
```

c. Export the aggregate:

```
# mkdfslfs -d /dev/dfs001 -n dfs001
```

d. Define the new replication site:

```
# fts addsite -fileset warp001.ft -server /.:/hosts/ev4 -aggr dfs001
```

e. Create the read-only fileset, and force replication from the read/write source using the update command:

```
#fts update -fileset warp001.ft -all
fts update: Repserver on ev1 requested to update fileset 0,,5
fts update: Repserver on ev4 requested to update fileset 0,,5
```

At this point, the fileset is replicated.

Since the /:/warp001 directory with the regular mount point existed before, we need not force an update of the parent directory (/:) at this point.

f. Configure a DFS client if it does not already exist:

```
# mkdce dfs_cl
```

g. Leave the /:/warp001 directory which might still be connected to the read/write fileset, if you had accessed it from *ev4* before you created the first replica for warp001.ft.

```
# cd /:
```

h. Force the local cache manager to read the new fileset information:

```
# cm checkfilesets
```

i. Check whether you can create a file in /:/warp001 now:

```
# cd /:/warp001
# touch testfile
touch: 0652-046 Cannot create testfile.
```

You are not able to create files in /:/warp001 because this path accesses the read-only fileset. You can access the read/write fileset via /:/.rw/warp001, or you can create a read/write mount point, /:/.warp001, if you do not plan to keep /:/.rw available for daily use.

To create the read/write mount point, issue:

```
# fts crmount /:/.rw/.warp001 warp001.ft -rw
```

j. Update the read-only copy of root.dfs to make the /:/.warp001 directory available:

```
# fts rel root.dfs
```

k. Force the local cache manager to read the new fileset information:

```
# cm checkfilesets
```

Now you can access the read/write fileset and create a file.

4.4.5 Double-Check Your Work

You can use the following commands on every DFS machine.

- Consult the FLDB:

```
# fts lsfldb

root.dfs
        readWrite   ID 0,,1  valid
        readOnly    ID 0,,2  valid
        backup      ID 0,,3  invalid
number of sites: 1
  Release repl: maxAge=2:00:00; failAge=1d0:00:00; reclaimWait=18:00:00
   server           flags     aggr   siteAge principal      owner
ev1                 RW,RO    lfsroot 0:00:00 hosts/ev1      <nil>

warp001.ft
        readWrite   ID 0,,4  valid
        readOnly    ID 0,,5  valid
        backup      ID 0,,6  invalid
number of sites: 2   <<--- Here note that number is two!!
  Release repl: maxAge=2:00:00; failAge=1d0:00:00; reclaimWait=18:00:00
   server           flags     aggr   siteAge principal      owner
ev1                 RW,RO    dfs001  0:00:00 hosts/ev1      <nil>
ev4                 RO       dfs001  0:00:00 hosts/ev4      <nil>
```

  The readOnly fileset is now marked valid.

- List the fileset header:

```
ev4::/-> fts lsheader -server ev1
Total filesets on server ev1 aggregate lfsroot (id 1): 2
root.dfs                0,,1 RW     17 K alloc     17 K quota On-line
root.dfs.readonly       0,,2 RO     17 K alloc     17 K quota On-line
Total filesets on-line 2; total off-line 0; total busy 0

Total filesets on server ev1 aggregate dfs001 (id 2): 2
warp001.ft              0,,4 RW     16 K alloc     17 K quota On-line
warp001.ft.readonlyl    0,,5 RO     17 K alloc     17 K quota On-line
Total filesets on-line 2; total off-line 0; total busy 0

Total number of filesets on server ev1: 4
```

- Access to DFS filespace:

```
# dce_login cell_admin mypasswd

# cd /:/warp001
```

4.5 Defining Home Directories in DFS

Defining the home directory for DCE users is now easy with the integrated login feature provided in AIX/DCE 2.1. You can log in from anywhere into any system supporting DFS using the Common Desktop Environment (CDE) screen on the console or a remote login (telnet, rlogin). When authenticated, you will automatically reach your home directory. We suppose here that the system into which you are logging implements the integrated login. Then, defining a DFS home directory is straightforward. For more information about the integrated login see 7.4, “Integrated Login AIX and DCE” on page 265.

Let's assume that user *brice* is to have a home directory in DFS, namely */:/home/brice*. User brice may or may not have a local account in the system into which he is logging. The most important thing is that user brice has to be registered in the DCE registry database. Below are the steps to follow to add an account with the home directory in the DFS space. For more information about configuring DFS, see 4.2, "Configuring a DFS Server" on page 76.

4.5.1 Defining the User in DCE

Creating a DCE user account with a DFS home directory must be performed by the cell_admin user from any system within the cell.

1. Log in as cell_admin.
2. Create a DCE user account:

```
dcecp> user create brice -uid 900 -gr none -org none -home /:/home/brice\
> -passw <brice_passwd> -mypwd <cell_admin_passwd>
```

3. Check the user profile:

```
dcecp> user show brice
{fullname {}}
{uid 900}
{uuid 00000384-6d39-21cf-9700-10005aa8cff8}
{alias no}
{quota unlimited}
{groups none}
{acctvalid yes}
{client yes}
{created /.../itsc.austin.ibm.com/cell_admin 1996-02-22-10:52:14.000-06:00I-----}
{description {}}
{dupkey no}
{expdate none}
{forwardabletkt yes}
{goodsince 1996-02-22-10:52:13.000-06:00I-----}
{group none}
{home /.../itsc.austin.ibm.com/fs/home/brice}
{lastchange /.../itsc.austin.ibm.com/cell_admin 1996-02-22-10:52:14.000-06:00I-----}
{organization none}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid yes}
{renewabletkt yes}
{server yes}
{shell {}}
{stdtgtauth yes}
No policy
dcecp>
```

4. Design the DFS user space.

   Before we can create the home directory for brice, we must decide about how we will manage our DFS namespace. As you remember, we mentioned that the root.dfs fileset was purposely kept small because it would only contain a set of directories and no files. The home directory can reside here, and we use the `mkdir` command to create it. Your current working directory must be the read/write file tree.

```
# cd /:/.rw
# mkdir home
```

5. Create the necessary filesets.

We feel it is best for the user data to reside in a separate aggregate on its own logical volume. In addition, each user can be put into a unique fileset. This is just one way to manage users. Another way would be to place groups of users in a single fileset, but then it would not be possible to control space on an individual user basis this way. Possibly a combination of the two approaches is ultimately the best, consisting of a series of user filesets within an aggregate representing an administrative unit, such as a department or division.

```
# mklv -t lfs -y dfsu01 rootvg 2
# newaggr -aggregate /dev/dfsu01 -bl 8192 -fr 1024 -overwrite
# mkdfslfs -d /dev/dfsu01 -m /:/.rw/home/brice -n dfsu01 -f brice.ft
```

6. Back out in case errors have occurred.

   It is understandable that something may go wrong when the mkdfslfs command is running. It performs many steps. The following commands are needed to delete a fileset and logical volume after the above steps have completed. These commands will not remove every trace, they are considered the minimum that is needed. The rmdfslfs command can also be used by those who like to perform many steps in a single command to back out from an error.

   ```
   # fts delmount -dir /:/.rw/home/brice
   # fts delete -fileset brice.ft -server ev1 -aggregate dfsu01
   # vi /opt/dcelocal/var/dfs/dfstab (remove entry for aggregate)
   # dfsexport -aggregate dfsu01 -detach
   # rmlv dfsu01
   ```

7. To check, list the mount point and also the aggregates:

   ```
   # fts lsmount -dir /:/.rw/home/brice
   # fts lsfldb
   ```

8. Release the read/write fileset and update the cache:

   ```
   # fts release -fileset root.dfs
   # cm checkfileset
   ```

9. Create the ACL entry for user brice:

   ```
   dcecp> acl show /:/home/brice
   {user_obj rwxcid}
   {group_obj -----}
   {other_obj -----}
   dcecp> acl modify /:/home/brice -add {user brice rwxcid}
   dcecp> acl modify /:/home/brice -change {mask_obj rwxcid}
   ```

   These settings will allow user brice to define the ACLs he wishes to set. If he does not want to mess with any ACL definitions, which is probably typical for most of the users, the umask will be in effect. In this case, it is important for *cell_admin* to set at least the permissions that correspond to the umask on the top directory. Otherwise, no user will ever get access to anything underneath that top directory. If, for instance, Brice's umask is 022, set the permissions to 755:

   ```
   # chmod 755 /:/home/brice
   ```

10. Try to create a file with user cell_admin:

```
# cd /:/home/brice
# touch afile
touch: 0652-046 Cannot create afile.
# pwd
/:/home/brice

# klist
DCE Identity Information:
        Warning: Identity information is not certified
        Global Principal: /.../itsc.austin.ibm.com/cell_admin
        Cell:      f2d03530-6708-11cf-9367-10005aa8cff8 /.../itsc.austin.ibm.com
        Principal: 00000064-6708-21cf-9300-10005aa8cff8 cell_admin
        Group:     0000000c-6708-21cf-9301-10005aa8cff8 none
        Local Groups:
                0000000c-6708-21cf-9301-10005aa8cff8 none
                00000064-6709-21cf-8301-10005aa8cff8 acct-admin
                00000065-6709-21cf-8301-10005aa8cff8 subsys/dce/sec-admin
                00000066-6709-21cf-8301-10005aa8cff8 subsys/dce/cds-admin
                00000068-6709-21cf-8301-10005aa8cff8 subsys/dce/dts-admin
               00000067-6709-21cf-8301-10005aa8cff8 subsys/dce/dfs-admin
.......
# mkdir /:/home/brice/dir2
mkdir: 0653-357 Cannot access directory ..
.: The file access permissions do not allow the specified action.
```

You can notice here that even cell_admin cannot create a file or a directory on /:/home/brice once the ACL is set. This is different from UNIX, where the *root* account can do everything, whereas here in the DFS space, users, even *cell_admin*, need the appropriate ACLs entries.

The task of the cell_admin user stops here. It is now up to user brice to protect his own directory by setting appropriate ACL entries.

4.5.2 Tasks of the Local Administrator

The local administrator needs to enable the integrated login in the local system, which means defining DCE as an authentication method to the local AIX system. He or she must be careful about allowing DCE users to access the machine. For more information, see 7.4.2, “User Synchronization Between AIX 4.1+ and DCE” on page 267.

4.5.3 User’s Tasks

The user needs to set appropriate ACL entries for his or her directory, subdirectories or files.

1. Log into AIX and DCE at the same time:

```
AIX Version 4
(C) Copyrights by IBM and by others 1982, 1994.
login: brice
brice's Password:
*******************************************************************************
*                                                                             *
*  Welcome to AIX Version 4.1!                                                *
*                                                                             *
*******************************************************************************

...
```

```
$ pwd
/.../itsc.austin.ibm.com/fs/home/brice
$ id
uid=900(brice) gid=12(none)
$ touch myfile
$ ls -l
total 0
-rw-r--r--   1 brice    none           0 Feb 22 11:37 myfile

$ dcecp -c acl show .
{mask_obj rwxcid}
{user_obj rwxcid}
{user brice rwxcid}
{group_obj r-x---}
{other_obj r-x---}
```

2. It is not necessary to set any initial creation ACLs because for everything Brice creates, he will be the owner and obtain the *user_obj* permissions. He can set ACLs for other users, though.

   Add an entry for the home directory for a user, *friend*:

   ```
   $ dcecp -c acl modify /:/home/brice -add {user friend rwxcid}
   ```

   ACLs for further directories:

   ```
   $ dcecp -c acl modify -ic /:/home/brice -add {user friend rwxcid}
   $ dcecp -c acl modify -ic /:/home/brice -change {mask_obj rwxid}
   $ dcecp -c acl show -ic /:/home/brice
   {mask_obj rwx-id}
   {user_obj rwxcid}
   {user friend rwxcid effective rwx-id}
   {group_obj rwx-id}
   {other_obj rwx-id}
   ```

   Set ACL for further files:

   ```
   $ dcecp -c acl modify -io /:/home/brice -add {user friend rwxc}
   $ dcecp -c acl modify -io /:/home/brice -change {mask_obj rwx}
   $ dcecp -c acl show -io /:/home/brice
   {mask_obj rwx---}
   {user_obj rwxc--}
   {user friend rwxc-- effective rwx---}
   {group_obj rwx---}
   {other_obj rwx---}
   ```

   Note: If Brice creates a directory or file in his home directory, its *mask_obj* might be restricted by the umask setting. If, for instance, umask is 022, then the UNIX group permission will be r-x, which will create a *mask_obj* that has no write permission, which in turn will restrict *friend's* access.

The task of the user is basically done now. Try to log into a system again:

```
$exit


AIX Version 4
(C) Copyrights by IBM and by others 1982, 1994.
login: brice
brice's Password:
*******************************************************************************
*                                                                             *
*  Welcome to AIX Version 4.1!                                                *
```

```
*                                                                      *
************************************************************************
...
$ pwd
/.../itsc.austin.ibm.com/fs/home/brice
$ umask
022
$ ls -l
drwxr-xr-x   2 brice    none         256 Feb 22 11:57 dir1
-rw-r--r--   1 brice    none           0 Feb 22 11:37 myfile
```

Now everything should work correctly. If you log in from the Common Desktop Environment (CDE), the system automatically creates profiles for you.

```
$ls -la
total 18
drwxrwxr-x   4 brice    none         480 Feb 22 12:06 .
drwxrwxr-x   3 root     none         288 Feb 22 11:05 ..
-rw-------   1 brice    none          97 Feb 22 12:06 .Xauthority
drwxr-xr-x  10 brice    none         544 Feb 22 12:06 .dt
-rwxr-xr-x   1 brice    none        3970 Feb 22 12:06 .dtprofile
-rw-------   1 brice    none          78 Feb 22 12:07 .sh_history
drwxr-xr-x   2 brice    none         256 Feb 22 11:57 dir1
-rw-r--r--   1 brice    none           0 Feb 22 11:37 myfile
```

4.6 Summary

We achieved our objective as depicted in Figure 36 on page 75 in the introduction to this chapter. DFS clients have been set up on the OS/2 Warp machines. The components configured for DFS can be determined in the following way on AIX.

```
ev1::/-> lsdfs
Current state of DFS configuration:
dfs_cl       COMPLETE   DFS Client Machine
dfs_fldb     COMPLETE   DFS Fileset Database Machine
dfs_repsrv   COMPLETE   DFS Replicated Fileset Server Machine
dfs_scm      COMPLETE   DFS System Control Machine
dfs_srv      COMPLETE   DFS File Server Machine
```

On OS/2 Warp, only clients can be configured. A server cat command issued locally or remotely shows us the presence of the dfsd daemon in the *srvrconf.db* maintained by the DCE daemon on every machine:

```
dcecp>  server cat /.:/hosts/EV5
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/cdsadv
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/dtsd
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/time_provider
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/dfsd
```

Locally, the command could look like:

```
DCECP> server cat
```

In the table below we indicate the daemons/processes which have to be activated on machines with respect to the roles that they have to fulfill.

Machine Role	Function Performed	Processes	Remarks
System Control Machine	Distribute Admin Privileges	bosserver upserver (1) upclient (2)	Mgmt Command Server to Fileset/FLDB Server from Binary Distribution Machine (BDM)
FLDB Server	Store Fileset Location	bosserver flserver upclient (1) upclient (2)	Mgmt Command Server FLDB Server Process from SCM from BDM
Fileset Server	Store/export Filesets	bosserver ftserver fxd dfsbind repserver upclient (1) upclient (2)	Monitoring and Mgmt Fileset Management File Exporter Authentication Replica Management from SCM from BDM
DFS Client	Access DFS Files	dfsd dfsbind	Data Access, Caching Path Resolution

Figure 41. List of Processes and Daemons for DFS Machine Roles

Chapter 5. Implementing Various LAN/WAN Scenarios

To gain experience with different topologies, cell layouts, and administration issues, we decided to implement different scenarios. Our approach was to create scenarios based on different technical aspects rather than to try to describe examples of business areas and possible solutions for them. We figured that many different businesses would end up with the same cell layout based on abstract factors such as:

- Amount of central data or service access
- Need for interchange of data and services between branch offices, subsidiaries, or even other companies
- Size of branch offices
- Availability requirements
- Interoperability with other systems, clusters, and so on
- Growth expectations
- Cost

It is much easier for customers to decide what abstract technical solution fits their business best when they know what they should pay attention to. The scenarios we were looking at can be divided into the following groups:

1. Local (LAN-type) scenarios
2. LAN/WAN scenarios
3. Intercell setup

We look at different network topologies and vary the placement of the different core services. We provide step-by-step implementation instructions for selected scenarios to enable the reader to do a quick installation of each scenario. Besides configuration instructions, we also document our experiences with the different environments and discuss performance and availability issues.

For detailed installation and configuration instructions see:

- For DCE — Chapter 3, “Implementing DCE Cells” on page 37
- For DFS — Chapter 4, “Implementing DFS” on page 75

Note: Whenever we create a new fileset, we set the permissions for its parent directory to 777. This is not the recommended setup for mount-point directories. We do this for the sake of simplicity. See 4.2.7, “Fixing Access Permissions (ACLs)” on page 84, for a discussion on ACL settings.

5.1 Local (LAN-type) Cells

This section discusses cell topologies without slow communication links. They can be considered to be a local environment. The simplest case is a single LAN with all nodes attached to it. Also, a much more complex topology with bridges and routers in between multiple LANs can logically be thought of as a LAN. These LANs can be directly connected by a router or a bridge, or they can be part of Metropolitan Area Network (MAN) with fast connections, such as:

- Fiber Distributed Data Interface (FDDI)

- Asynchronous Transfer Mode (ATM)
- Fiber Channel Standard (FCS)

An example can be a university campus with an FDDI backbone dropping off several Ethernet LANs in each different building or even on different floors in all the buildings.

This section concentrates on logically pure LAN topology. If there are bridges and routers involved, we assume that the router network provides a fast and reliable environment to be logically considered as a single LAN. We describe step by step how to configure a DCE cell with the following different layout of the core services:

- All master services on one server and replicated servers on another
- Master and replica servers on different nodes

We provide all commands to create each specific scenario so that there is a complete guideline that can be followed. For more explanations, sample SMIT screens, and command output, see Chapter 3, “Implementing DCE Cells” on page 37.

5.1.1 Scenario 2: Master Servers on One Machine and Replicas on Another

As shown in Figure 42, we configure all master servers on *ev1* and all replication servers on *ev4*.

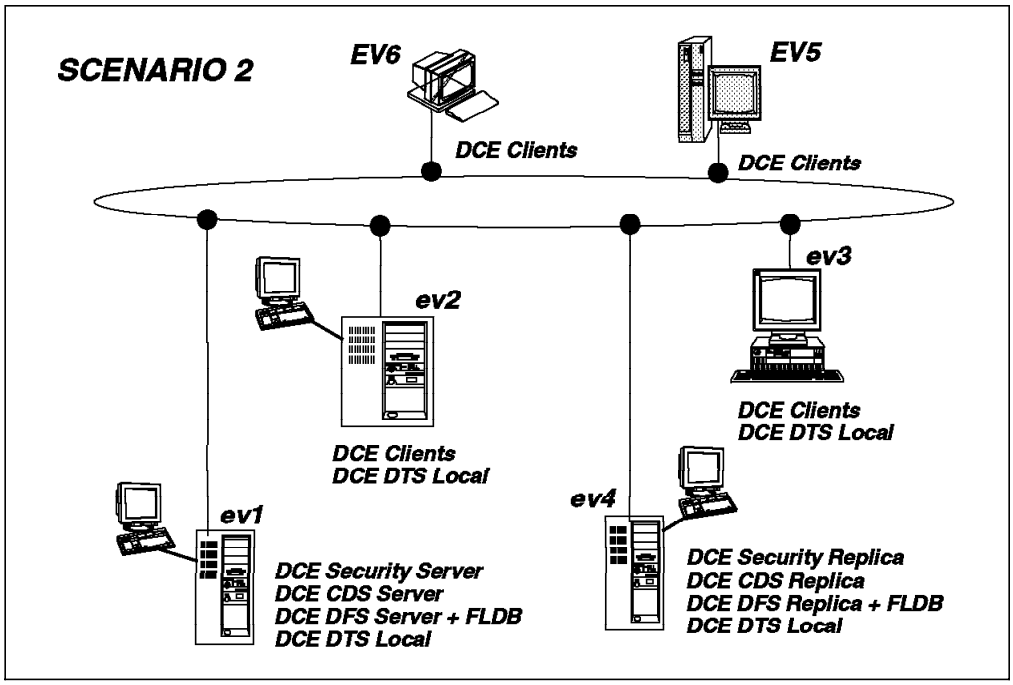


Figure 42. Scenario 2: One Master Server - One Replica Server

5.1.1.1 Preparation Steps

Before you configure any of the DCE machines, you should have:

- Created the necessary file systems
- Checked network name resolution
- Checked network routing
- Checked the network interfaces

- Synchronized the system clocks
- Installed DCE (last of these steps)

For details, see 3.2, “Preparing for DCE Configuration on AIX” on page 38.

5.1.1.2 DCE Configuration Steps

Following are all the configuration steps for this scenario.

Configuring machine ev1

1. Configure the core components:

   ```
   # mkdce -n itsc.austin.ibm.com sec_srv cds_srv dts_local
   ```

   Test a few commands to see if DCE is working correctly:

   ```
   # dce_login cell_admin cell_password
   # dcecp
   dcecp> cell show
   dcecp> directory list /.:
   dcecp> principal cat
   dcecp> quit
   ```

2. Configure the DFS components:

   a. Configure the system control machine (SCM), DFS fileset database (FLDB), DFS server, and DFS client all in one step. The -e flag loads the DFS kernel extension for now and for subsequent restarts:

      ```
      # mkdfs -e dfs_scm dfs_fldb dfs_srv dfs_cl
      ```

   b. Create an aggregate for the root.dfs fileset:

      ```
      # mklv -t lfs -y lfsroot rootvg 1
      # newaggr -aggreg /dev/lfsroot -bl 8192 -fr 1024 -overwrite
      ```

   c. Export the root.dfs fileset:

      ```
      # mkdfslfs -r -d /dev/lfsroot -n lfsroot
      ```

   d. Log in as cell_admin:

      ```
      # dce_login cell_admin cell_password
      ```

   e. Give cell_admin the permission to access the DFS filespace:

      ```
      # chmod 777 /:
      ```

   f. Try to access the DFS filespace:

      ```
      # cd /:
      ```

      For the first access, you normally have to wait a minute. If you are not successful, try again after one minute. The DFS server always goes into TSR mode (Token Status Recovery) even though there has not been any data access by any client.

   g. Replicate the root.dfs fileset:

      Before we can define a replicated fileset, replication should first be done on the primary file server machine. We use the release replication just to show how to replicate a fileset. If you want more information about replicating filesets, see sections 7.2, “DFS Replication” on page 252 and 4.4, “Replicating Filesets on AIX” on page 89.

      1) Configure the fileset replication server:

         ```
         # mkdfs dfs_repsrv
         ```

2) Create read/write mount point for root.dfs:

```
# fts crmount /:/.rw root.dfs -rw
```

3) Define the replication type for root.dfs:

```
# fts setrepinfo -fileset root.dfs -rel
```

4) Define the same machine as a replication site:

```
# fts addsite -fileset root.dfs -server /.:/hosts/ev1 -aggr lfsroot
```

5) Create the read-only fileset and force replication from the read/write source:

```
# fts release -fileset root.dfs
```

6) Leave the DFS root directory; otherwise you are still connected to the read/write fileset of the /: directory:

```
# cd
```

7) Force the local cache manager to refresh its knowledge about the fileset configuration:

```
# cm checkfilesets
```

8) Check whether you can create a file in /: now:

```
# cd /:
# touch testfile
touch: 0652-046 Cannot create testfile.
```

9) You can create the *testfile* only via the read/write mount point:

```
# cd /:/.rw
# touch testfile
# ls
```

h. Create another fileset:

- Create a logical volume /dev/usrbin with five blocks of 4 MB:

```
# mklv -t lfs -y usrbin rootvg 5
```

- Create an aggregate on the /dev/usrbin:

```
# newaggr -aggreg /dev/usrbin -bl 8192 -fr 1024 -overwrite
```

- Export the aggregate:

```
# mkdfslfs -d /dev/usrbin -n usrbin
```

- Create a fileset without a mount point:

```
# mkdfslfs -f usrbin.ft -n usrbin
```

- See if the fileset is correctly exported:

```
# fts lsfldb
```

i. Replicate this fileset before you create the mount point:

1) Define the replication type for usrbin.ft:

```
# fts setrepinfo -fileset usrbin.ft -rel
```

2) Define the same machine as a replication site:

```
# fts addsite -fileset usrbin.ft -server /.:/hosts/ev1 -aggr usrbin
```

3) Create the read-only fileset and force replication from the read/write source:

```
# fts release -fileset usrbin.ft
```

j. Mount the fileset and test access to it:

1) Create the regular mount point, /:/usrbin, which becomes the read-only access path. Since /: is read-only, you must do it as follows:

```
# fts crmount /:/.rw/usrbin usrbin.ft
# chmod 777 /:/.rw/usrbin
```

2) Update the read-only copy of root.dfs to make the directory /:/usrbin available:

```
# fts rel root.dfs
```

3) Force the local cache manager to read the new fileset information:

```
# cm checkfilesets
```

You will not be able to create files in /:/usrbin because this path accesses the read-only fileset. You can access the read/write fileset via /:/.rw/usrbin, or you can create a read/write mount point /:/.usrbin if you do not plan to keep /:/.rw available for daily use.

To create the read/write mount point, issue:

```
# fts crmount /:/.rw/.usrbin usrbin.ft -rw
```

Configuring machines ev2, ev3

1. Configure the core components for ev2:

```
# mkdce -n itsc.austin.ibm.com -s ev1 sec_cl cds_cl dts_local
```

Test a few commands to see if DCE is working correctly:

```
# dce_login cell_admin cell_password
# dcecp
dcecp> cell show
dcecp> directory list /.:
dcecp> principal cat
dcecp> quit
```

2. Configure the DFS components for ev2:

```
# mkdfs dfs_cl
```

Test a few commands to see if DFS is working correctly:

```
# fts lsfldb
# cd /:
# ls -al
```

3. If this DFS client had access to /: before the fileset usrbin.ft was created, you would have to force the local cache manager to read the new fileset information:

```
# cm checkfilesets
```

4. Repeat above steps for ev3:

Configuring machine ev4

1. Configure the core components:

```
# mkdce -n itsc.austin.ibm.com -s ev1 sec_cl cds_cl dts_local
```

2. Configure the CDS replication server:

```
# mkdce cds_second
```

See 3.7.1, “Replicating a CDS Server” on page 65 for more details about CDS replication.

3. Configure the security replication server:

   ```
   # mkdce -R -r ev4 sec_srv
   ```

4. Configure the DFS components:

   a. Force a bind to the master security server:

      ```
      export BIND_PE_SITE=1
      ```

      This avoids problems that we have encountered when mkdfs is bound to the slave security server.

   b. Configure the DFS client:

      ```
      # mkdfs dfs_cl
      ```

   c. Configure the fileset database (FLDB):

      ```
      # mkdfs -s /.:/hosts/ev1 dfs_fldb
      ```

   d. Configure the DFS file server with the option to load the kernel extension:

      ```
      # mkdfs -s /.:/hosts/ev1 -e dfs_srv
      ```

   e. Configure the DFS replication server machine:

      ```
      # mkdfs -s /.:/hosts/ev1 dfs_repsrv
      ```

   f. Release the forced connection to the master security server:

      ```
      # unset BIND_PE_SITE
      ```

   g. Create logical volumes as large as on *ev1*:

      ```
      # mklv -t lfs -y lfsroot rootvg 1
      # mklv -t lfs -y usrbin rootvg 5
      ```

   h. Create the aggregates:

      ```
      # newaggr -aggreg /dev/lfsroot -bl 8192 -fr 1024 -overwrite
      # newaggr -aggreg /dev/usrbin -bl 8192 -fr 1024 -overwrite
      ```

   i. Export the aggregates:

      ```
      # mkdfslfs -d /dev/lfsroot -n lfsroot
      # mkdfslfs -d /dev/usrbin -n usrbin
      ```

   j. Define the new replication site:

      ```
      # fts addsite -fileset root.dfs -server /.:/hosts/ev4 -aggr lfsroot
      # fts addsite -fileset usrbin.ft -server /.:/hosts/ev4 -aggr usrbin
      ```

   k. Create the read-only filesets and force replication from the read/write sources:

      ```
      # fts release -fileset root.dfs
      # fts release -fileset usrbin.ft
      ```

   l. If this DFS client had access to /: before the fileset usrbin.ft was created, you would have to force the local cache manager to read the new fileset information:

      ```
      # cm checkfilesets
      ```

5.1.1.3 Scenario Experiences

Note that being successful configuring a secondary CDS server does not mean that any directories and names are replicated. You have to manually replicate each selected directory to machine *ev4*. Suppose we have a directory called /.:/branch1 and we want to replicate it on machine *ev4*. Here is how you do it:

```
# dce_login cell_admin cell_password
# cdsli -rd | grep branch1 | xargs -i -t cdscp create replica {}  \
clearinghouse /.:/ev4_ch
# cdsli -rd | grep branch1 | xargs -i -t cdscp set dir {} to new \
epoch master /.:/ev1_ch readonly /.:/ev4_ch
```

The commands recursively copy all directories underneath the directory /.:/branch, too.

There is a shell script `copy_ch` provided on the diskette with this publication that copies all directories to a new clearinghouse.

5.1.1.4 Special Issues

The DFS recommendation is to have an odd number (preferably three) FLDBs to ease their voting process for a master FLDB. If we had more file server nodes, we would have to install one more FLDB server as well. Most environments do not need more than three FLDB servers; the voting process would be more complicated and would create more network traffic.

There is not a right or a wrong sequence of steps to set up fileset replication. We replicate a fileset immediately after having created it and before we create the mount point. By doing so, we prevent any DFS clients from accessing the read/write fileset via the path name consisting of regular mount points. But if you do not know yet whether you will ever replicate a fileset later on, it is perfectly right to first create all filesets and mount points, populate the file space, and then possibly replicate some of the filesets. See 4.4, “Replicating Filesets on AIX” on page 89 and 7.2, “DFS Replication” on page 252 for more information on how to replicate filesets.

5.1.1.5 Response Times

All the DCE/DFS commands we tested in this scenario (`dce_login`, `rgy_edit`, `cdscp`, `rpccp` and others) took less than five seconds when all servers are available.

5.1.1.6 Performance Discussion

Having replicated servers means load balancing for registry, CDS, FLDB, and DFS fileset access, provided that the right CDS directories are replicated and the DFS data access to replicated filesets is read-mostly.

Depending on the problems that may be experienced, the following improvements are possible:

- Removing other services/applications from the DCE server machines
- Creating more replicated servers of all types to spread load
- If frequent write access to CDS and/or DFS files occur, consider distributing CDS master replicas and/or DFS read/write filesets to different nodes close to where they are accessed

5.1.1.7 Availability Discussion

In this scenario the entire registry database is replicated, which makes sure that tickets can be issued as long as one of the security servers is reachable. Changes to the registry, such as adding a new principal, might be temporarily impossible, if *ev1* is unavailable.

The FLDB is replicated, which improves availability in the case of a network partition or if one of the servers becomes unavailable.

In order to get highly available read access to the fileset usrbin.ft.readonly, all filesets containing the mount point and its parent directories, all the way up to the root directory (/:), should be replicated, too. This means root.dfs must be replicated on *ev4* to make sure /:/usrbin can be accessed if *ev1* breaks.

Note that for CDS, you must explicitly replicate each directory you want to make highly available. The same is true for DFS filesets. For both CDS and DFS files, high availability is only assured for read access. Write access always goes to the master copy, which might become temporarily unavailable.

The use of the IBM AIX High-Availability Cluster Multi-Processing (HACMP) product should be considered for all cases in which even temporary unavailability to read/write databases is unacceptable. Having the CDS and security master servers for the core services, or a DFS fileset server for DFS, in an HACMP cluster improves availability of write access to these services.

5.1.2 Scenario 3: Master Servers and Replicas on Different Machines

As shown in Figure 43, we spread all master and replication servers over different machines.

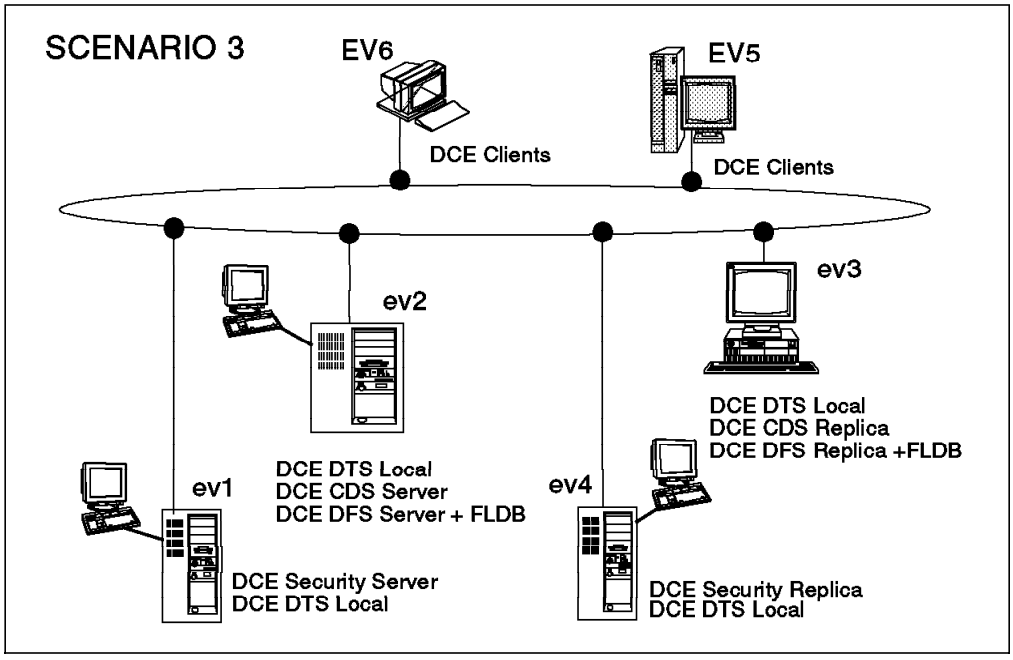


Figure 43. Scenario 3: DCE Servers on Different Machines

5.1.2.1 Preparation Steps

Before you configure any of the DCE machines, you should have:

- Created the necessary file systems
- Checked network name resolution
- Checked network routing
- Checked the network interfaces
- Synchronized the system clocks
- Installed DCE (last of these steps)

For details, see 3.2, “Preparing for DCE Configuration on AIX” on page 38.

5.1.2.2 DCE Configuration Steps

Following are all the configuration steps for this scenario.

Configuring machine ev1

1. Configure the core components:

   a. Configure the security server machine:

      ```
      # mkdce -n itsc.austin.ibm.com sec_srv
      ```

      Note that you have to configure the core services on machine *ev2* now, before you continue with the next steps.

   b. Configure the DTS server and other DCE core clients:

      ```
      # mkdce cds_cl dts_local
      ```

   Test a few commands to see if DCE is working correctly:

   ```
   # dce_login cell_admin cell_password
   # dcecp
   dcecp> cell show
   dcecp> directory list /.:
   dcecp> principal cat
   dcecp> quit
   ```

2. Configure the DFS client machine:

   ```
   # mkdfs dfs_cl
   ```

Configuring machine ev2

1. Configure the core components:

   ```
   # mkdce -n itsc.austin.ibm.com -s ev1 cds_srv sec_cl dts_local
   ```

2. Configure the DFS components:

   a. Configure the system control machine (SCM):

      ```
      # mkdfs dfs_scm
      ```

   b. Configure the DFS fileset database:

      ```
      # mkdfs dfs_fldb
      ```

   c. Configure the DFS file server with the option to load the kernel extension:

      ```
      # mkdfs -e dfs_srv
      ```

   d. Configure the DFS client:

      ```
      # mkdfs dfs_cl
      ```

   e. Create an aggregate for the root.dfs fileset:

```
# mklv -t lfs -y lfsroot rootvg 1
# newaggr -aggreg /dev/lfsroot -bl 8192 -fr 1024 -overwrite
```

f. Export the root.dfs fileset:

```
# mkdfslfs -r -d /dev/lfsroot -n lfsroot
```

g. Log in as cell_admin:

```
# dce_login cell_admin cell_password
```

h. Give cell_admin the permission to access the DFS filespace:

```
# chmod 777 /:
```

i. Try to access the DFS filespace:

```
# cd /:
```

For the first access, you normally have to wait a minute. If you are not successful, try again after one minute. The DFS server always goes into TSR mode (Token Status Recovery) even though there has not been any data access by any client.

j. Replicate the root.dfs fileset:

Before we can define a replicated fileset, replication should first be done on the primary file server machine. We use the release replication just to show how to replicate a fileset. If you want more information about replicating filesets, see sections 7.2, “DFS Replication” on page 252 and 4.4, “Replicating Filesets on AIX” on page 89.

1) Configure the fileset replication server:

```
# mkdfs dfs_repsrv
```

2) Create the read/write mount point for root.dfs:

```
# fts crmount /:/.rw root.dfs -rw
```

3) Define the replication type for root.dfs:

```
# fts setrepinfo -fileset root.dfs -rel
```

4) Define the same machine as a replication site:

```
# fts addsite -fileset root.dfs -server /.:/hosts/ev2 -aggr lfsroot
```

5) Create the read-only fileset, and force replication from the read/write source:

```
# fts release -fileset root.dfs
```

6) Leave the DFS root directory; otherwise you are still connected to the read/write fileset of the /: directory:

```
# cd
```

7) Force the local cache manager to refresh its knowledge about the fileset configuration:

```
# cm checkfilesets
```

8) Check whether you can create a file in /: now:

```
# cd /:
# touch testfile
touch: 0652-046 Cannot create testfile.
```

9) You can create the *testfile* only via the read/write mount point:

```
# cd /:/.rw
# touch testfile
# ls
```

k. Create another fileset:

- Create a logical volume /dev/usrbin with five blocks of 4 MB:

  ```
  # mklv -t lfs -y usrbin rootvg 5
  ```

- Create an aggregate on the /dev/usrbin:

  ```
  # newaggr -aggreg /dev/usrbin -bl 8192 -fr 1024 -overwrite
  ```

- Export the aggregate:

  ```
  # mkdfslfs -d /dev/usrbin -n usrbin
  ```

- Create a fileset without a mount point:

  ```
  # mkdfslfs -f usrbin.ft -n usrbin
  ```

- See if the fileset is correctly exported:

  ```
  # fts lsfldb
  ```

l. Replicate this fileset before you create the mount point:

1) Define the replication type for usrbin.ft:

   ```
   # fts setrepinfo -fileset usrbin.ft -rel
   ```

2) Define the same machine as a replication site:

   ```
   # fts addsite -fileset usrbin.ft -server /.:/hosts/ev2 -aggr usrbin
   ```

3) Create the read-only fileset, and force replication from the read/write source:

   ```
   # fts release -fileset usrbin.ft
   ```

m. Mount the fileset, and test access to it:

1) Create the regular mount point, /:/usrbin, which becomes the read-only access path. Since /: is read-only, you must do it as follows:

   ```
   # fts crmount /:/.rw/usrbin usrbin.ft
   # chmod 777 /:/.rw/usrbin
   ```

2) Update the read-only copy of root.dfs to make the directory /:/usrbin available:

   ```
   # fts rel root.dfs
   ```

3) Force the local cache manager to read the new fileset information:

   ```
   # cm checkfilesets
   ```

You will not be able to create files in /:/usrbin because this path accesses the read-only fileset. You can access the read/write fileset via /:/.rw/usrbin, or you can create a read/write mount point /:/.usrbin if you do not plan to keep /:/.rw available for daily use.

To create the read/write mount point, issue:

```
# fts crmount /:/.rw/.usrbin usrbin.ft -rw
```

Configuring machine ev3

1. Configure the DTS server and DCE core clients:

```
# mkdce -n itsc.austin.ibm.com -s ev1 sec_cl cds_cl dts_local
```

2. Configure the CDS replication server:

   ```
   # mkdce cds_second
   ```

   See 3.7.1, “Replicating a CDS Server” on page 65 for more details about CDS replication.

3. Configure the DFS components:

   a. Configure the DFS client:

      ```
      # mkdfs dfs_cl
      ```

   b. Configure the DFS fileset database (FLDB):

      ```
      # mkdfs -s /.:/hosts/ev2 dfs_fldb
      ```

   c. Configure the DFS file server machine with the option to start the kernel extension:

      ```
      # mkdfs -s /.:/hosts/ev2 -e dfs_srv
      ```

   d. Configure the DFS replication server machine:

      ```
      # mkdfs -s /.:/hosts/ev2 dfs_repsrv
      ```

   e. Create logical volumes as large as on *ev2*:

      ```
      # mklv -t lfs -y lfsroot rootvg 1
      # mklv -t lfs -y usrbin rootvg 5
      ```

   f. Create the aggregates:

      ```
      # newaggr -aggreg /dev/lfsroot -bl 8192 -fr 1024 -overwrite
      # newaggr -aggreg /dev/usrbin -bl 8192 -fr 1024 -overwrite
      ```

   g. Export the aggregates:

      ```
      # mkdfslfs -d /dev/lfsroot -n lfsroot
      # mkdfslfs -d /dev/usrbin -n usrbin
      ```

   h. Define the new replication site:

      ```
      # fts addsite -fileset root.dfs -server /.:/hosts/ev3 -aggr lfsroot
      # fts addsite -fileset usrbin.ft -server /.:/hosts/ev3 -aggr usrbin
      ```

   i. Create the read-only filesets and force replication from the read/write sources:

      ```
      # fts release -fileset root.dfs
      # fts release -fileset usrbin.ft
      ```

   j. If this DFS client had access to /: before the fileset usrbin.ft was created, you would have to force the local cache manager to read the new fileset information:

      ```
      # cm checkfilesets
      ```

Configuring machine ev4

1. Configure the DTS server and DCE core clients:

   ```
   # mkdce -n itsc.austin.ibm.com -s ev1 sec_cl cds_cl dts_local
   ```

2. Configure the security replication server:

   ```
   # mkdce -R -r ev4 sec_srv
   ```

3. Configure the DFS client machine:

   ```
   # mkdfs dfs_cl
   ```

4. If this DFS client had access to /: before the fileset usrbin.ft was created, you would have to force the local cache manager to read the new fileset information:

   ```
   # cm checkfilesets
   ```

5.1.2.3 Scenario Experiences

The same discussion as in scenario 2 applies. Please see 5.1.1.3, “Scenario Experiences” on page 111.

5.1.2.4 Special issues

There are no special issues in this scenario.

5.1.2.5 Performance Discussion

The same discussion as in scenario 2 applies. Please see 5.1.1.6, “Performance Discussion” on page 111.

5.1.2.6 Availability Discussion

The same discussion as in scenario 2 applies. Please see 5.1.1.7, “Availability Discussion” on page 112.

5.2 LAN/WAN Cells

This section discusses cell topologies that involve remote sites connected to a central site via wide area networks (WANs) that use relatively slow communication links. This is probably the most common real-world picture of companies today, where a (usually) big number of subsidiaries or branch offices need access to a (usually) small number of central sites. These remote sites are very different in size. They may range from a single remote workstation to a site with hundreds of workstations.

In our limited test environment, we set up a few scenarios with a central site, which is marked by the token-ring LAN, and one remote site. We looked at two different types of remote sites, a small one that does not run any servers or services and a large one that consists of a few servers and many client workstations. The following is a list of two-site scenarios with different link types that we look at in this section:

- A small branch connected via WAN (scenario 4)
- A large branch connected via WAN (scenario 5)
- A branch with redundant communication links (scenario 6)
- An intercell scenario

Today companies are establishing their vital communication links and networks with routers and bridges which allow for several protocols to be transmitted so the same infrastructure can be used in a heterogeneous environment. They may even implement alternate communication links so that we need not worry about availability of the network. The more sophisticated the network, the less we need to be concerned about the location of the DCE servers and replicas because the entire network resembles one big LAN.

However, there are environments where IP routing is done by regular workstations or servers. This is what we implemented in our scenarios. The most important result we want to get across is that the DCE servers should not export slow WAN interfaces into CDS. Always exclude, for instance, X.25 or SLIP

interfaces by setting the environment variable RPC_UNSUPPORTED_NETIFS. See 3.2.3, “Checking Network Routing” on page 40, for details.

For a summary of our findings and planning hints, see Chapter 2, “Planning DCE Cells” on page 19.

We provide all commands to create each specific scenario; so there is a complete guideline which can be followed. For more explanation, sample SMIT screens, and command output, see Chapter 3, “Implementing DCE Cells” on page 37.

In these scenarios, be careful with the routing. For the sake of simplicity, we used the /etc/hosts file and static routes. Please be aware that you most likely will find domain name servers and routing daemons in a customer environment. It is beyond the scope of this document to explain their setup.

5.2.1 Scenario 4: A Small Branch Connected via WAN (X.25/SLIP)

As shown in Figure 44, we install all servers in the central site. The Ethernet network with *ev3* and *ev4* simulates a small branch.

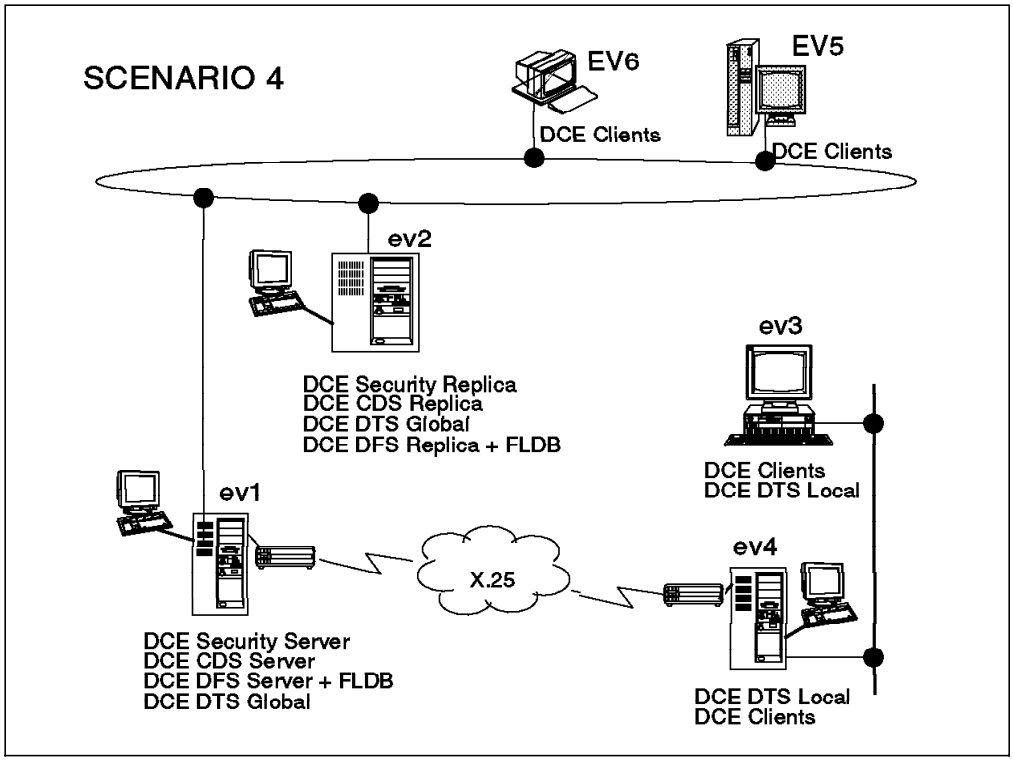


Figure 44. Scenario 4: A Small Branch Connected via 19,200 bps X.25

There are many kinds of communication links for WANs at present. However, in respect to DCE configuration, the underlying communication medium does not matter if a TCP/IP connection is configured on it. We used an X.25 link between two sites as an example of a WAN link. Other communication links, for example SLIP (which we tested in the previous version of this book, GG24-4348), can be used following almost the same steps.

5.2.1.1 Preparation Steps

Before you configure any of the DCE machines, you should have:

- Created the necessary file systems
- Checked network name resolution
- Checked network routing - see below
- Checked the network interfaces
- Synchronized the system clocks
- Installed DCE (last of these steps)

For details, see 3.2, “Preparing for DCE Configuration on AIX” on page 38. To communicate to each other, we have to set the routes on each machine:

On machine ev1: List the network interfaces:

```
# netstat -i
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16896 <Link>                         12316     0    12316     0     0
lo0   16896 127         localhost          12316     0    12316     0     0
tr0   1492  <Link>10.0.5a.4f.46.29         11431     0     5585     0     0
tr0   1492  9.3.1       ev1.itsc.austin    11431     0     5585     0     0
xs0   1500  <Link>                        248151     0   165313     0     0
xs0   1500  192.1.20    ev1x25             248151     0   165313     0     0
```

Check name resolution:

```
# host ev1
ev1 is 9.3.1.68
# host ev1x25
ev1x25 is 192.1.20.3
# host ev4et
ev4et is 193.1.10.4
# host ev4x25
ev4x25 is 192.1.20.2
```

In order for *ev1* to get to the Ethernet network, we have to specify *ev4*’s X.25 interface as the gateway:

```
# route add -net 193.1.10 ev4x25 1
```

Note that AIX 4.1 doesn’t do IP forwarding by default. To make IP routing available, issue:

```
# no -o ipforwarding=1
```

We recommend placing the above command in a file, for example /etc/rc.tcpip, which is called during system boot.

Exclude the X.25 interface now and forever:

```
# export RPC_UNSUPPORTED_NETIFS=xs0
# echo "export RPC_UNSUPPORTED_NETIFS=xs0" >> /etc/environment
```

SLIP

This is the only X.25-specfic part in the configuration process. In case of SLIP, you should issue:

```
# export RPC_UNSUPPORTED_NETIFS=sl0
# echo RPC_UNSUPPORTED_NETIFS=sl0 >> /etc/environment
```

If both X.25 and SLIP exist on one machine, you should issue:

```
# export RPC_UNSUPPORTED_NETIFS=xs0:sl0
# echo RPC_UNSUPPORTED_NETIFS=xs0:sl0 >> /etc/environment
```

On machine ev2: List the network interfaces:

```
# netstat -i
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16896 <Link>                         16153     0    16153     0     0
lo0   16896 127         localhost          16153     0    16153     0     0
tr0   1492  <Link>10.0.5a.a8.cf.f8        399804     0   148114     0     0
tr0   1492  9.3.1       ev2.itsc.austin   399804     0   148114     0     0
```

Set the appropriate route to always go via *ev1*:

```
# route add default ev1 1
```

On machine ev3: List the network interfaces:

```
# netstat -i
Name  Mtu   Network     Address         Ipkts    Ierrs Opkts    Oerrs Coll
lo0   1536  <Link>                         1611     0     1611     0     0
lo0   1536  127         localhost          1611     0     1611     0     0
en0   1500  <Link>                          964     0      942     0     0
en0   1500  193.1.10    ev3et               964     0      942     0     0
```

Check name resolution, and set the appropriate route:

```
# host ev4et
ev4et is 193.1.10.4
# host ev3et
ev3et is 193.1.10.3
# host ev1
ev1 is 9.3.1.68
# host ev1x25
ev1x25 is 192.1.20.3
# route add default ev4et 1
```

On machine ev4: List the network interfaces:

```
# netstat -i
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16896 <Link>                          5407     0     5422     0     0
lo0   16896 127         loopback           5407     0     5422     0     0
en0   1500  <Link>2.60.8c.2f.6.53          3995     0     4445     0     0
en0   1500  193.1.10    ev4et              3995     0     4445     0     0
xs0   1500  <Link>                        304244     0   126259     0     0
xs0   1500  192.1.20    ev4x25           304244     0   126259     0     0
```

Check name resolution, and set the appropriate route:

```
# host ev1
ev1 is 9.3.1.68
# host ev1x25
ev1x25 is 192.1.20.3
# route add -net 9.3.1 ev1x25 1
```

Make IP forwarding available:

```
# no -o ipforwarding=1
```

5.2.1.2 DCE Configuration Steps

Following are all the configuration steps for this scenario.

Configuring machine ev1

1. Configure the core components:

   ```
   # mkdce -n itsc.austin.ibm.com sec_srv cds_srv dts_global
   ```

   Test a few commands to see if DCE is working correctly:

   ```
   # dce_login cell_admin cell_password
   # dcecp
   dcecp> cell show
   dcecp> directory list /.:
   dcecp> principal cat
   dcecp> quit
   ```

2. Configure the DFS components:

   a. Configure the system control machine (SCM), DFS fileset database (FLDB), DFS server, DFS client all in one step: The -e flag loads the DFS kernel extension for now and for subsequent restarts:

      ```
      # mkdfs -e dfs_scm dfs_fldb dfs_srv dfs_cl
      ```

   b. Create an aggregate for the root.dfs fileset:

      ```
      # mklv -t lfs -y lfsroot rootvg 1
      # newaggr -aggreg /dev/lfsroot -bl 8192 -fr 1024 -overwrite
      ```

   c. Export the root.dfs fileset:

      ```
      # mkdfslfs -r -d /dev/lfsroot -n lfsroot
      ```

   d. Log in as cell_admin:

      ```
      # dce_login cell_admin cell_password
      ```

   e. Give cell_admin the permission to access the DFS filespace:

      ```
      # chmod 777 /:
      ```

   f. Try to access the DFS filespace:

      ```
      # cd /:
      ```

      For the first access, you normally have to wait a minute. If you are not successful, try again after one minute. The DFS server always goes into TSR mode (Token Status Recovery) even though there has not been any data access by any client.

   g. Replicate the root.dfs fileset:

      Before we can define a replicated fileset, replication should first be done on the primary file server machine. We use the release replication just to show how to replicate a fileset. If you want more information about

replicating filesets, see sections 7.2, “DFS Replication” on page 252 and 4.4, “Replicating Filesets on AIX” on page 89.

1) Configure the fileset replication server:

```
# mkdfs dfs_repsrv
```

2) Create read/write mount point for root.dfs:

```
# fts crmount /:/.rw root.dfs -rw
```

3) Define the replication type for root.dfs:

```
# fts setrepinfo -fileset root.dfs -rel
```

4) Define the same machine as a replication site:

```
# fts addsite -fileset root.dfs -server /.:/hosts/ev1 -aggr lfsroot
```

5) Create the read-only fileset, and force replication from the read/write source:

```
# fts release -fileset root.dfs
```

6) Leave the DFS root directory; otherwise you are still connected to the read/write fileset of the /: directory:

```
# cd
```

7) Force the local cache manager to refresh its knowledge about the fileset configuration:

```
# cm checkfilesets
```

8) Check whether you can create a file in /: now:

```
# cd /:
# touch testfile
touch: 0652-046 Cannot create testfile.
```

9) You can create the *testfile* only via the read/write mount point:

```
# cd /:/.rw
# touch testfile
# ls
```

h. Create another fileset:

- Create a logical volume /dev/usrbin with five blocks of 4 MB:

```
# mklv -t lfs -y usrbin rootvg 5
```

- Create an aggregate on the /dev/usrbin:

```
# newaggr -aggreg /dev/usrbin -bl 8192 -fr 1024 -overwrite
```

- Export the aggregate:

```
# mkdfslfs -d /dev/usrbin -n usrbin
```

- Create a fileset without a mount point:

```
# mkdfslfs -f usrbin.ft -n usrbin
```

- See if the fileset is correctly exported:

```
# fts lsfldb
```

i. Replicate this fileset before you create the mount point:

1) Define the replication type for usrbin.ft:

```
# fts setrepinfo -fileset usrbin.ft -rel
```

2) Define the same machine as a replication site:

```
# fts addsite -fileset usrbin.ft -server /.:/hosts/ev1 -aggr usrbin
```

3) Create the read-only fileset, and force replication from the read/write source:

```
# fts release -fileset usrbin.ft
```

j. Mount the fileset and test access to it:

1) Create the regular mount point, /:/usrbin, which becomes the read-only access path. Since /: is read-only, you must do it as follows:

```
# fts crmount /:/.rw/usrbin usrbin.ft
# chmod 777 /:/.rw/usrbin
```

2) Update the read-only copy of root.dfs to make the directory /:/usrbin available:

```
# fts rel root.dfs
```

3) Force the local cache manager to read the new fileset information:

```
# cm checkfilesets
```

You will not be able to create files in /:/usrbin because this path accesses the read-only fileset. You can access the read/write fileset via /:/.rw/usrbin, or you can create a read/write mount point, /:/.usrbin if you do not plan to keep /:/.rw available for daily use.

To create the read/write mount point, issue:

```
# fts crmount /:/.rw/.usrbin usrbin.ft -rw
```

Configuring machine ev2

1. Configure the DTS server and DCE core clients:

```
# mkdce -n itsc.austin.ibm.com -s ev1 sec_cl cds_cl dts_global
```

2. Configure the CDS replication server:

```
# mkdce cds_second
```

See 3.7.1, “Replicating a CDS Server” on page 65, for more details about CDS replication.

3. Configure the security replication server:

```
# mkdce -R -r ev2 sec_srv
```

4. Configure the DFS components:

a. Force a bind to the master security server:

```
export BIND_PE_SITE=1
```

This avoids problems that we have encountered when mkdfs is bound to the slave security server.

b. Configure the DFS client:

```
# mkdfs dfs_cl
```

c. Configure the the fileset database (FLDB):

```
# mkdfs -s /.:/hosts/ev1 dfs_fldb
```

d. Configure the DFS file server with the option to load the kernel extension:

```
# mkdfs -s /.:/hosts/ev1 -e dfs_srv
```

e. Configure the DFS replication server machine:

```
# mkdfs -s /.:/hosts/ev1 dfs_repsrv
```

f. Release the forced connection to the master security server:

```
# unset BIND_PE_SITE
```

g. Create logical volumes as large as on *ev1*:

```
# mklv -t lfs -y lfsroot rootvg 1
# mklv -t lfs -y usrbin rootvg 5
```

h. Create the aggregates:

```
# newaggr -aggreg /dev/lfsroot -bl 8192 -fr 1024 -overwrite
# newaggr -aggreg /dev/usrbin -bl 8192 -fr 1024 -overwrite
```

i. Export the aggregates:

```
# mkdfslfs -d /dev/lfsroot -n lfsroot
# mkdfslfs -d /dev/usrbin -n usrbin
```

j. Define the new replication site:

```
# fts addsite -fileset root.dfs -server /.:/hosts/ev2 -aggr lfsroot
# fts addsite -fileset usrbin.ft -server /.:/hosts/ev2 -aggr usrbin
```

k. Create the read-only filesets, and force replication from the read/write sources:

```
# fts release -fileset root.dfs
# fts release -fileset usrbin.ft
```

l. If this DFS client had access to /: before the fileset usrbin.ft was created, you would have to force the local cache manager to read the new fileset information:

```
# cm checkfilesets
```

Configuring machines ev3, ev4

1. Configure the core components for ev3:

   ```
   # mkdce -n itsc.austin.ibm.com -s ev1 -c ev1 sec_cl cds_cl dts_local
   ```

   Test a few commands to see if DCE is working correctly:

   ```
   # dcecp
   dcecp> cell show
   dcecp> directory list /.:
   dcecp> principal cat
   dcecp> quit
   ```

2. Configure the DFS components for ev3:

   ```
   # mkdfs dfs_cl
   ```

   Test a few commands to see if DFS is working correctly:

   ```
   # fts lsfldb
   # cd /:
   # ls -al
   ```

3. If this DFS client had access to /: before the fileset usrbin.ft was created, you would have to force the local cache manager to read the new fileset information:

   ```
   # cm checkfilesets
   ```

4. Repeat above steps for ev4:

5.2.1.3 Scenario Experiences

When we configure the DCE/DFS client machines on the Ethernet side, it takes more time than when we configure them on the token-ring side where the servers are located.

5.2.1.4 Special Issues

The DFS recommendation is to have at least three FLDBs to ease their voting process for a master FLDB. If we had more file server nodes, we would have to install one more FLDB server as well.

The DTS recommendation is to have at least three servers on each LAN segment. We would have to add more local DTS servers if we had more nodes. Make sure that every DTS entity can reach at least the required number of servers. Per default, DTS clerks need one DTS server; DTS servers need three time values from DTS servers, including their own clock value. If there are not enough DTS servers defined, adjust the number of required servers.

We decided that only the central site is responsible for the correct time. By defining global servers on the central site only, we make sure that:

1. The central site's servers adjust their clocks only among themselves.
2. The remote sites synchronize with one global server of the central site and with their own local servers.

Note that if a third local DTS server were added to the Ethernet LAN, *ev4* would have to be a courier to make sure that a global server is contacted. Since there are only two, the DTS local servers on the Ethernet automatically contacts a global server to get three clock values.

Since the CDS servers are on another LAN, we must specify the `-c` flag with the first `mkdce` command.

5.2.1.5 Response Times

On the token-ring side, even though we integrate a WAN, performance is still as good as in scenario 2 and 3. However, on the Ethernet side, response is not as good as on the token-ring side. The reason is that all DCE client commands have to go across the slower WAN link, which is an X.25 line with 19,200 bps in this scenario.

When a WAN is between DCE/DFS clients and servers, access to DCE/DFS always takes more time, but all DCE commands work correctly with an acceptable response time.

5.2.1.6 Performance Discussion

Having replicated servers means load balancing, as we discussed in scenario 2 (see 5.1.1.6, "Performance Discussion" on page 111), as long as we do not replicate servers to the remote sites. The improvements mentioned there are valid only within the central site. Faster communication links would be the first step for better performance at the remote sites.

Further improvements in the branches can be achieved by moving or replicating certain DCE servers and resources to the remote sites. This has to be well designed; otherwise you will experience a lot of unnecessary traffic to servers in the remote sites. This would affect performance of the entire cell. In our scenarios we would consider a remote site with DCE/DFS servers a large

branch. Therefore, see in 5.2.2.6, “Performance Discussion” on page 128 (scenario 5), for a discussion on server replication over WAN.

5.2.1.7 Availability Discussion

The availability discussion as far as the central site is concerned is the same as in scenario 2 (see 5.1.1.7, “Availability Discussion” on page 112).

The remote sites are at a certain risk. If the link becomes unavailable or one of the gateway nodes drop out, DCE is not working for this branch anymore. The solution for improved availability is either replicating the servers (which is very delicate according to the above performance discussion) or redundant layout of the underlying TCP/IP with dynamic routing. See also 5.2.3, “Scenario 6: A Branch Connected with Two Links” on page 130.

5.2.2 Scenario 5: A Large Branch Connected via WAN (X.25/SLIP)

As shown in Figure 45, the Ethernet network with *ev3* and *ev4* simulates a large branch with replication servers this time.

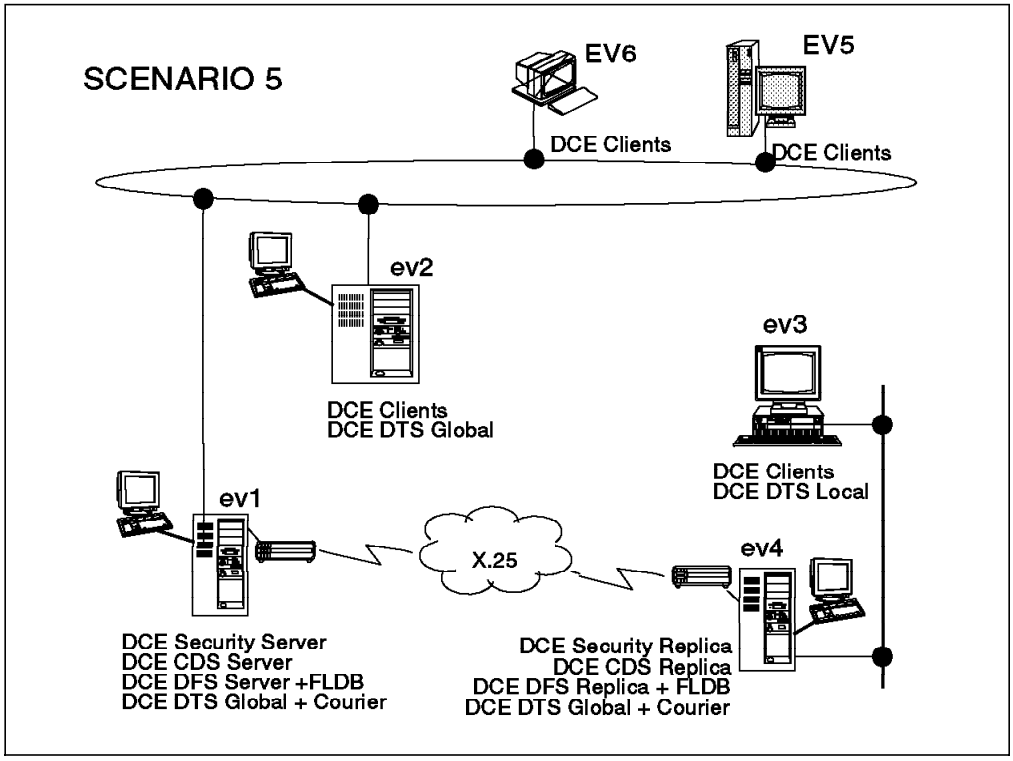


Figure 45. Scenario 5: A Large Branch Connected via X.25

5.2.2.1 Preparation Steps

Before you configure any of the DCE machines, you should have:

- Created the necessary file systems
- Checked network name resolution
- Checked network routing
- Checked the network interfaces
- Synchronized the system clocks
- Installed DCE (last of these steps)

For details, see 3.2, “Preparing for DCE Configuration on AIX” on page 38.

Be careful with the routing. The routes to be set are the same as in scenario 4a. See 5.2.1.1, “Preparation Steps” on page 119.

Exclude the X.25 interface on *ev1* and *ev4* now and forever:

```
# export RPC_UNSUPPORTED_NETIFS=xs0
# echo RPC_UNSUPPORTED_NETIFS=xs0 >> /etc/environment
```

SLIP

This is the only X.25-specfic part in the configuration process. In case of SLIP, you should issue:

```
# export RPC_UNSUPPORTED_NETIFS=sl0
# echo RPC_UNSUPPORTED_NETIFS=sl0 >> /etc/environment
```

If both X.25 and SLIP exist on one machine, you should issue:

```
# export RPC_UNSUPPORTED_NETIFS=xs0:sl0
# echo RPC_UNSUPPORTED_NETIFS=xs0:sl0 >> /etc/environment
```

5.2.2.2 Configuration Steps

To configure all machines in this scenario, you can follow exactly the same steps as in scenario 2, as described in 5.1.1.2, “DCE Configuration Steps” on page 107. The *only exception* is you must replace the very first mkdce command for these systems. The correct commands are shown below:

First mkdce for machine ev1

```
# mkdce -n itsc.austin.ibm.com -t courier sec_srv cds_srv dts_global
```

First mkdce for machine ev2

```
# mkdce -n itsc.austin.ibm.com -s ev1 sec_cl cds_cl dts_global
```

First mkdce for machine ev3

```
# mkdce -n itsc.austin.ibm.com -s ev1 -c ev1 sec_cl cds_cl dts_local
```

First mkdce for machine ev4

```
# mkdce -n itsc.austin.ibm.com -s ev1 -c ev1 -t courier sec_cl cds_cl dts_global
```

5.2.2.3 Scenario Experiences

We can configure all services on machine *ev4* as planned, but it all takes more time to configure. We tested with and without an FLDB server on *ev4*. When FLDB servers were on both sides of the X.25 link, response times in the whole cell became really slow. Response times for all DCE operations went up, and even regular TCP/IP commands over this link became very slow. As soon as the FLDB server was removed from *ev4*, operation went back to normal, and response times were good.

5.2.2.4 Special Issues

Since we assume that the branch is a large one with a sufficient number of local DTS servers, we need a courier-type server to make sure this site’s clocks are synchronized with the central site. The courier server always includes the time of one global DTS server in the calculation for adjustment of its own clock.

Since the central site also has a courier DTS server, it takes into consideration the time values of global DTS servers of remote sites. If you wanted the central site to synchronize the clocks internally and remote sites to adjust to the central site, you would define the DTS server on *ev1* as *noncourier*.

5.2.2.5 Response Times

There is no difference from scenario 4a (5.2.1.5, “Response Times” on page 125) as far as DCE core service access times is concerned when we do not have an FLDB server on *ev4*. Configuring FLDB servers on both sides of the X.25 link generates considerable extra traffic on this (slow) line, and commands sometimes take a long time to complete.

5.2.2.6 Performance Discussion

Replicating servers usually means load balancing. Since the servers are all randomly selected, statistically, half of the read-only access calls go to servers on *ev1* and the other half to *ev4*. That is the case for:

- Getting a ticket from the security server
- Finding a service from CDS
- Finding the location of a fileset from the FLDB
- Read access to the replicated fileset, *usrbin.ft*

This can also be the case with DCE-based products or customer-developed applications that support replicated services, if they rely on RPC group entries in CDS and/or use random selection of binding handles. However, an application developer has all the freedom to implement some sophisticated features for server selection. One can, for instance, analyze the binding handles received from CDS or read an environment variable with a preferred server address and so on. There are many possibilities. In the core components or DFS, there are some built-in optimization features and/or configuration options that we discuss in following sections.

The nice thing about load balancing through random server selection can turn into a major performance penalty in the whole cell when almost half of all these calls have to go across a slow WAN link. Is there anything that can be done to override random selection? We have some configuration options that we describe below for every component. However, before we implement any of these optimizations, which certainly do not make administration of a cell any easier, we must anticipate what the performance gain might be. We should not try to improve something that is happening infrequently at the cost of complicated configuration and administration efforts. We must be aware of the fact that all DCE and DFS components are extensively caching. If clients are statically acting on the same resources all day long, caching is very effective, and the cell layout can be very simple.

Security: The binding handles of all security servers are defined in the file /opt/dcelocal/etc/security/pe_site. This file builds a fallback address repository and is consulted when the binding handle for the security daemon is not in the client’s CDS cache and CDS is not reachable. The security API tries all binding handles in that file from top to bottom.

We can force the pe_site file to be used right away by exporting the environment variable BIND_PE_SITE. The top entries are built from the security server site specified in the `mkdce` command, which should be the master. All updates of the file for additional security server entries have to be manually initiated by calling

chpesite. The steps for *ev3* to always access the security server on *ev4* first would be:

1. Update the pe_site file to contain all existing security servers:

   ```
   chpesite
   ```

   This command overwrites the pe_site file with binding information about all existing security servers. The master security server is put on top of the list. In order for this command to succeed, CDS must be running normally. Otherwise, you must add the entries manually with an editor.

2. Edit the pe_site file so that the binding handles for *ev4* become the top entries.

3. Set and export the environment variable:

   ```
   export BIND_PE_SITE=1
   ```

However, this option has to be used with caution because it would introduce static definitions and manual interaction on each node. It does make sense in large LAN/WAN cells if there is a lot of security server access or many slow links. The security server is mainly accessed when a ticket needs to be issued. Once a ticket is issued, it remains valid for a configurable amount of time in which no further security server access is needed. So, an over-proportional load would occur when ticket lifetimes are too short as well as when many users log in at the same time or do frequent logins and logoffs.

CDS: Access is to the clearinghouse in the same LAN if the requested directory has a replica there. If the directory is not there, another CDS server is randomly selected, and there is no way to bias the CDS clerk towards a specific CDS server. The only configuration option we have for such cases is to put specific directories only on specific clearinghouses so that accesses over slow WAN links are minimized. In other words, we need to make sure that CDS is correctly designed so that all the directories with frequent access from the remote site have a replica there. See 2.5, “Planning the CDS Namespace” on page 26, for a CDS design discussion.

FLDB and DFS file server: Access to the FLDB can now be predetermined. On the DFS client, preferences can be set for the cache manager to access certain FLDB and/or file servers with higher priority. The `cm setpreferences` command does this.

As with CDS, we have to be careful while designing the layout of the servers. The need to contact the FLDB should be minimized, which can be achieved with a flat hierarchy of the file tree as far as mount points are concerned. What this means is that filesets should not be mounted too many levels underneath each other because during path-name resolution the FLDB has to be contacted at each mount point. The FLDB should never be replicated to locations that are connected over a slow WAN link with the rest of the cell.

The filesets should be defined location-oriented so they can be as geographically close to the DFS clients as possible. If there is a lot of read-only access, replicas should be made for load balancing. If the filesets are defined location-oriented, only few replicas have to be defined for each read-only fileset. This ensures that updates of certain filesets do not have to go to all locations, which would cause performance problems.

5.2.2.7 Availability Discussion

From an availability point of view, all resources should be replicated in all locations where at least read access is needed all the time and where there is a possibility that the communication link to the rest of the cell might become unavailable.

These requirements might introduce a conflict of interest with the configuration requirements for good performance as mentioned above. Putting replicated servers on branches connected with slow WAN links certainly enhances availability, but careful DCE cell design is required to also achieve load balancing and to avoid too much traffic on the slow links (see performance discussion above). One might be able to do that for a couple of branches, but for a cell with hundreds of branches, we would probably need more than three times as many DCE core servers, which is difficult to manage and costs a lot of money.

Instead of putting a sophisticated cell configuration in place, which automatically also complicates cell administration, it might be easier to just make the link to the central site more highly available. This can be achieved with a multiprotocol router network or by simply building a backup link that can be activated in case of a failure of the primary link. 5.2.3, “Scenario 6: A Branch Connected with Two Links” discusses this topic.

5.2.3 Scenario 6: A Branch Connected with Two Links

As shown in Figure 46, we connect the branch with two links to make the connection with the DCE servers more highly available.

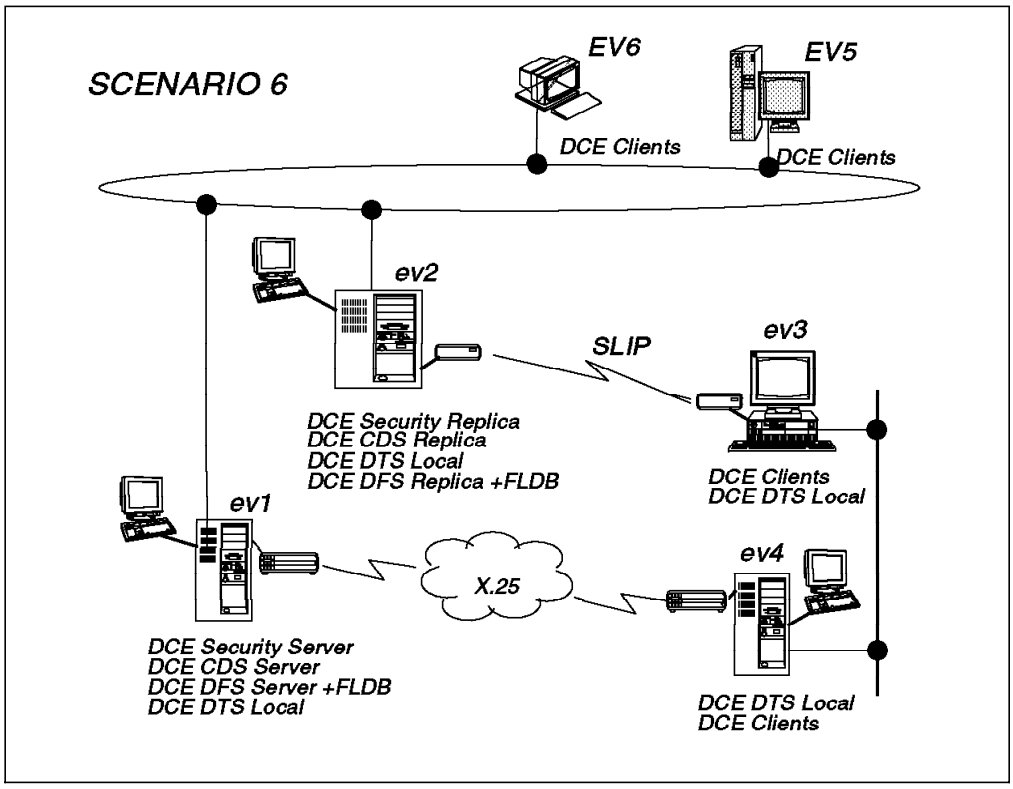


Figure 46. Scenario 6: A Branch Connected with Two Links

Because of a lack of time, we did not install this scenario. Nevertheless, we would like to discuss it.

The main purpose of this scenario is to provide redundant connections to the main site. This allows us to make DCE highly available in remote sites without having to install replica servers. We might want to install replica servers for performance reasons. However, this does not affect the recommendation to exclude all WAN interfaces from being used in binding handles. For instance, on a system with two X.25 interfaces and one SLIP interface, you would issue the following command:

```
# echo RPC_UNSUPPORTED_NETIFS=xs0:xs1:sl0 >> /etc/environment
```

The result is that when DCE server programs export all their binding interfaces, these interfaces are ignored and hence not exported to CDS or the local endpoint map.

In this way, we completely rely on TCP/IP routing for DCE calls crossing the WAN links. Suppose *ev3* needs access to the security server because a user logs in. It needs to contact CDS first to find a binding handle for the security server. The servers are all on *ev1*. Since broadcasting is not supported on most WAN links or IP routers, the cds_clerk on *ev3* needs a hint where CDS is located. This is done by the command `cdscp defined cached server`. This command is implicitly added to the /etc/rc.dce start-up file by the `mkdce -c CDS_server_name` command. Since CDS on *ev1* has only exported its token-ring (T/R) interface, the binding handle for CDS contains the T/R IP address. Thanks to correct IP routing definitions, the call from *ev3* to *ev1*'s T/R network interface will be found over X.25 or SLIP, depending on which one is available and how the routes are set.

The call to CDS will return possible binding handles for the security server. Again, since we had excluded X.25, these will all be T/R addresses. The call to the security server will find its way to *ev1* thanks to IP routing.

With dynamic IP routing and multiple links, we will never get stuck with DCE time-outs because of having tried a binding handle for which the link is not available. Remember that server binding handles are randomly selected by all DCE/DFS clients. If X.25 were not excluded and we happened to get a binding handle for an X.25 network interface, chances are higher that IP routing would direct us to the X.25 link even though it might be down. We would experience a 30-second DCE time-out before the next handle is tried, which again could be an X.25 binding handle.

The problem of avoiding time-outs is shifted from DCE to TCP/IP, or setting up correct IP routing, respectively. Most likely, you will set up dynamic routing with *routed* or, preferably, *gated* because it supports more routing protocols and is more sophisticated. If TCP/IP encounters a problem with one link, the routes are adjusted to use the backup link. Routing mechanisms might even be able to optimize network usage and prioritize faster links if there are redundant routes between two nodes. Multiprotocol routers are usually able to do this.

The simplest case of redundant network connection to a branch is shown in Figure 46 on page 130. The X.25 network is the primary link, whereas the SLIP connection is a backup link only and is usually not up. The SLIP link would be manually started when a network operator is alerted that the X.25 network is down. If the routes are not managed by routing daemons on the DCE client machines, the routes then have to be manually changed with the `route` command.

There are many automation possibilities to get an environment somewhere between this most simple case of a SLIP backup connection and a full-fledged router network. The two connections can be any combination of X.25, ISDN, SLIP, or even something faster. The only concern is to make sure they are really independent of each other to minimize the chance that both links become unavailable at the same time.

5.2.3.1 Performance Discussion

The advantage of highly available network connections is we can focus on load balancing issues when we plan the layout of the server in the DCE cell.

As discussed in scenario 5, 5.2.2.6, “Performance Discussion” on page 128, there are many factors which need to be considered for a decision on whether to configure replicated servers in branches. The slower the network link, the more sophisticated the distributed CDS or DFS design needs to be to avoid unnecessary calls over the slow links.

5.2.3.2 Availability Discussion

By having redundant links, there is no need for replicated servers in the branches to have a highly available DCE environment. As outlined above, there are many levels of comfort with which such an environment can be built. The nice thing about shifting the responsibility for availability from DCE to TCP/IP is that we can decouple performance and availability issues to a great extent. We can limit our discussion about server replication to performance issues.

5.2.4 Scenario 7: Intercell Communication

In Figure 47 on page 133, the Ethernet network with *ev3* and *ev4* simulates a pretty much self-sufficient subsidiary of the company marked by the Token-Ring network.

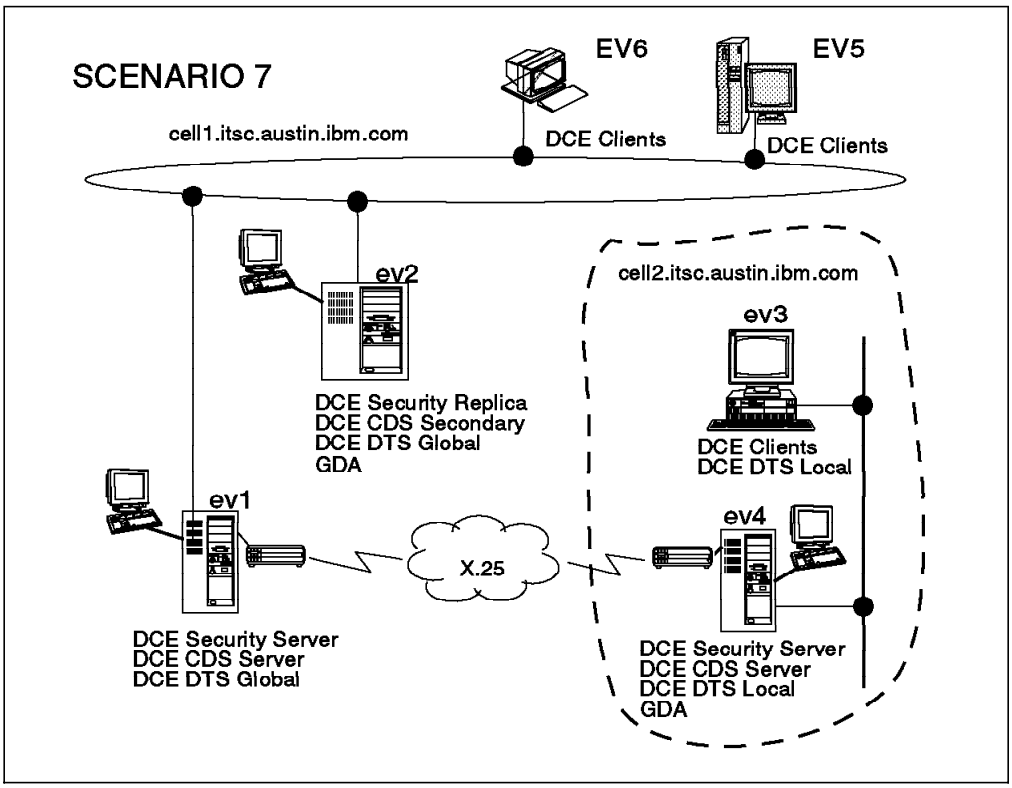


Figure 47. Scenario 7: Intercell Communication

To enable intercell communication, we must globally define both cells (either as X.500 or DNS). As shown in Figure 47, we have two DCE cells in the same TCP/IP domain (itsc.austin.ibm.com). So, we must define these cells in DNS.

In our environment, we set *ev1* as the DNS server. For our test environment, we did that in a quick-and-dirty way by defining *ev1* as a secondary DNS server to our department's DNS server. To enable intercell communication, you will have to define your cells on the DNS server. After registering the cells globally, you must establish a trust relationship between the two cells.

5.2.4.1 Preparation Steps

Before you configure any of the DCE machines, you should have:

- Created the necessary file systems
- Checked network name resolution (Extremely important here)
- Checked network routing
- Checked the network interfaces
- Synchronized the system clocks
- Installed DCE (last of these steps)

For details, see 3.2, "Preparing for DCE Configuration on AIX" on page 38.

5.2.4.2 DCE Configuration Steps

Following are all the configuration steps for the server machines of this scenario.

Configuring cell2: On ev4:

```
# mkdce -n cell2.itsc.austin.ibm.com sec_srv cds_srv dts_local gda
```

Do not forget to append the domain name to the cell name. Otherwise, you may fail later. Test a few commands to see if DCE is working correctly:

```
# dce_login cell_admin cell_password
# dcecp
dcecp> cell show
dcecp> directory list /.:
dcecp> principal cat
dcecp> quit
```

Configuring cell1: For all machines, follow exactly the steps in 5.2.1, “Scenario 4: A Small Branch Connected via WAN (X.25/SLIP)” on page 118. However, for the cell name, you put *cell1.itsc.austin.ibm.com*, and on *ev2*, you add a GDA.

On ev1:

```
# mkdce -n cell1.itsc.austin.ibm.com sec_srv cds_srv dts_global
```

On ev2:

```
#mkdce -R -n cell1.itsc.austin.ibm.com -s ev1 sec_srv cds_seond dts_global gda
```

5.2.4.3 Intercell Communication Configuration Steps

To register *cell1* globally, log in as *root* on *ev1*. Then log in to DCE as the cell administrator. Call smit in the following way:

```
# dce_login cell_admin <password>
# smitty mkdce
   ->Register Cell Globally
```

```
                              Register Cell Globally

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Name of INPUT File                                 []
  Name of named DATA FILE                            [/etc/named.data]



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Fill in the fields with your appropriate file names, and select **Do**. This command will add the following resource records to the named.data file:

```
;BEGIN DCE CELL /.../cell1.itsc.austin.ibm.com INFORMATION
;Initial CDS server
cell1.itsc.austin.ibm.com.              IN      MX      1       ev1.itsc.austin.
ibm.com.
cell1.itsc.austin.ibm.com.              IN      A       9.3.1.68
cell1.itsc.austin.ibm.com.              IN      TXT     "1 5583b618-f812-11ce-99
ca-10005a4f4629 Master /.../cell1.itsc.austin.ibm.com/ev1_ch 54aa689a-f812-11ce-
99ca-10005a4f4629 ev1.itsc.austin.ibm.com"
;Secondary CDS server
cell1.itsc.austin.ibm.com.              IN      MX      1       ev4.itsc.austin.
ibm.com.
cell1.itsc.austin.ibm.com.              IN      A       9.3.1.123
cell1.itsc.austin.ibm.com.              IN      TXT     "1 5583b618-f812-11ce-99
```

```
ca-10005a4f4629 Read-only /.../cell1.itsc.austin.ibm.com/ev4_ch 657883c2-f813-11
ce-9520-02608c2f0653 ev4.itsc.austin.ibm.com"
;END DCE CELL /.../cell1.itsc.austin.ibm.com INFORMATION
```

The first resource record (of type Mail Exchanger, MX) contains the host name of the system where the CDS server resides. The second resource record (of type Address record, A) contains the address of the system where the CDS server resides. The third record of type TXT, contains information about the replica of the CDS root directory that the server maintains. This information includes the UUID of the cell namespace, the type of replica (*Master*), the global CDS name of the clearinghouse (ev1_ch), the UUID of the clearinghouse, and the DNS name of the host where the clearinghouse resides.

The same information is added for additional CDS servers in the cell. After having added the information to the DNS server's configuration file, this command refreshes the named daemon. Run the following command to check that the machine, *ev4*, that runs the global directory agent in *cell2* can resolve the name of *cell1*. On *ev4*, type the following command:

```
# host cell1.itsc.austin.ibm.com
cell1.itsc.austin.ibm.com is 9.3.1.68
```

On Error

If the cell name is not known, make sure the /etc/resolv.conf file on *ev4* points to the name server on *ev1* and not to the regular name server of your main domain, unless you modified your primary name server for the intercell setup.

If this was not the reason, then signal the name daemon on *ev1*:

```
# kill -1 `cat /etc/named.pid`
```

If this does not help, try the following:

```
# stopsrc -s named; startsrc -s named
```

On any machine in *cell2*, log in as root and as cell_admin, and do the following:

```
# cdscp show cell as dns > /tmp/dns_cell2
# cdscp show clearinghouse /.:/* CDS_CHLastAddress >> /tmp/dns_cell2
```

Transfer the *dns_cell2* file to the DNS server (*ev1*), and call SMIT on *ev1* in the following way:

```
# smitty mkdce
   ->Register Cell Globally
```

```
                          Register Cell Globally

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
  Name of INPUT File                               [/tmp/dns_cell2]
  Name of named DATA FILE                          [/etc/named.data]
```

This command will add the information of cell2 to the DNS data file and refresh the DNS server (named process). Note that a temporary file, such as

/tmp/dns_cell2, was not necessary in *cell1* because the cell registration on the DNS name server machine has direct access to *cell1* information.

Now, we must establish the *direct trust peer relationship* between the two cells. The gdad daemon must be running on both cells. If it is not running, start it with the following command on the system where it resides:

```
# rc.dce gdad
```

On any machine in cell1, run the following command:

```
# dcecp
dcecp> registry connect /.../cell2.itsc.austin.ibm.com -group none \
-org none -mypwd dce -fgroup none -forg none -facct cell_admin -facctpwd dce
```

The registry connect command creates a mutual authentication surrogate in both cells. You can also create the same trust relationship by running the rgy_edit command as follows:

```
# rgy_edit
Current site is: registry server at /.../cell1.itsc.austin.ibm.com/subsys/dce/se
c/master
rgy_edit=> cell /.../cell2.itsc.austin.ibm.com
Enter group name of the local account for the foreign cell: none
Enter group name of the foreign account for the local cell: none
Enter org name of the local account for the foreign cell: none
Enter org name of the foreign account for the local cell: none
Enter your password:
Enter account id to log into foreign cell with: cell_admin
Enter password for foreign account:
Enter expiration date [yy/mm/dd or 'none']: (none)
```

On Error

If you get an error message, make sure that the GDA is correctly configured on *ev4*.

```
# host cell1
cell1.itsc.austin.ibm.com is 9.3.1.68
```

If this command does not find the address of *ev1*, check the /etc/resolv.conf file on *ev4*. It must point to the name server on *ev1*. Then stop and restart the GDA daemon on *ev4*:

```
# dce.clean gdad
# rc.dce gdad
```

You can now view the contents of the registry database to check the principals that have been created in both cells. Still in *cell1*, you can access the registry service of *cell2* with the following commands:

```
# rgy_edit
Current site is: registry server at /.../cell1.itsc.austin.ibm.com/subsys/dce/se
c/ev2   (read-only)
rgy_edit=> site /.../cell2.itsc.austin.ibm.com
Site changed to: registry server at /.../cell2.itsc.austin.ibm.com/subsys/dce/se
c/master
rgy_edit=> do p
Domain changed to: principal
rgy_edit=> v
```

```
nobody                                    -2
root                                       0
daemon                                     1
 ...
dce-ptgt                                  20
dce-rgy                                   21
cell_admin                               100
krbtgt/cell2.itsc.austin.ibm.com         101
hosts/ev4/self                           102
hosts/ev4/cds-server                     103
hosts/ev4/gda                            104
krbtgt/cell1.itsc.austin.ibm.com         108
```

In the registry of *cell2*, the registry connect command has created a principal, *krbtgt/cell1.itsc.austin.ibm.com* for *cell1*. You can run the same command on *cell1* to check the principal that has been created for *cell2*. The principal there is *krbtgt/cell2.itsc.austin.ibm.com*.

To further check out intercell access from a machine in *cell1*, you can log into an account of *cell2*:

```
# dce_login /.../cell2.itsc.austin.ibm.com/cell_admin
Enter password:
```

5.2.4.4 Scenario Experience

We can configure intercell communication between two cells. If DNS has been correctly configured, it is not difficult to configure intercell communication. But some commands take more time to respond. For example, when we issue the following on *ev4* to see the CDS namespace on cell1:

```
# cdsli -woRld /.../cell1.itsc.austin.ibm.com
```

The response time is acceptable, but takes more time than the single cell case. This means intercell communication is a burden to DCE. Especially in WAN environments, you should take care when you design cell topology. If you estimate many intercell RPCs, you should consider the additional burden.

Chapter 6. Administering DCE Cells

We identified a list of tasks administrators might have to perform in their DCE cell(s) and which we felt were not documented sufficiently or not supported by the existing commands and tools. We created the task list from our own experience with customers and from issues which were discussed in news groups or with development.

We grouped them into categories of tasks, some of which are overlapping and could have been assigned to other categories as well. You might find a certain task you want to perform in a different place than you would expect, or you might not find it all because, besides our creativity, time was a limiting factor. We cannot claim to present a complete workbook for administrators, but we believe at least some useful guidelines, tools, and step-by-step instructions are included.

Our task categories are:

- Migrating from IBM DCE 1.x to IBM DCE 2.1
- Changing Cell Configurations

  Once defined, cells cannot easily be reconfigured. Changes of IP addresses, host names, server locations, or even splitting and joining cells are realistic challenges for administrators. Machines can be added and servers can be replicated or moved as the customer's business grows. Faster networks can be added, and slower networks can be removed.
- Backup/Restore

  All of the core DCE servers and DFS servers can be replicated; so there seems to be no need for backup. However, one can never completely exclude bad things from happening. Databases can be corrupted by inadvertent administrator actions or software defects.
- Mass user and group management

  This category shows how to perform tasks such as adding, modifying, and deleting users, accounts, and groups in DCE on a large scale.
- Managing the cell_admin account

  cell_admin is per default the omnipotent DCE account. If the cell_admin password or the entire cell_admin account gets lost, specific steps have to be followed to restore the lost information.
- Integrating an NFS/NIS environment

  Many customer installations today use NFS/NIS to store common configuration files and share files. The purpose of this section is to discuss and give instructions on how to integrate NFS/NIS into DCE/DFS and how to migrate from NFS/NIS to DCE/DFS.
- Managing Remote Servers

  The DCE daemon (dced) running on every DCE node manages the DCE processes and the DCE application servers running on its system. This section explains the dced objects and how remote DCE (application) servers can be configured and managed from any single point in the cell. This includes server password management.

6.1 Migrating a DCE 1.x Cell to DCE 2.1

This section outlines our experience in migrating machines in a cell configured with DCE 1.3 (on AIX Version 3.2.5) to DCE 2.1 (on AIX Version 4.1.4+; which means 4.1.4 or higher). We set out the steps that we used to migrate the cell.

If a customer has a one-machine cell and no other AIX machines available, options for migration are very limited. A machine with AIX 3.2.5 and DCE 1.3 can be migrated very easily. Migrate the AIX (standard migration), install the new DCE 2.1 code *without reconfiguring* it, and restart DCE. The databases will be automatically converted to the new format.

If, however, a customer has the DCE servers spread over several machines in a cell, a couple of options are available. The customer can choose to migrate all of the machines in the cell, one after the other, to AIX Version 4.1.4 and then to AIX DCE Version 2.1 as explained above for the one-machine cell. The problem here is the availability. While a machine is being upgraded, the DCE services it provides are unavailable.

The migration strategy that we employ takes advantage of the fact that, by design, machines running OSF DCE 1.0.x and machines running OSF DCE 1.1 may coexist within a cell. We also take advantage of the fact that the primary security server and the initial CDS server can be moved to other machines in the cell. So, we move the DCE services to another machine in the cell while the original server machine is upgraded. This second approach gives the administrator more flexibility in scheduling the upgrade of the machine, and the server is made unavailable for only a few minutes. The downside of this is that moving servers around can be very complex.

Because servers and clients with different DCE levels can coexist in the same cell, the upgrade process does not have to be performed all at once. In any case, we recommend creating backups for DCE servers before you run the migration.

6.1.1 Compatibility

We tested a mix of DCE servers and clients running at both the AIX DCE 1.3 level (OSF DCE 1.0.3) and the AIX DCE 2.1 level (OSF DCE 1.1) in the same cell, and this is what we found.

Servers and clients at both levels can coexist in the same cell. You can even use the dcecp interface from an AIX DCE 2.1 machine to perform a subset of functions on AIX DCE 1.3 machines. The functions you can perform are limited to those you would normally be able to perform with the `cdscp` or the `sec_admin` commands. Commands for distributed management of clients and servers are not available for AIX DCE 1.3 machines because the dced daemon is not available on those machines. The value would be in using the dcecp shell (or the Tcl language) to automate or customize some tasks.

Some features of the CDS in OSF DCE 1.1, such as cell name aliases and hierarchical cells, require that you upgrade the Directory Version attribute on the cell root directory to Version 4.0. The default Directory Version for AIX DCE 2.1 is 3.0. At the 3.0 level, the CDS of AIX DCE 1.3 and 2.1 is compatible. AIX DCE 1.3 CDS is not compatible with CDS Directory Version 4.0.

Note: Cell name aliasing and hierarchical cells are still not available to any vendor from OSF or the integrator responsible for these CDS changes.

By design, the Security Service provides for incompatibilities between security clients and servers with different DCE levels. For instance, DCE servers on the OSF DCE 1.1 level provide Extended Registry Attributes (ERAs) that are included in the Privilege Attribute Certificates (PACs), now called Extended PACs (EPACs). The EPACs are used to enable authorization checks. Towards OSF DCE 1.0.x clients, the OSF DCE 1.1 security server behaves like an OSF DCE 1.0.x server and does not expect the client to send preauthentication information. In tickets issued for use with OSF DCE 1.0.x servers, it does not provide ERAs and EPACs, only PACs. For more details, see DCE Security Service component and facilities in the *Understanding OSF DCE 1.1 for AIX and OS/2* redbook, SG24-4616-00.

As outlined above, we are able to move server functions between machines running AIX DCE at the 2.1 and the 1.3 level. This allows for a greater degree of flexibility when migrating a cell to the new DCE version. Although we used these procedures in a migration process, they have equal value in maintaining the availability of a cell.

6.1.2 One-Shot Migration

Each individual machine in a cell can be upgraded from AIX 3.2.5 with DCE 1.3 to AIX 4.1.4 and DCE 2.1. We upgrade an AIX Version 3.2.5 system to AIX 4.1 using the migration option. After the base operating system is at the new level, we have to install the DCE for AIX 2.1 code. When the installation completes, we find that DCE has been upgraded to the 2.1 level, and all of our data remains intact. We also find the installation procedure does not remove the reference to the AIX DCE 1.3 code from the object data manager. This can easily be remedied by using `smit install_remove` to remove the reference.

6.1.3 Migration Scenario

As outlined above, our strategy is to move the DCE services to other machines to provide continuous DCE availability during the migration. Figure 48 on page 142 shows our test environment for the migration.

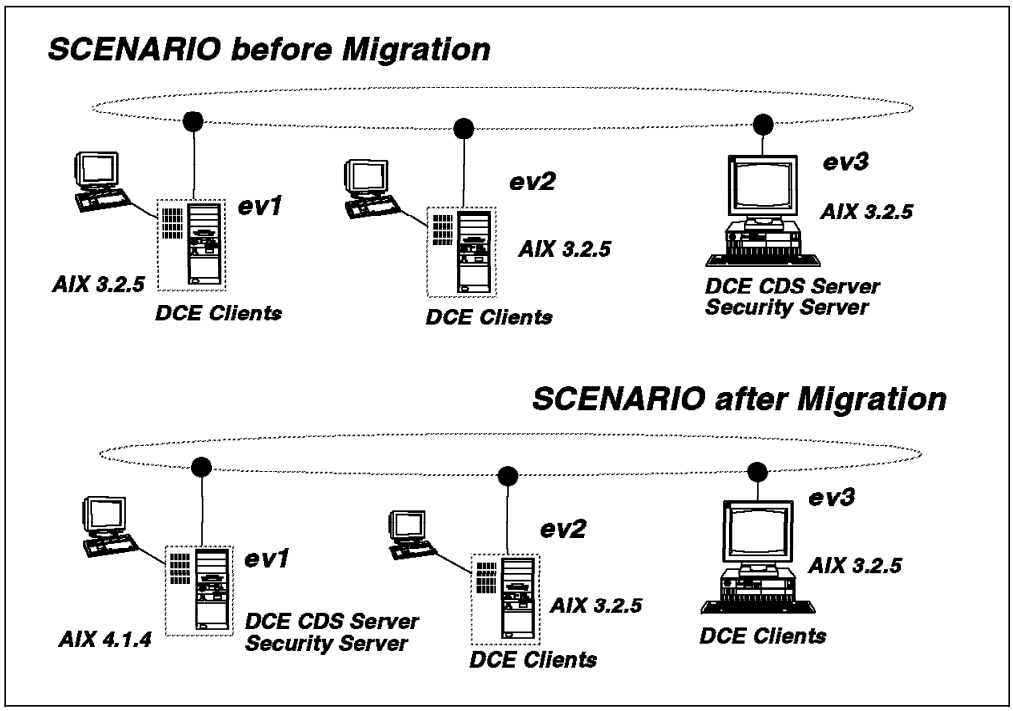


Figure 48. Migration Scenario

We first upgrade the DCE client machine *ev1* to AIX 4.1.4 and DCE 2.1. Then we configure secondary security and CDS servers on *ev1* and move the security and CDS servers from *ev3* to *ev1*. This is what corresponds to Figure 48 above.

Once the DCE core services are on another machine, we upgrade the original DCE server machines from AIX 3.2.5 to AIX 4.1.4 and install the new AIX DCE 2.1 code. Then we move the relocated services back to the original system.

6.1.4 Migrating the Security Server

To upgrade the security server on *ev3*, we move it to *ev1,* which was previously migrated to AIX 4.1.4. Then we upgrade *ev3* and move the security server back to *ev3*. Details about this procedure can be found in 6.2.4.4, "Relocating the Primary Security Server" on page 165. Here is a summary of the steps you can follow:

1. Migrate ***ev1*** to AIX 4.1.4 and configure DCE.
   a. Install AIX 4.1.4 on *ev1* using the migration option.
   b. Install the AIX/DCE 2.1 core components on *ev1.*
   c. Restart DCE (it still is a client):

      ```
      # /etc/rc.dce
      ```

   d. Configure a secondary DCE security server:

      ```
      # mkdce -R -r ev1 sec_srv
      ```

   Note: On a DCE 1.3 security server, you will get many of the messages shown below when it is accessed by requests from DCE 2.1 nodes. The messages, appearing on the console, can be ignored.

   ```
   RPC_CN_AUTH_VFY_CLIENT_REQ) on server failed status = 14129090
   RPC_CN_AUTH_VFY_CLIENT_REQ) on server failed status = 14129090
   ```

2. Stop the security server on ***ev3***:

```
# /etc/dce.clean secd
```

3. Log in to DCE as *cell_admin* on any machine in the cell.
4. Designate the previously defined security replica to be the new master:

```
# dcecp
dcecp> registry designate -master /.:/subsys/dce/sec/ev1
```

5. Upgrade the registry to the OSF DCE 1.1 level:

```
dcecp> registry modify -version secd.dce.1.1
```

6. Adjust the *pe_site* files on ***ev1*** and ***ev3***.

   Using your favorite editor, edit the */opt/dcelocal/etc/security/pe_site* file on both machines, and enter the correct addresses. Remove the entries pointing to *ev3* and any *ncacn_unix_stream* protocol entries (local RPCs) on AIX; they are no longer supported under AIX DCE 2.1.

7. Unconfigure the original master security server on ***ev3:***

   a. Stop and restart DCE:

   ```
   # /etc/dce.clean
   # /etc/rc.dce
   ```

   b. Unconfigure the original master server:

   ```
   # rmdce sec_srv
   ```

   Ignore the message warning you that you would have to reconfigure the entire cell. The master server is not on *ev3* anymore.

8. From ***ev1*** delete the entries for the old master from CDS and the replica list:

```
# dcecp -c rpcgroup remove -member /.:/subsys/dce/sec/master /.:/sec
# dcecp -c rpcgroup remove -member /.:/subsys/dce/sec/master /.:/sec-v1
# dcecp -c object delete /.:/subsys/dce/sec/master
# dcecp -c registry delete /.:/subsys/dce/sec/master -force
```

9. Restart DCE also on the new security server on ***ev1***:

```
# /etc/dce.clean
# /etc/rc.dce
```

10. Change all *pe_site* files of all your clients (here on ***ev4***).
11. If time-outs are experienced on the DCE clients now, a refresh of their CDS cache might help:

```
# dcecp -c rpcgroup list /.:/sec
/.../test/subsys/dce/sec/ev1
# dcecp -c rpcgroup list /.:/sec-v1
/.../test/subsys/dce/sec/ev1
```

Now migrate the CDS Server before you upgrade *ev3* to AIX 4.1.4.

Note: There is no longer a security server named *master*. However, some other programs, such as the DFS configuration, rely on the presence of the original master server name. Therefore, we recommend creating a secondary security server with the following command:

```
# mkdce -R -r master sec_srv
```

Then either just designate this new server the master or follow the above steps for moving the security server again, correctly replacing the names *ev1* and *master* .

6.1.5 Migrating the CDS Server

This section contains instructions on how to relocate a CDS server to another system in the cell. We will migrate the CDS server from AIX/DCE 1.3 to AIX/DCE 2.1.

The improved copy_ch command provided on the diskette in this book is used to move the clearinghouse from one machine to another. In certain circumstances, the script may produce warnings or even errors. In this case, do not stop the execution of the program. Let it finish and clean up remaining entries on the original CDS server later on. The messages will not produce any loss of data.

For details about this procedure, see "Relocating a CDS Server by Merging Clearinghouses" on page 163. Below is a summary of the steps to follow in our scenario:

1. Configure an additional CDS Server on ***ev1***:

   ```
   # mkdce cds_second
   ```

2. On ***ev1*** run the script provided on the diskette of this book:

   ```
   # copy_ch -m -s ev3_ch -t ev1_ch | tee logfile
   ```

   The command only runs on a DCE 2.1 machine. It takes a while to finish. Log all messages into a log file. In order to be able to check the result of the copy_ch command, we recommend to also run the list_ch command (see next step) beforehand.

3. From ***ev1*** check the *source clearinghouse* (ev3_ch) for master replica entries:

   ```
   # list_ch | tee cds_struct
   ```

   This command lists all replicas in the cell with their location. If you find any leftover master replica entries in ev3_ch, you must move them manually as described in "Relocating the Master Replica for a Directory" on page 161.

4. Update the /opt/dcelocal/etc/mkdce.data file.

   You have swapped the *Initial CDS Server* and the *Additional CDS Server.* Now we must modify the */opt/dcelocal/etc/mkdce.data* files on both machines to reflect the change.

   On ***ev3*** change the line:

   ```
   from:   cds_srv     COMPLETE   Initial CDS Server
   to:     cds_second  COMPLETE   Additional CDS Server
   ```

   On ***ev1*** change the line:

   ```
   from:   cds_second  COMPLETE   Additional CDS Server
   to:     cds_srv     COMPLETE   Initial CDS Server
   ```

5. Unconfigure CDS Server on ***ev3***

   ```
   # rmdce cds_second
   ```

   If you have proceeded successfully so far, you can now unconfigure the CDS server with your *source clearinghouse* if you need to.

Now you can upgrade ***ev3*** to AIX 4.1.4, and if you want to bring back the Security and CDS server from ev1 to ev3, repeat all steps described in 6.1.4, "Migrating the Security Server" on page 142 and in this section, interchanging ev1 and ev3.

6.2 Changing Cell Configurations

Once defined, cells cannot easily be reconfigured. Changes of IP addresses, host names, server locations, or even splitting and joining cells are realistic challenges for administrators. Machines can be added and servers can be replicated or moved as the customer's business grows. Faster networks can be added, and slower networks can be removed. In this section, we describe the following tasks:

- Splitting a cell
- Joining cells
- Changing IP addresses
- Moving services
- Changing a replica into a master service

Many of these tasks are performed with tools we have developed or modified. The tools we use are all on the diskette that comes with this redbook.

6.2.1 Splitting Cells

Splitting an existing cell means defining some machines into another cell and moving some services, data, and users to the new cell.

Please Note

The steps outlined here are a summary of ideas based on our experiences in this project and were not tested due to lack of time. However, we wanted to include them to give you ideas on how to tackle this important issue.

Splitting a cell is a complex undertaking, and the necessary steps depend on what is installed in the cell. DCE applications cannot be discussed because every application can require different steps:

- If services are duplicated in the new cell, it might be possible to install them in the new cell without any difficulties or conflicts with the original cell.
- If services administer common data that needs to be split, you might be able to use application-specific tools. Or, in the worst case, you have to delete and redefine part of the data.

DCE core services and DFS servers with their databases have to be rebuilt in the new cell. The databases cannot be moved over cell boundaries, nor can they be backed up and restored in the new cell. So we actually need to extract all the necessary information from the databases in the old cell and reconfigure it in the new cell.

The biggest effort in splitting a cell is probably the migration of users with their files and ACL definitions. We have created a user management tool that is designed to support moving users and their associated data. For additional important information about migration of users and files, see the following sections:

- 7.5, “Mass User/Group (and ACL) Management” on page 271
- 6.6.3, “Migrating NFS Files to DCE/DFS” on page 218
- 2.4, “Planning the User Namespace” on page 25

The following list describes a general procedure to move users from one cell to a new cell.

1. Create a list of user names to be moved.
2. Create a list of groups to be moved or copied.
3. If you are not sure whether the UDFs (user definition files) of these users are up to date, run `get_info_users`.

   This step extracts the current registry definitions and all ACLs for each user in the list and updates the UDFs.
4. Run `get_info_groups` for all groups that need to be created in the new cell.
5. Suspend the users with the `susp_users` command.
6. Delete the users with the `del_users` command.

   Before any user is deleted from all groups and from the DCE registry, all ACLs for this user are deleted. Then the UDF is moved to the cemetery directory.
7. Delete the groups which will not be present in the old cell anymore with the `del_groups` command.

   This step deletes the groups from the security registry and moves their GDFs to the cemetery directory, provided that they do not have any members left.
8. Back up the DFS files with AIX commands such as `tar`.

   The ACL information is intentionally destroyed by this step. The information to recreate it is in the UDFs/GDFs of the deleted users/groups. The UDFs/GDFs can be edited to remove entries not desired in the new cell, if necessary. This can be done with shell scripts that make global changes in multiple UDFs/GDFs.

   If ACLs were conserved by using DFS dump and restore, it would be a tedious job to adjust all ACLs to the new cell name. First we would have to define the users with the same UUID in the new cell. Then we would have to edit each ACL to change the cell name, and we might have to delete user and group entries that belong to nameless UUIDs because their users or groups do not exist in the new cell. Furthermore, it might also be necessary to change the cell name in ACL entries for foreign users or groups.
9. Delete the machines in the old cell with `rmdce all`.

   It might be necessary to move services away from those machines first. See 6.2.4, “Moving Services Within the Cell” on page 157 on how to achieve this.
10. Install the DCE and DFS servers in the new cell.
11. Move the UDFs (from the cemetery) to the new cell.
12. Copy the GDFs of groups which will be in both cells and move them together with the GDFs of the deleted groups to the new cell.
13. Inspect the UDFs/GDFs and make changes, if necessary.

    Since we are going to a new cell, the UIDs will remain unique and need not be changed.
14. Add the groups with `add_groups`.
15. Add the users with `add_users`.

    The `add_users` procedure only adds DCE users. If you are not using the integrated login, you would have to create AIX user accounts as well.

16. Run `rgy_enable_users` to enable the DCE accounts.
17. Create the necessary fileset hierarchy in DFS.

    If you do not assign a fileset to each user, you should at least create their home directory now so that the initial creation ACLs can be assigned before the files are restored.
18. Set the initial object and container creation ACLs.
    - Manually for filesets which are not covered in the UDFs
    - Run `dfs_enable_users` to apply all ACLs to the users' home directories
19. Restore the DFS files.

    Do this as cell_admin, and use `tar -xp` to preserve file ownership and permission bits.

    Be aware that when you list the files, the owner seems to be non-existent from an AIX point of view. If you are using the integrated login, then you did not define local AIX accounts for the users and file owners are shown as UIDs and GIDs.
20. Once all DFS and CDS objects are present, run `acl_enable_users` to apply all ACLs the users might have in those objects.
21. Run `update_groups` to apply all group ACLs.

6.2.2 Joining Cells

For a general procedure on how to join cells or move parts of an existing cell to another cell, see 6.2.1, "Splitting Cells" on page 145. The procedure is basically the same: We cannot directly move anything across a cell boundary; we must extract the necessary information, delete it at the old place, and reconfigure it in the new cell.

If you are in an environment that you had originally designed for unique UIDs and principal names across multiple cells as discussed in 2.4, "Planning the User Namespace" on page 25, you can go ahead and follow the procedure outlined for splitting a cell.

Otherwise, you must inspect all UDFs/GDFs you are going to migrate to another cell. To do this, run `get_all_info` also in the target cell, and check the two sets of UDFs/GDFs for conflicting UIDs/GIDs and/or user/group names.

Some groups might be the same in both cells. To merge these, you must make sure they have the same GID in both cells. Otherwise, you must run a global change in one of the cells. For groups which will be new in the target cell, you must check for GID conflicts and possibly run a global change, too. Follow the same steps as outlined below for the users.

For every UDF of the old cell that would cause a conflict in the new cell, you must do the following in the old cell after you have deleted the respective user:

1. Find a new user name and/or UID.
2. Recursively change the user's DFS file ownerships to the new UID.
3. Find file or directory names that contain the old user name and change them to the new name. An example of this is the home directory name.
4. Change name and contents of the UDF to reflect the new name, UID, home directory, and so on.

Once this is done in the old cell, the procedure is the same as for splitting a cell. You can then continue with the step that backs up the DFS files.

6.2.3 Changing IP Addresses

Several reasons might make it necessary to change the IP address of a workstation or server. For instance, a machine is to be relocated to another floor or a whole network is to get a new IP address. If this happens, we need to change the TCP/IP definitions, such as LAN interface configuration, name server entry, and routing for the machines involved. As a summary of the considerations presented in the following section, 6.2.3.1, "RPC Binding Information or CDS Towers," we can say that for DCE we need to change every occurrence of an IP address in CDS and in all clients' caches.

To support the necessary reconfiguration steps in DCE, we have created the following shell scripts:

`cleanif`	Searches for IP address entries in the namespace
`cleanup_ip`	Changes the evaluated entries
`renew_dir_entries`	Updates Tower information in affected CDS directories

The following shell scripts are also needed to refresh the local CDS cache on each client machine. They are described in 6.3.6, "Managing Caches on Client Machines" on page 181:

`cleanup_cache`	Refreshes CDS and credential caches; requires DCE restart
`cleanup_cds_cache`	Refreshes CDS clerk cache only, without DCE restart
`create_cds_entry`	Enters knowledge of a CDS server into the CDS clerk cache

The following subsections describe:

- Binding handles
- Each shell script
- The generalized procedure on how an IP address change is performed with the help of our scripts
- Our experiences

6.2.3.1 RPC Binding Information or CDS Towers

The difference between, for example, reading a local file on a single machine and performing the same read on a remote file in DCE is like the difference between reading information from a phone book yourself and dialing an operator for the same information. The remote operation requires the addition of another active entity that can be requested to perform it for you. Associated with every remote object (for instance, a data file) available on a network is a remote server to manage that object and make it available. The user may not be aware of that server, but it is there.

Clients call remote procedures. They need to find a service on a remote server node. CDS is the operator that tells you which number to dial to get to that server node. The number, also called a binding handle, contains an IP address and is stored in the directory service.

The DCE documentation often speaks of *binding to an object*. In reality, clients can bind only to servers, which may then be requested to perform operations on objects that are under their management. A binding handle consists of the server node IP address, a protocol sequence such as UDP or TCP, an interface UUID to select the server process, and object UUIDs to specify certain objects on

which an action is to be performed. The following example shows RPC binding information as it is stored in CDS for access to the FLDB server:

```
# rpccp show entry /.:/hosts/donald/flserver
objects:
  001a1f6c-4816-1e0a-9950-08005a01befd
binding information:
  <interface id>   003fd39c-7feb-1bbc-bebe-02608c2ef4d2,1.0
  <string binding> ncadg_ip_udp:9.13.113.156[]
  <string binding> ncacn_ip_tcp:9.13.113.156[]
```

The same entry displayed from a CDS point of view:

```
# cdscp show object /.:/hosts/donald/flserver
              SHOW
            OBJECT   /.../itsc.austin.ibm.com/hosts/donald/flserver
                AT   1994-09-09-00:09:24
  RPC_ClassVersion = 0100
   RPC_ObjectUUIDs = 6c1f1a0016480a1e995008005a01befd
          CDS_CTS = 1994-06-24-02:42:32.898697100/08-00-5a-01-be-fd
          CDS_UTS = 1994-06-24-02:42:34.844623100/08-00-5a-01-be-fd
        CDS_Class = RPC_Entry
  CDS_ClassVersion = 1.0
       CDS_Towers = :
            Tower = ncacn_ip_tcp:9.13.113.156[]
       CDS_Towers = :
            Tower = ncadg_ip_udp:9.13.113.156[]
```

The string bindings as seen with the rpccp command are stored as CDS attributes called CDS_Towers. We will use the term Towers hereafter.

Other entries, such as the one for */.:/subsys/dce/sec/master,* can have multiple interfaces and object UUIDs managed all by one server process. To build binding handles from this CDS entry, all object UUIDs, interface UUIDs, and string bindings are combined. So, in the above example, we would get two binding handles, one for UDP and one for TCP.

The following figure shows an example of the tree structured CDS namespace. To make it more confusing, CDS calls its leaf entries objects.

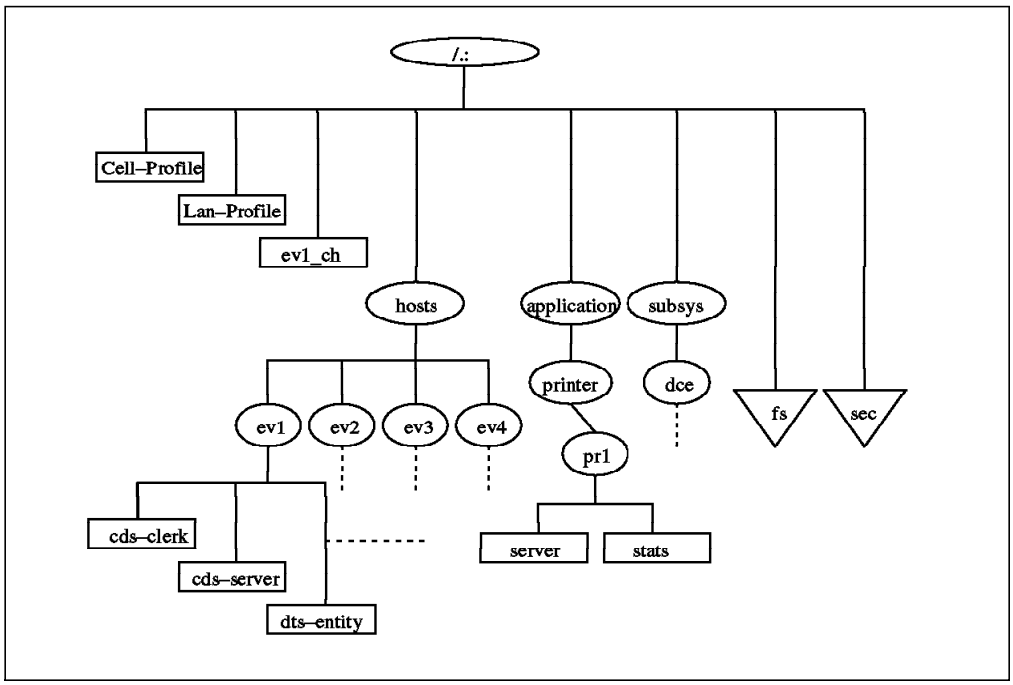


Figure 49. Extract of a CDS Namespace

Many services mediated by CDS can be running on the same server node. That means that multiple CDS objects can contain one specific IP address multiple times.

Once a client has obtained the dial number for a service, it memorizes (caches) the information so it does not have to call the operator again for the same information. If the number changes, we have to tell that to each client that memorized it.

To change an IP address in a cell, we need to change every occurrence of it in CDS and in all clients' caches.

6.2.3.2 The cleanif Procedure

We created the `cleanif` shell script, which finds all the objects in the namespace containing a specific IP address. It generates `rpccp` commands to change the binding information pertaining to that IP address for all these CDS objects. The generated commands are written into a file.

```
#cleanif
Usage: cleanif -i <ipaddr> -n <newipaddr> -f <output file> [-s] [-h]

  -i <ipaddr>             ipaddr is the IP address to be removed
  -n <newipaddr>          new IP address to replace iaddr
  -f <output file>        generated rpccp input file
  -s                      save temp files (for debugging cleanif)
  -h                      help
```

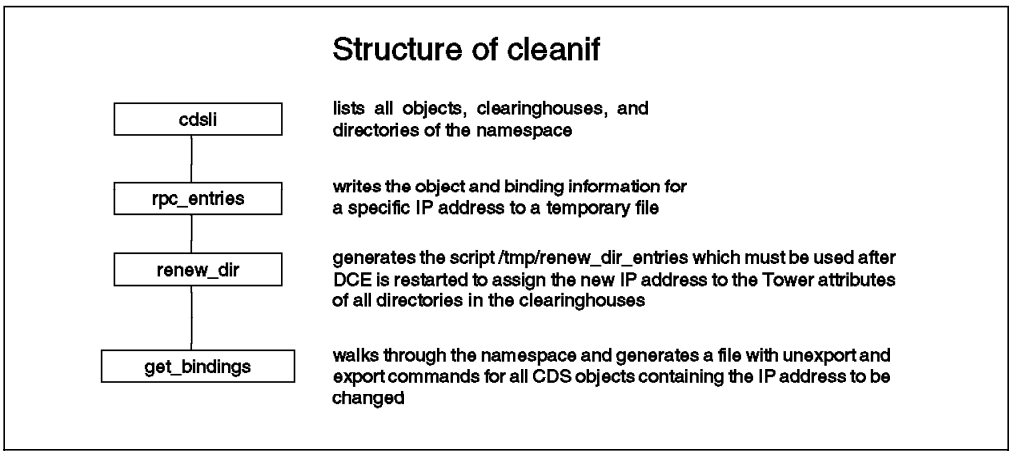


Figure 50. Workflow Description of the cleanif Procedure

The file that is generated contains unexport and export subcommands for the rpccp command. The unexport commands affect CDS entries. They remove interfaces that contain binding handles with that IP address. Since unexport can only remove an entire interface, which might also contain other valid IP addresses, we must re-export the interface with the bindings that are still valid and with the ones that have a new IP address. This is what the generated export commands do. The following is an extract of a generated file:

```
unexport -i 003fd39c-7feb-1bbc-bebe-02608c2ef4d2,1.0
-o 00013376-feed-1eb0-b6cc-10005aa8cff8 /.:/hosts/ev2/bosserver

export -b ncadg_ip_udp:9.3.1.123[]  \
-i 003fd39c-7feb-1bbc-bebe-02608c2ef4d2,1.0  \
-o 00013376-feed-1eb0-b6cc-10005aa8cff8 /.:/hosts/ev2/bosserver

export -b ncacn_ip_tcp:9.3.1.123[]  \
-i 003fd39c-7feb-1bbc-bebe-02608c2ef4d2,1.0  \
-o 00013376-feed-1eb0-b6cc-10005aa8cff8 /.:/hosts/ev2/bosserver

unexport -i 4ea31de8-9a94-11c9-bb60-08002b0f79aa,3.0  \
-o dc8c6fc0-6143-11ca-b4b9-08002b1bb4f5 /.:/hosts/ev2/cds-clerk

export -b ncadg_ip_udp:9.3.1.123[]  \
-i 4ea31de8-9a94-11c9-bb60-08002b0f79aa,3.0   \
-o dc8c6fc0-6143-11ca-b4b9-08002b1bb4f5 /.:/hosts/ev2/cds-clerk

export -b ncacn_ip_tcp:9.3.1.123[]  \
-i 4ea31de8-9a94-11c9-bb60-08002b0f79aa,3.0   \
-o dc8c6fc0-6143-11ca-b4b9-08002b1bb4f5 /.:/hosts/ev2/cds-clerk
```

At the same time, cleanif creates another shell script called /tmp/renew_dir_entries. This procedure is explained in 6.2.3.4, “The renew_dir_entries Procedure” on page 152 and will be used only if CDS servers change their address.

This script was tested with DCE core services and DFS. It may not work for certain DCE applications. Therefore, use it with care and consider testing it before using it within a productive cell. Check the contents of the generated files. When you use it with DCE applications, be sure to stop all applications before you run cleanif. In this way, CDS will be cleaned from binding information that applications might export and unexport by themselves.

The cleanif procedure can also be used to search the entire CDS namespace for binding information containing a specific IP address:

```
# cleanif -i 9.3.1.120 -n 1.1.1.1 -f /tmp/out.cds
```

Notice the comments that are displayed. As soon as it says *used* for a CDS object or directory, you know that this object contains that IP address. However, do not run cleanup_ip, which would change all occurrences of 9.3.1.120 into 1.1.1.1 in CDS!

6.2.3.3 The cleanup_ip Procedure

The script cleanup_ip takes the file previously generated by cleanif and executes all the rpccp commands in it. Again, we recommend reviewing this file carefully before using it. Then call it as follows:

```
cleanup_ip -f <file_generated_by_cleanif>
```

6.2.3.4 The renew_dir_entries Procedure

If the node to be changed is a CDS server, then its IP address is stored in the CDS_Tower attribute of all directories that have an occurrence (replica) on that server. You can check them with cdscp show dir <dir_name>. The Tower is used to bind to the CDS server that hosts a certain directory. If the server's address changes, the Tower information has to be changed for all these directories.

The renew_dir_entries shell script is generated by cleanif. It has commands that rebuild each directory's replica set, which forces an update of the Tower information for each directory. The following is an example of what renew_dir_entries contains:

```
#!/bin/ksh
/bin/klist | grep Principal | grep cell_admin > /dev/null 2>&1
if [ $? -ne 0 ] ; then
  echo "Must be cell_admin, please dce_login."
  exit 1
fi
echo "\n !!! Execution may take a long time !!! \n"
cdscp set dir /.:/hosts to new epoch master /.../itsc.austin.ibm.com/ev1_ch
readonly /.../itsc.austin.ibm.com/ev4_ch
cdscp set dir /.:/subsys to new epoch master /.../itsc.austin.ibm.com/ev1_ch
  ...
```

Check the file before you execute it. In order to succeed, the whole replica set has to be specified for each directory to set a new epoch master.

6.2.3.5 Generalized Procedure to Change an IP Address

Figure 51 on page 153 contains a workflow description of all necessary steps to change an IP address. We have tested this procedure in many different combinations of DCE and DFS server configurations. All steps need to be performed on the system of which the IP address needs to be changed. Some additional steps need to be executed on other systems, as indicated, depending on the DCE/DFS server role of the system that has been changed.

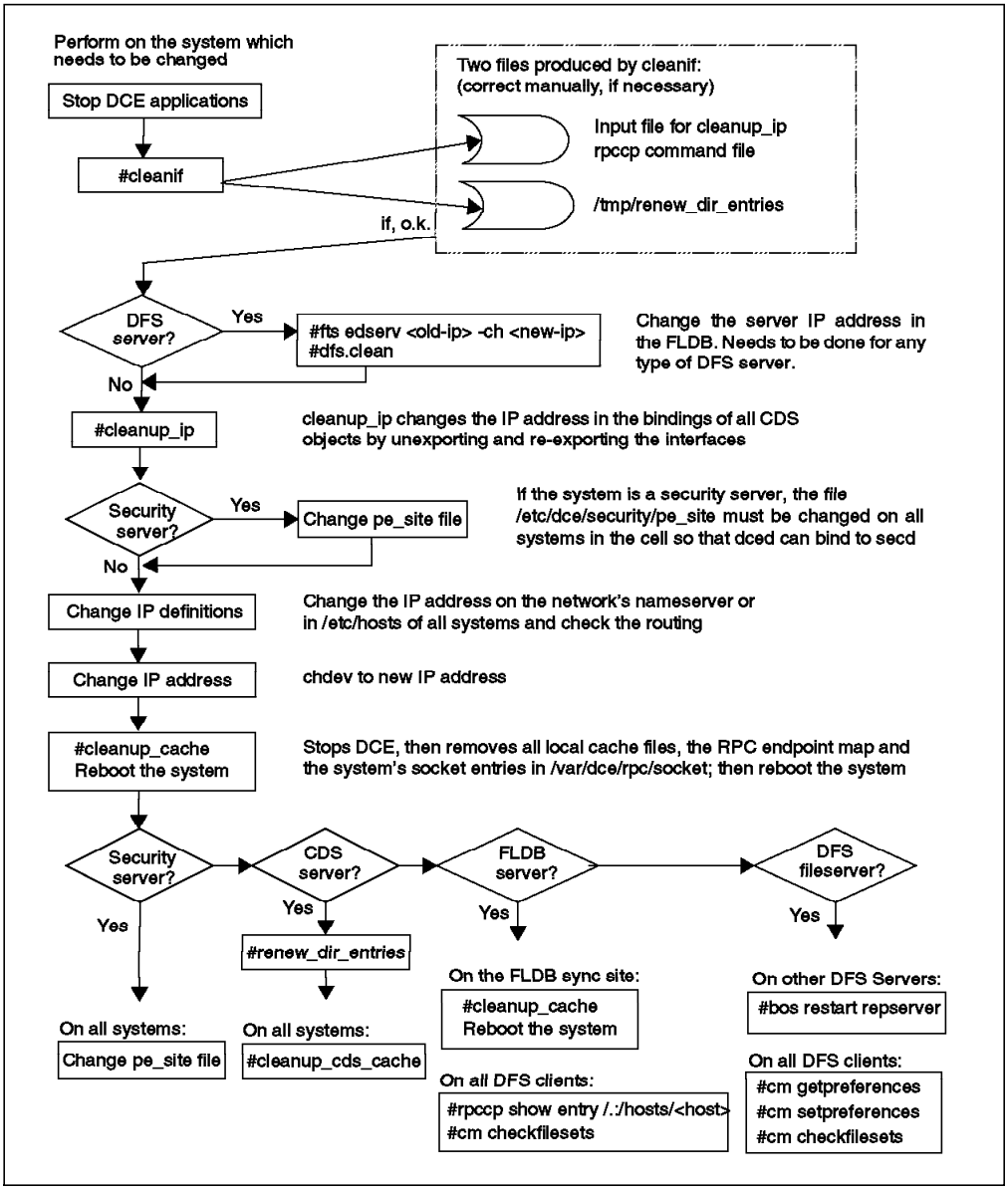


Figure 51. Workflow Description to Change an IP Address

6.2.3.6 Experiences

In order to change an IP address within all namespace entries, it is absolutely necessary that cell_admin has access to all these entries. If this is not the case, a full change of an IP address may not be guaranteed. However, in case of an unauthorized access to a directory, object, or clearinghouse, `cleanif` will alert you, and this information can be logged.

Remember that you must release a fileset and run `cm checkfilesets` on a DFS client to see any changes made to the read/write filesets. This actually has nothing to do with an IP address change, but it may be the reason for not seeing changes when you test your DFS after an IP address change.

The steps that need to be executed to change the IP address of a DCE/DFS node are outlined in Figure 51. What needs to be done on that particular machine and on other systems in the cell depends on the role of machine on which the

address is changed. If the machine has multiple roles, the suggested actions need to be performed altogether. Let's have a look at every machine role and the special considerations for each:

DCE Client: To change to IP address of a pure client, stop all applications first. Then follow the procedure; it should work. Only a few changes are necessary in CDS, and no other machines are affected. So, no cache refreshes are necessary. Rebooting the system is necessary if a DFS client runs on that system. Otherwise, restarting DCE might be sufficient.

Another option is to simply unconfigure DFS and DCE, change the IP address, adjust the TCP/IP definitions, and reinstall the client. If split configuration is used, the central administrator part has to be performed as well. This method is easier to understand, and hardly anything can go wrong.

DCE Security Server: The /opt/dcelocal/etc/security/pe_site file contains binding information to the security server. If you change an IP address on a security server, this file must be changed on all systems in the cell. If CDS is running normally, you can run `chpesite` on every AIX system to update the pe_site file. The following example shows the contents of a pe_site file:

```
/.../itsc.austin.ibm.com 006cd1ee-148c-1e06-8e09-10005a4f15da@ncacn_ip_tcp:9.3.1.68[]
/.../itsc.austin.ibm.com 006cd1ee-148c-1e06-8e09-10005a4f15da@ncadg_ip_udp:9.3.1.68[]
```

To change the address from 9.3.1.68 to 9.3.1.120, edit the file with a text editor:

```
/.../itsc.austin.ibm.com 006cd1ee-148c-1e06-8e09-10005a4f15da@ncacn_ip_tcp:9.3.1.120[]
/.../itsc.austin.ibm.com 006cd1ee-148c-1e06-8e09-10005a4f15da@ncadg_ip_udp:9.3.1.120[]
```

If time-outs or other problems are experienced on other machines in the cell, you might also want to perform additional actions on these machines:

- Applications might have security server bindings cached. In particular, long-running applications that automatically renew their ticket tend to cache a binding handle. For instance, dced is one. It keeps the machine principal valid. When dced is stopped and restarted with the following commands, it destroys all existing credentials for the machine principal:

  ```
  # dce.clean dced
  # dced -p
  ```

- One of the next commands should also be executed to refresh the CDS clerk cache entry for the security server, depending on whether it was the master or a replica server:

  ```
  # dcecp -c rpcentry show /.:/subsys/dce/sec/master
  # dcecp -c rpcentry show /.:/subsys/dce/sec/rep<number>
  ```

- If you do not know which other applications cache information, stop all applications and run `cleanup_cache` to destroy credentials and cache entries; see 6.3.6, "Managing Caches on Client Machines" on page 181.

CDS Server: We have tested the procedure in a cell with two CDS servers. The `renew_dir_entries` procedure recognizes the entire replica set of all directories if none of the replicas are excluded (see 6.2.3.4, "The renew_dir_entries Procedure" on page 152). However, check the contents of `renew_dir_entries` before you execute it to be sure the replica sets are correctly generated.

In AIX DCE 1.3, there used to be a file, /opt/dcelocal/etc/*cds_config,* in which the IP address had to be changed. This file no longer exists in DCE 2.1.

The CDS clerk caches should be refreshed if the client systems immediately need the updated information. After approximately 8 to 12 hours, the entries would be invalidated, and the clerk would obtain updated information from the CDS server if the caches are not refreshed. See 6.3.6.1, “Managing the CDS Clerk Cache” on page 181, for more information about cache refreshing.

DFS FLDB: The IP address of each FLDB server is stored in the FLDB. Change this address with the `fts edserv` command, and regularly stop DFS on the FLDB server on which you change the IP address. This allows for correct DFS token management before you run `cleanup_ip`.

Also determine which of the FLDB servers is the sync site:

```
# /opt/dcelocal/bin/udebug /.:/fs /.:/hosts/ev2 -long
Host 9.3.1.120, his time is -1
Vote: Last yes vote for 9.3.1.120 at -12 (sync site); Last vote started at -12
Local db version is 783799213.1
I am sync site until 78 (3 servers)
Recovery state 1f
Sync site's db version is 783799213.1
0 locked pages, 0 of them for write
This server last became sync site at -5981

Server 9.3.1.123: (db 783799049.1)
    last vote rcvd at -11, last beacon sent at -11, last vote was yes
    dbcurrent=1, up=1 beaconSince=1

Server 193.1.10.4: (db 0.0)
    last vote rcvd at -11, last beacon sent at -11, last vote was yes
    dbcurrent=1, up=1 beaconSince=1
```

Run this command for each FLDB server defined in the /.:/fs RPC group. If the IP address is changed on an FLDB server that is not the sync site, you must now run `cleanup_cache` and reboot the sync site server as well (just restarting the `flserver` on the sync site was not sufficient when we tested it). Otherwise, the sync site FLDB server would keep the old IP address cached, and ubik would try to synchronize the FLDB on the old address.

It is sufficient to reboot the sync site server only. Our recommendation, however, is to clean up and reboot all FLDB server systems in the cell because otherwise ubik might take a long time to synchronize all servers and eliminate the stale address everywhere.

Then check whether the sync site FLDB stores the correct IP address of the other FLDB servers with the following command:

```
# /opt/dcelocal/bin/udebug /.:/fs /.:/hosts/<sync_site_fldb> -long
```

The DFS client systems use the CDS entry /.:/hosts/<fldb-host>/self to get to an FLDB server. To avoid possible time-outs on the DFS clients, you may want to refresh this CDS clerk cache entry on all DFS clients:

```
# dcecp -c rpcentry show /.:/hosts/<fldb-host>/self
old # rpccp show entry /.:/hosts/<fldb-host>/self -u
```

If more than one FLDB server is running, the DFS clients would get to an FLDB without the above command. However, they may experience a 30-second time-out when the old binding information is tried.

New in this release of DFS (2.1) is that the clients maintain FLDB server preferences just as they did for DFS file servers in the previous release (1.3). So, use the `cm getp -fldb` command to check, and the `cm setp -fldb` command to update, the preferences if necessary. Look under *DFS file server* for details.

Run the `cm check` command to refresh the DFS clients' binding information to the FLDB and the to DFS file servers.

DFS File Server: If a DFS file server is to be changed, its IP address in the FLDB has to be changed first (`fts edserv` command). Then after all steps have been performed on this server and it is up and running again, all repservers for which the changed file server contains the master filesets need to be restarted:

```
# /opt/dcelocal/bin/bos restart /.:/hosts/<repserver-name> repserver
```

Otherwise, a release or update of the fileset would not reach the repservers. The release command for the fileset would seem to succeed, but actually, the read-only filesets would not be updated on the systems that had not restarted their repservers.

On all DFS client systems, the cache manager preferences need to be checked:

```
# cm getp
cm getp
ev4                                          40006
ev2                                          20015
9.3.1.123                                    20008
```

In the above example, *ev2* was changed from 9.3.1.123 back to its original address 9.3.1.120. The cache manager has the highest priority (lowest number) on a stale address where there is no longer a server. The only way to get rid of this entry is to reboot the client system. However, to avoid using the stale address, we could assign a very low priority to that address:

```
# cm setp 9.3.1.123 60000
```

Run the `cm check` command to refresh the DFS clients' binding information to the FLDB and to the DFS file servers.

DCE applications: When you use this procedure with DCE applications, be sure to stop all applications before you run `cleanif`. In this way, CDS will be cleaned from binding information which applications might export and unexport by themselves. After the change, the application should be able to start and export their new interfaces, provided that the DCE core services are running normally after the change.

If an installation procedure for an application had defined static binding information in CDS, these entries will be changed to the new IP address while the procedure depicted in Figure 51 on page 153 is executed.

However, if an application stores or caches binding information internally on its clients or peer servers, you have to somehow be able to refresh this application cache on each system running a client or another server of this application. This is very application dependent.

Also consider refreshing the CDS clerk caches on all systems in order to avoid time-outs. See 6.3.6.1, "Managing the CDS Clerk Cache" on page 181, for more information about cache refreshing.

Caution

Back up all your databases before you change the IP address of a server, and be ready to redefine the cell just in case. See 6.3, “Backup/Restore and Other Housekeeping Tasks” on page 166.

6.2.4 Moving Services Within the Cell

Client/server application frameworks are supposed to be easily scalable. When demand for a service increases, a new server can be added if the application is designed to allow for this. In such a dynamic environment, requirements for services can change. We must be able to move the following services from one machine to another, possibly without interrupting their availability:

- DCE applications
- DFS services
- CDS resources
- Security server

6.2.4.1 DCE Applications

The effort it takes to move a DCE application depends on how the application is designed. An application server may provide CPU access for parallel processing of numerically intensive tasks. This case is easy to handle. You can just add and delete servers.

It is more difficult to handle redundant services when they access data. You may have one defined master server and replicas which have a read-only copy of the data. This would be the DCE security server-type approach. Or you can have multiple servers that know of each other and agree on a master among themselves, such as the DFS FLDB.

There are many variants of the above-mentioned patterns. Each application should provide specific instructions on how to relocate their services.

There is one general rule for making applications relocatable. If an application leaves all address lookup tasks to CDS and does not store any addresses of peer servers locally, it is easier to relocate. The application may cache information, but the caches should be refreshable with a management command. In addition, it should register itself in CDS and remove the interfaces upon termination.

The relocation steps would then be:

1. Stop the service; this should remove the interfaces from CDS.
2. Remove all RPC mappings for that application from the local machine if the server is not going to be rebooted; a well behaving application does that by unregistering itself from dced.
3. Backup its data(base).
4. Install the server in the new location.
5. Restore the data(base).
6. Start the service; this should read the database and export the interfaces to CDS.

7. Refresh the CDS cache on all client systems if you want to prevent time-outs with one of the following methods:
   - Refresh the entire cache with the procedures described in 6.3.6, “Managing Caches on Client Machines” on page 181
   - Force the CDS client to read the CDS object from CDS rather than from the local cache with one of the following commands, depending on whether it is a regular RPC interface entry or an RPC group entry:

     ```
     # dcecp -c rpcentry show <entry_name>
     # dcecp -c rpcgroup list <entry_name>
     ```

     This will update the queried entries in the CDS clerk cache.

6.2.4.2 DFS Services

DFS has various machine roles that can be on the same or different machines. We discuss them separately. For more details, consult the *The Distributed File System (DFS) for AIX/6000* redbook.

FLDB and Backup Server: These two machine roles work the same as far as replication and database access is concerned. The ubik routines designate one master server and update that database. The other servers, if there are any, become slaves, and their database is automatically updated by the ubik routines.

All we have to do is add another FLDB server with `mkdfs -s <scm-host> dfs_fldb` issued on the new node and remove the old one with `rmdfs dfs_fldb`.

It is seldom necessary to restart client or server machines if you reconfigure a cell’s FLDB machines. As long as at least one FLDB machine remains the same after reconfiguration, all machines can continue to access the FLDB via that machine. Eventually, all machines will recognize the current set of FLDB machines.

If no FLDB remains the same, a CDS clerk cache refresh for the DFS junction, an RPC group entry, may be helpful. Use one of the following commands:

```
# dcecp -c rpcgroup show /.:/fs
old # rpccp show group /.:/fs -u
```

DFS File Server: Read/write filesets can be moved to other machines in the same cell, whereas read-only filesets should be removed and recreated as shown in the following steps:

1. Install new file server machine:

   ```
   # mkdfs -s <scm_machine> -e dfs_srv
   # mkdfs -s <scm_machine> dfs_repsrv
   ```

2. Create and export the necessary aggregates on the new site, for example:

   ```
   # mklv -t lfs -y lfsroot rootvg 1
   # newaggr /dev/lfsroot 8192 1024 -overwrite
   # mkdfslfs -d /dev/lfsroot -n lfsroot
   ```

3. Use the `fts move` command to move every read/write fileset to the new location:

   ```
   # fts move root.dfs ev1 lfsroot ev2 lfsroot
   ```

   The fileset to be moved must not exist as a replica on the target system; otherwise the move fails.

4. Follow the steps outlined in 4.4, “Replicating Filesets on AIX” on page 89, to create the same replica filesets on the new location as are defined at the old location

5. Use `fts rmsite` to remove all replica filesets at the old location.

   The replica filesets associated with read/write filesets that were moved away from this server need not be removed; they are transferred with the master copy.

6. Unexport the aggregate(s), and remove the logical volume(s):

   ```
   # rmdfslfs -n lfsroot
   # rmlv -f lfsroot
   ```

7. Remove the DFS file server at the old location, `rmdfs dfs_srv`, and reboot the system to get rid of the kernel extension.

SCM Machine: The System Control Machine (SCM) machine is running an upserver process that is contacted by the upclient processes of other DFS servers in the same administrative domain. An administrative domain is built by all DFS servers defined to belong to a particular SCM machine. Administrative lists, which define who is authorized to administer DFS servers, are maintained only on the SCM. The upclient processes then contact the upserver process on the appropriate SCM to send down all administrative lists.

The SCM machine does not know which client machines are in its domain. The clients know which SCM machine is theirs because it was specified with the `-s` flag of the `mkdfs` command.

So, what we have to do is:

1. Be sure to be root and cell_admin.
2. Install a new SCM machine with `mkdfs dfs_scm`.
3. Copy all administrative list files in directory /opt/dcelocal/var/dfs from the old to the new SCM machine (or backup/restore the files).
4. On each SCM client, which means on all DFS servers in the same administrative domain, change the entry for the upclient.scm (or upclient) process in the BosConfig file. Issue all of the following commands on one line each:

   ```
   # /opt/dcelocal/bin/bos stop -s <scm_client_system> -p upclient.scm
   # /opt/dcelocal/bin/bos delete -s <scm_client_system> -p upclient.scm

   # /opt/dcelocal/bin/bos create -s <scm_client_system> -p upclient.scm \
   -type simple -cmd '/opt/dcelocal/bin/upclient -s <new_scm_machine> -path \
   /opt/dcelocal/var/dfs /admin.bos /opt/dcelocal/var/dfs/admin.ft \
   /opt/dcelocal/var/dfs/admin.fl /opt/dcelocal/var/dfs/admin.bak'
   ```

5. On the old SCM site, delete the upserver process:

   ```
   # /opt/dcelocal/bin/bos stop -s /.:/hosts/<old_scm> -p upserver
   # /opt/dcelocal/bin/bos delete -s /.:/hosts/<old_scm> -p upserver
   ```

6. Delete the SCM machine on the old site:

   ```
   # rmdfs dfs_scm
   ```

7. Then create an upclient.scm entry for the former SCM machine if the machine runs other DFS servers and needs to become an SCM client now.
8. You can check the SCM clients with the following command:

```
# /opt/dcelocal/bin/bos status /.:/hosts/<scm_client_system> upclient.scm
-long
```

6.2.4.3 Moving CDS Resources

CDS controls a distributed database. Replication is performed on the directory level. Each directory is a replica, one of them being the master. Master replicas of different directories can be on different systems. Before we move CDS resources, we should know the structure of our namespace. Therefore, we include a short caveat to show how to analyze the namespace:

- How to list directories?
- Where are replicas of a directory and which one is the master?
- What type of directory is in the /.:/ev1_ch clearinghouse?

The commands that give answers to the above questions can be combined to answer other questions, such as: ″Where are the master replicas for all directories″?

Then we want to look at relocating different CDS resources, such as:

- Master replica of a directory
- Clearinghouse
- CDS server

After all these changes, a refresh of all CDS clerk caches is useful to avoid time-outs caused by a client still trying to access outdated information. The clients become aware of the change through advertisements sent out by the CDS servers.

Change or add cached server definitions

If clients are not in the same LAN as a CDS server, they do not receive any advertisements from CDS servers because broadcasts do not go across any IP routers. If a CDS server is moved, you must check whether any clients need to change or if the cached server definitions need to be added.

Let's assume we have moved a CDS server from *ev2* to *ev1*, and *ev4* is connected to them via X.25. To enable *ev4* to find the new CDS location, we need to run the following commands on *ev4*:

```
cdscp show cached server ev*    (optional; it lists all entries)
cdscp clear cached server ev2
cdscp define cached server ev1 tower ncacn_ip_tcp:9.3.1.68
```

When you first configure a DCE node with `mkdce -c`, the cached server entry is put into the /etc/rc.dce file. When you change this definition or add a new one, you must update your /etc/rc.dce file. Look for the line that assigns a value to the variable CACHE_SRV and update it:

```
CACHE_SRV=″cdscp define cached server ev1 tower ncacn_ip_tcp:9.3.1.68″
```

The `create_cds_entry` command that comes with this book can be used to create a cached server entry. See in 6.3.6, “Managing Caches on Client Machines” on page 181.

Listing Directories: The IBM-provided option is `cdsli -rd`, which lists all directories in the namespace. With one of the following OSF-provided commands, you can list the directories that are present in a certain clearinghouse:

```
# dcecp -c clearinghouse show /.:/ev7_ch | grep Dir_Name
# dcecp -c clearinghouse show /.:/ev7_ch | grep Dir_Name | cut -f2 -d'{' \
  | cut -f2 -d' '| sed "s/}//g"
old # cdscp show object /.:/ev1_ch | grep Name | cut -f2 -d=
```

Finding all Replicas and the Master: The following commands look up a certain directory and print which clearinghouses contain the master and read-only copies of that directory:

```
# dcecp -c directory show /.:/hosts | egrep "CH_Name|Replica_T"
old # cdscp show directory /.:/hosts CDS_Replicas | sed /Tower/d | sed /UUID/d
```

Finding the Replica Type in a Specific Clearinghouse: You can answer the question by looking at the output of the `cdscp show directory` command as shown above. If you want to include that query into a shell script, the following commands provide a more direct method:

```
# dcecp -c directory show /.:/hosts -clearinghouse /.:/ev1_ch | grep caType
{CDS_ReplicaType Master}

old # cdscp show replica /.:/hosts clearinghouse /.:/ev1_ch  | grep caType
        CDS_ReplicaType = master
```

Finding the Location of Replicas in the Cell: Probably the most useful command for getting a picture of the distributed structure of the name space is the `list_ch` command that we provide on the diskette in the back of this book:

```
# list_ch
```

It lists all directories and for each of them, the location (clearinghouses) of the master and all the read-only replicas. Also on the diskette is a `show_cds` Tcl script that runs on AIX and OS/2 and does the same as the `list_ch` command.

```
[C:] dcecp show_cds
```

Relocating the Master Replica for a Directory: As mentioned earlier, each occurrence of the same directory is called a replica. One of them must be defined as the master. In order to relocate an existing master directory, for instance /.:/hosts, you must have a read-only replica of that directory. Check which replicas are defined and where they are.

If you want the master to be in a new clearinghouse, create one, for example /.:/xyz_ch, in one of the two following ways:

1. Install a secondary CDS server, which automatically creates a clearinghouse:

   ```
   mkdce [-n <cell_name> -s <sec_srv_name>] cds_second
   ```

2. Create another clearinghouse on any CDS server system:

   ```
   # dcecp -c clearinghouse create /.:/xyz_ch
   old # cdscp create clearinghouse /.:/xyz_ch
   ```

   If this command fails, refresh the local CDS clerk cache with the command below, and then try again.

   ```
   # <tool_dir>/cleanup_cds_cache
   ```

See Appendix A, “Installing the Tools” on page 303, and replace <tool_dir> in the above command with the directory name you choose for the tools, which is /dce_tools according to the instructions.

If you want the master to be in a clearinghouse that does not have a read-only replica of that directory, create a replica:

```
# dcecp -c directory create /.:/hosts -replica -clearinghouse /.:/xyz_ch
old # dcecp create replica /.:/hosts clearinghouse /.:/xyz_ch
```

The actual relocation is done through redefining the replica set. You must specify which replica will become the master and which will be read-only. For this command, you have to specify *all* existing replicas of a directory. Assume the master was on *ev1,* a read-only was on *ev2*, and you want the master to be on /.:/xyz_ch. Use the following command to achieve this:

```
# cdscp set directory /.:/hosts to new epoch master /.:/xyz_ch
readonly /.:/ev1_ch /.:/ev2_ch
```

There is no equivalent dcecp command for this task.

Relocating a Clearinghouse by Transferring the Database Files: The steps described hereafter back up the whole clearinghouse and restore it to another *existing* CDS server. If the target system does not have a CDS server installed, you must first install one. Assuming you want to relocate the clearinghouse on *ev1* to *ev2*, follow these steps:

1. Install a CDS server on *ev2* if it is not already there:

   ```
   mkdce -n <cell_name> -s <sec_srv_name> cds_second
   ```

2. Disconnect the clearinghouse from the server where it is currently running:

   ```
   # dcecp -c clearinghouse disable /.:/ev1_ch
   old # cdscp clear clearinghouse /.:/ev1_ch
   ```

   This command must be run on *ev1*. It updates the clearinghouse disk files and ensures the files are consistent.

3. Backup the ev1_ch clearinghouse files on *ev1*:

   ```
   # tar -cvf/dev/rmt0 /opt/dcelocal/var/directory/cds/*ev1_ch.*
   /opt/dcelocal/etc/cds_attributes
   ```

4. Transfer the clearinghouse database files from their current location (source server system) to their new location (target server system).

5. Perform the next steps on the target system (*ev2*):

   a. Restore the files.

   b. Create a new clearinghouse:

      ```
      # dcecp -c clearinghouse create /.:/ev1_ch
      old # cdscp create clearinghouse /.:/ev1_ch
      ```

      You must use the same clearinghouse name as used on the source server system from which you copied the database files. If this command detects the copied clearinghouse database, it will reinitialize it to be used on your new server.

   c. This step is optional; it explains what to do in an error situation that used to occur in DCE 1.3 but is fixed in DCE 2.1.

      Other than in the previous DCE version, the IP address (Tower or binding information) of the new location is correctly updated in all objects now.

To see which elements in CDS still contain the old address (9.3.1.68), you can run a cleanif command:

```
# cleanif -i 9.3.1.68 -n 9.3.1.120 -f out.rpc | tee logfile
 ...
Checking cds object /.:/ev1_ch ... skipped.
Checking cds object /.:/hosts/ev1/config ... used.
 ...
```

Objects that indicate *used*, still contain the old address. Of course, entries that point to a still-existing service on ev1 *must* display used. Only if you detected an object that should have been changed would you have to manually change the IP address. In the previous release, the address was not updated in the clearinghouse object itself and in all the directory entries. In such a case, you would edit the generated out.rpc file, delete to good entries and leave the bad entries that need a change. Then you would run the cleanup_ip command to perform these changes. See 6.2.3, “Changing IP Addresses” on page 148 for information on the commands.

d. Refresh the CDS clerk cache:

```
# <tools_dir>/cleanup_cds_cache
```

e. The clearinghouse is now on system *ev2.* If you had to install a new CDS server (mkdce cds_second) on ev2 for the purpose of restoring the clearinghouse, a /.:/ev2_ch clearinghouse was created for you. You might want to delete it and keep only /.:/ev1_ch.

```
# dcecp -c clearinghouse delete /.:/ev2_ch
old # cdscp delete clearinghouse /.:/ev2_ch
```

6. If this was the only clearinghouse on the old location *ev1*, you might want to remove the CDS server from the cell with the following steps:

a. If *ev1* is the initial CDS server:

The initial CDS server only distinguishes itself from secondary CDS servers in that it hosts the master copy of /.: (root directory). Since you have moved the master replica of the root directory, the roles of Initial CDS Server and Secondary CDS Server were switched between the two systems.

To reflect this fact in your cell configuration, you must edit the /opt/dcelocal/etc/mkdce.data file on *ev1* and *ev2* and switch the two following lines between these systems:

```
cds_srv     COMPLETE   Initial CDS Server
cds_second  COMPLETE   Additional CDS Server
```

b. Refresh the CDS clerk cache on *ev1*:

```
# <tools_dir>/cleanup_cds_cache
```

c. Remove the CDS server locally from *ev1*:

```
rmdce cds_second
```

7. Refresh the CDS clerk caches on all systems now:

```
# <tools_dir>/cleanup_cds_cache
```

Relocating a CDS Server by Merging Clearinghouses: In the above-mentioned procedure of relocating a clearinghouse, you get a clearinghouse (ev1_ch) on the new site (ev2) that is named after the old location. That prevents you from regularly installing a secondary CDS server on the old system (ev1) in the future.

Also, if you already had a clearinghouse on the target system, you would have two clearinghouses after the relocation.

The challenge to overcome these problems is to merge the contents of ev1_ch into ev2_ch that might already contain a number of master or read-only replicas, and then delete ev1_ch. The improved copy_ch command provided on the diskette that comes with this book allows you to do that. Here are the steps to follow:

1. In order to be able to track the relocation, list the namespace now. Run the following command on a DCE 2.1 machine:

   ```
   # list_ch | tee cds_struct_before
   ```

2. Move the contents of the source to the target clearinghouse:

   ```
   # copy_ch -m -s ev1_ch -t ev2_ch | tee logfile
   ```

   This command creates a replica in the target clearinghouse for every replica it finds in the source clearinghouse. It correctly redefines the replica set no matter how many replicas of directories exist or how many clearinghouses there are. The -m flag makes it establish the master in the target clearinghouse for every master replica currently found in the source clearinghouse.

3. Skip this step if you intend to delete the whole CDS server on ev1. If you just want to delete the source clearinghouse, which is now purely read-only, do the following:

   ```
   # dcecp -c clearinghouse delete /.:/ev1_ch
   ```

   Although all replicas and the clearinghouse are removed, the command may end with an error message and leave a /.:/ev1_ch entry in the namespace, which is just an empty pointer. You can delete it with the following command:

   ```
   # dcecp -c object delete /.:/ev1_ch
   ```

   Should the deletion of the clearinghouse fail for any other reason, you need to check the /.:/ev1_ch clearinghouse for leftover master replica entries. At any time you can run the list_ch command to check the replication structure of the whole namespace.

4. Remove the CDS server on ev1.

   Once the old clearinghouse /.:/ev1_ch contains only read-only replicas, it can be deleted. This is done by the rmdce command.

   If this server was the initial CDS server, we must edit the /opt/dcelocal/etc/mkdce.data file on both sites and switch the two following lines between the two nodes:

   ```
   cds_srv     COMPLETE   Initial CDS Server
   cds_second  COMPLETE   Additional CDS Server
   ```

   This is described above in “Relocating a Clearinghouse by Transferring the Database Files” on page 162. Then the secondary CDS server can be removed with:

   ```
   rmdce cds_second
   ```

Note: The list_ch and copy_ch scripts used here run on AIX 4.1.3+ only. They use dcecp commands. However, while running on an AIX 4.1.3+ system, copy_ch can move a clearinghouse on any platform in the cell, regardless of its operating system (AIX, MVS, OS/2).

6.2.4.4 Relocating the Primary Security Server

This section contains instructions on how to relocate the primary security server, its registry database, and associated files to another machine in the cell. The target machine may be an existing client or a security secondary machine. This procedure may also be used to relocate a security server to a different platform or to migrate from OSF DCE 1.0.3 to OSF DCE 1.1.

Here are the steps you can follow to move the master security server from *ev3* to *ev1*:

1. Configure a secondary DCE security server on the target system (*ev1*) which is already configured as a DCE client. Run the following command on *ev1*:

   ```
   # mkdce -R -r ev1 sec_srv
   ```

2. Stop the security server on the source machine (ev3) by running the following command:

   ```
   # dce.clean secd
   ```

3. On *ev1*, log in to DCE as *cell_admin*.

4. Designate the previously defined security replica to be the new master:

   ```
   # dcecp
   dcecp> registry designate -master /.:/subsys/dce/sec/ev1
   ```

   This could be done from any DCE 2.1-configured machine in the cell.

5. Using your favorite editor, edit the */opt/dcelocal/etc/security/pe_site* file on both machines (*ev1* and *ev3*), and enter the correct addresses.

6. If you move your master registry from OSF DCE 1.0.3 to OSF DCE 1.1 and you would like to have full DCE 1.1 support, you must now modify your registry version. With the `registry show` subcommand of dcecp, you get something similar to the following:

   ```
   dcecp> registry show
   {deftktlife +0-10:00:00.000I-----}
   {hidepwd yes}
   {maxuid 2147483647}
   {mingid 100}
   {minorgid 100}
   {mintktlife +0-00:05:00.000I-----}
   {minuid 100}
   {version secd.dce.1.0.2}
   ```

   The last line shows the current version of your registry. To upgrade to the OSF DCE 1.1 supported version, enter the command:

   ```
   dcecp> registry modify -version secd.dce.1.1
   ```

7. Unconfigure the original master security server (*ev3)*:

   a. Stop and restart DCE now on your original security master server.

      ```
      # dce.clean
      # rc.dce
      ```

   b. Unconfigure the DCE security server now on the original master server.

      ```
      # rmdce sec_srv
      ```

8. From *ev1*, delete the entries for the old master from CDS and the replica list:

```
# dcecp -c rpcgroup remove -member /.:/subsys/dce/sec/master /.:/sec
# dcecp -c rpcgroup remove -member /.:/subsys/dce/sec/master /.:/sec-v1
# dcecp -c object delete /.:/subsys/dce/sec/master
# dcecp -c registry delete /.:/subsys/dce/sec/master -force
```

As long as these entries exist, all DCE clients would be stuck at this point. They still get answers from the new security server, but communication with CDS fails with a message that the Key Distribution Center cannot be reached.

9. Restart DCE also on the new security server (on *ev1*):

```
# dce.clean
# rc.dce
```

10. Change all *pe_site* files on systems in the cell.

    If you have not already changed the *pe_site* files of your clients, you must do it now. Change the IP address found in the */opt/dcelocal/etc/security/pe_site* file to the one assigned for the new master security server. In DCE 2.1, the pe_site file can be remotely managed as a dced object (see 6.7.5, “Working with the Hostdata” on page 235).

11. If time-outs are experienced on the DCE clients now, a refresh of the their CDS cache might help:

```
# dcecp -c rpcgroup list /.:/sec
/.../test/subsys/dce/sec/ev1
# dcecp -c rpcgroup list /.:/sec-v1
/.../test/subsys/dce/sec/ev1
```

Note: There is no longer a security server named *master*. However, some other programs, such as the DFS configuration, rely on the presence of the original master server name. Therefore, we recommend creating a secondary security server with the following command:

```
# mkdce -R -r master sec_srv
```

6.3 Backup/Restore and Other Housekeeping Tasks

As with any operating system, application program, and data, a failure of hardware or software or a mistaken administrator operation can jeopardize the functionality of the service.

All DCE and DFS services provide replication of their databases, which can be considered an on-line backup. However, replication is not sufficient, and traditional backups are still needed or at least recommended in cases such as:

- System time accidentally set backwards
- Release upgrades
- Attempt to change the cell name
- Unsuccessful change of an IP address
- Accidental removal of database files, namespace entries, or DCE accounts

A DCE cell consists of different types of data. Some data are never changed; some are changed from time to time, and others are highly dynamic. In order to recover a system after an accident, we need to make proper backups. Since the implementation of certain configuration files are platform-specific, most of the following approaches are only valid for AIX and may slightly vary on other platforms.

The following sections cover:

1. Full system backup
2. Backup/restore of DCE core services databases
3. Backup/restore of DFS databases
4. Backup/restore of DFS data
5. Controlling system-created files that may fill disk space
6. Managing caches

6.3.1 Full System Backup

AIX offers a very efficient method to make a clean, bootable full system backup of a machine. It is done with the `mksysb` command or through SMIT with the `smitty mksysb` command.

You are advised to perform an `mksysb` frequently or at least every time you change something in your system configuration or install new applications or PTFs.

In order to backup your DCE configuration properly with `mksysb`, you must shut down DCE first with the `dce.clean all` command. However, because `mksysb` may take between half an hour and two hours, depending on the size of your server, it may not be convenient for you to make *full* backups too frequently. Because DCE has dynamic data which need to be backed up frequently, we recommend you use another backup method in addition to the occasional *mksysb*.

The same is, of course, true for OS/2. Every time you run an OS/2 backup which also saves the OS/2 DCE data, you must stop DCE first!

6.3.2 Backing Up DCE-Core-Services-Related Information

This step focuses on the DCE data that is necessary for the DCE core services to run as configured. The subsequent sections give an overview of the files that need to be backed up and show different ways to back up the core services.

6.3.2.1 Important Files

Below, we describe the DCE data you should back up frequently (we recommend once a day):

- */opt/dcelocal/var/dced*

  This is the directory where all the dced databases are stored. Some of the files are never changed; some are highly dynamic. The *.db files together build the dced database, whereas the other ones are carrying configuration information only. When dced starts the first time, it initializes all necessary *.db files. All *.db may be recreated from scratch when using the `dced -i` command. It initializes the dced databases and ACLs and exits. If the databases exist, this option displays an error.

  The *Ep.db* file contains the local RPC endpoint maps. This is the endpoint map that is stored on disk so that the DCE daemon (formerly *rpcd*) can be stopped and restarted without requiring servers to reregister with the DCE daemon. After a system reboot, RPC-based servers restart and reregister with the endpoint map service; so the database file needs to be deleted before the DCE daemon starts. This is done by the */etc/rc.dce* script, which starts up all the DCE processes.

- */opt/dcelocal/var/security*

  This directory stores all the security-relevant information.

- */opt/dcelocal/var/directory/cds*

  This directory contains all the clearinghouse-specific information.

- /opt/dcelocal/etc/cds_attributes

  This file stores the CDS-specific attributes.

The exact same files are also used on OS/2. However, the path names contain the backslash () instead of the forward slash (/), and the *cds_attributes* file is *cds_attr* because it can only be eight characters long.

6.3.2.2 Configuring the DCE Backup Server

DCE Version 1.1 includes a very nice backup tool within the dcecp Tcl environment. Once it is configured, you simply enter the `cell backup` subcommand of dcecp, which starts a `tar` backup of your important security and/or CDS server data. It backs up the master security database and each clearinghouse with master replicas in the cell. This operation requires that a dced is running on each of the server hosts being backed up. It ***does not*** take care of the DCE host daemon (dced) data.

Unfortunately, the dcecp Tcl scripts were not readily ported to OS/2 when we tested this. The `tar` command that the cell backup uses is not available on OS/2. However, since `dcecp> cell backup` is the OSF proposed backup method, it must eventually be platform-independent. So, if you run DCE core servers on OS/2, we suggest you use this method as soon as it is available.

To enable the use of the cell backup, the cell must be prepared by setting up a backup destination (*bckp_dest*) Extended Registry Attribute (ERA). Its value specifies a file that needs to be *accessible through the local file tree* of the respective server. It is typically a tape archive (disk file or tape device file). It can be a DFS file if a DFS client is configured.

Add the new ERA to the principals for the master DCE security registry database and all CDS clearinghouses with master replicas that you want to back up. To do this, follow these steps:

1. Create an ERA as a string that specifies a backup destination. Name the ERA */.:/sec/xattrschema/bckp_dest,* and set the type to *printstring*. Select the *principal* ACL manager, and set its four permission bits to r (read), m (manage), r (read), and D (Delete) as shown in the following command:

   ```
   dcecp> xattrschema create /.:/sec/xattrschema/bckp_dest
   -encoding printstring > -aclmgr {principal r m r D}
   ```

2. Add the new ERA (*bckp_dest*) to the principal, *dce-rgy,* that represents the DCE Security Service registry database. Set the value to be the tar file name or the device that is the backup destination:

   ```
   dcecp> principal modify dce-rgy -add {bckp_dest tarfilename_or_device}
   dcecp>
   ```

3. Add the new ERA (*bckp_dest*) to all */.:/hosts/<hostname>/cds-server* principals (the CDS Servers). Set the value to be the tar file name or the device that is the backup destination

```
dcecp> principal modify /.:/hosts/hostname/cds-server
       -add {bckp_dest tarfilename_or_device}
dcecp>
```

Now, whenever you want to back up your registry database or CDS database, you need only invoke a `cell backup` operation. You can back up another cell by including the cell name as an argument to the `cell backup` operation. Note that you need the necessary permissions in the remote cell.

The `dcecp> cell backup` method does not take care of your */opt/dcelocal/var/dced* files. These have to be manually backed up using the `tar` command. You can also add this backup to the existing backup Tcl script. If you chose to add the backup of /opt/dcelocal/var/dced to the backup Tcl scripts of dcecp, you must store it in another directory; otherwise, installing a PTF or another DCE release may remove your changes.

The privileges required to run the cell backup command are:

- Local superuser (*root*)
- DCE cell administrator (*cell_admin*)

This new backup only works on systems that run OSF DCE 1.1 or AIX DCE 2.1. If you have servers on an older DCE level, you must still perform a traditional file-based backup for every service on that platform. Following, we describe the common backup methods for mixed DCE environments.

6.3.2.3 Backing Up the Files of the DCE Daemon

Besides the databases that it can re-create by itself, the DCE daemon also stores user-defined information, such as the keytab objects. The should be backed up with the following commands:

1. Log in at the machine running the DCE daemon.
2. Stop the DCE daemon:

   ```
   # /etc/dce.clean dced
   ```

3. Back up all dced files:

   ```
   # tar -cvf/dev/rmt0 /opt/dcelocal/var/dced
   ```

4. Restart the DCE daemon:

   ```
   # /etc/rc.dce dced
   ```

On OS/2 use `dcestop` and `dcestart` commands and back up the same directory.

6.3.2.4 Restoring the Files of the DCE Daemon

Make sure the DCE daemon is not running and restore all the files. If the system was rebooted, the Ep.db file can be deleted. Then start the DCE daemon.

6.3.2.5 Backing Up the Files of the Security Server

Please Note

The following backup methods have been used and tested with AIX only. The same is also true for the backup tools.

The following steps need to be performed on the master security server. There is no need or benefit in saving a replica server's database.

1. Log in at the master security server site as cell_admin.
2. Put the security registry database in read-only mode, which causes the in-memory copy to be saved to disk.

   ```
   # sec_admin
   sec_admin> state -maintenance
   ```

   or stop secd (dce.clean sec_srv)
3. Back up the associated files, for example:

   ```
   # tar -cvf/dev/rmt0 /opt/dcelocal/var/security/.mkey
   /opt/dcelocal/var/security/rgy_data
   ```

   If you use the cp command, be sure to save the master key file, .mkey, in the same directory, too. The master key is used to encrypt the registry database, and if the key is lost, the backup is useless.
4. Reactivate the security server.

   ```
   # sec_admin
   sec_admin> state -service
   ```

   If stopped, restart it with the rc.dce sec_srv command.

6.3.2.6 Restoring the Files of the Security Server

Follow these steps to restore the security server:

1. Log in at the master security server site as cell_admin.
2. Stop the security server:

   ```
   # dce.clean sec
   ```
3. Restore the associated files:

   ```
   # tar -xvf/dev/rmt0
   ```
4. Restart the security server:

   ```
   # rc.dce sec
   ```
5. Check the state of the security server, and put it in the service state if it is still in maintenance mode:

   ```
   # sec_admin
   sec_admin> info
   sec_admin> state -service
   ```

6.3.2.7 Backing Up the Files of a CDS Server

Please Note

The following backup methods have been used and tested with AIX only. The same is also true for the backup tools.

The CDS namespace is divided into one or several clearinghouses. This section looks at how to back up the files associated with one specific clearinghouse. We tested two different cases:

1. The first case represents the situation where CDS can be suspended for copying the CDS files from that specific server. This is the most general way that works even if the namespace is distributed and replicated.

2. The second case considers keeping the name service active by creating a copy for the clearinghouse to be backed up. If uninterrupted write access to the clearinghouse is required, you can back up the read-only clearinghouse.

Backing Up by Disabling the Service: This is the normal procedure as suggested in the DCE product documentation. It will work no matter how distributed and replicated your CDS is.

If your CDS is distributed, you have to back up all clearinghouses that contain one or more master replicas of any directory. Back up all these clearinghouses at the same time as much as possible, and try not to change the replica set of any directory until you have backed up all clearinghouses. This means that you should not make new replicas or change a read-only into a master in the meantime. Otherwise, you might have to do some extra work to re-create consistency upon restore.

1. Display the whole CDS replication structure and save it in a file:

   ```
   # list_ch | tee /tmp/cds_structure_062096
   ```

   The list_ch shell script, issued from an AIX 4.1.3+ machine, shows you where the master and read-only replicas of all directories are. See also 6.2.4.3, “Moving CDS Resources” on page 160, for other helpful commands for finding master and read-only replicas.

2. Log in as cell_admin on the CDS server you want to back up.
3. Disable the clearinghouse you want to back up, for instance /.:/ev1_ch, with one of the two following methods:

   - Disabling the CDS server, which would affect all clearinghouses on *ev1:*

     ```
     # dce.clean cds
     ```

     The same is achieved with cdscp disable clerk followed by cdscp disable server.

   - Disconnecting a specific clearinghouse from the CDS server:

     ```
     # dcecp -c clearinghouse disable /.:/ev1_ch
     old # cdscp clear clearinghouse ev1_ch
     ```

     This updates the clearinghouse files and ensures the files are consistent. However, this method takes a long time to reconfigure the clearinghouse.

4. Back up the files associated with the clearinghouse(s) on *ev1*.

   If you back up *all* clearinghouses on ev1, issue the following command:

   ```
   # tar -cvf/dev/rmt0 /opt/dcelocal/var/directory/cds
   /opt/dcelocal/etc/cds_attributes /opt/dcelocal/etc/cds.conf
   ```

   If you back up a particular clearinghouse, do the following:

   ```
   # tar -cvf/dev/rmt0 /opt/dcelocal/var/directory/cds/*ev_1ch.*
   /opt/dcelocal/var/directory/cds/cds_files \
   /opt/dcelocal/etc/cds_attributes /opt/dcelocal/etc/cds.conf
   ```

   Remember to also back up the file you created before with the list_ch command.

5. Reactivate the clearinghouse according to the method by which you had stopped activities before:

   - Restarting the CDS server:

     ```
     # /etc/rc.dce cds
     ```

Or with OSF commands:

```
# cdsadv
# cdsd
```

- Reconnecting the clearinghouse with the CDS server

```
# dcecp -c clearinghouse create /.:/ev1_ch
old # cdscp create clearinghouse /.:/ev1_ch
```

Keeping a Clearinghouse Active: If you cannot afford to disable the clearinghouse, you can copy it to a temporary clearinghouse. While access is possible to that temporary clearinghouse, you can back up the original clearinghouse. If you establish the master replicas contained in the original clearinghouse in the temporary clearinghouse, you could even provide uninterrupted write access. However, if you back up the clearinghouse information while an extra clearinghouse exists, the restore of the backups will be more complex, because you need to define the replica set for each directory, thereby explicitly excluding the temporary clearinghouse.

Which method should you use? Introducing the temporary clearinghouse requires extra, manual steps to re-create the correct replica set because while the backup is made, some information concerning this temporary clearinghouse is stored along with the data you actually want to conserve. Test how long the previously described method takes and whether you can accept the unavailability of CDS for such an amount of time. Remember that CDS clients are caching information and do not need to access a server for every look-up request.

If you need to enable continuous CDS access, perform the following steps:

1. Create a temporary clearinghouse, /.:/xxx_ch, on the same CDS server:

   ```
   # dcecp -c clearinghouse create /.:/xxx_ch
   old # cdscp create clearinghouse /.:/xxx_ch
   ```

2. Copy the contents of /.:/ev1_ch to /.:/xxx_ch:

   ```
   # copy_ch -s ev1_ch -t xxx_ch
   ```

   If you want continuous read/write access to /.:/xxx_ch, use the `-m` flag to move the master role for every directory to /.:/xxx_ch if present in /.:/ev1_ch:

   ```
   # copy_ch -m -s ev1_ch -t xxx_ch
   ```

3. Back up /.:/ev1_ch as described above in “Backing Up by Disabling the Service” on page 171.

4. If you specified the `-m` flag above, copy the clearinghouse back to /.:/ev1_ch, again using the `-m` option.

5. Once /.:/ev1_ch is backed up and running, the temporary clearinghouse can be deleted on the machine that is housing it:

   ```
   # dcecp -c clearinghouse delete /.:/xxx_ch
   old # cdscp delete clearinghouse /.:/xxx_ch
   ```

6.3.2.8 Restoring the Files of a CDS Server

To restore any of the clearinghouses, perform the following steps:

1. Stop CDS.
2. Restore the database(s).
3. Edit the list of existing clearinghouses to make sure it reflects exactly the valid clearinghouse files:

   ```
   # vi /opt/dcelocal/var/directory/cds/cds_files
   ```

   Especially, remove from the file any temporary clearinghouses that you used for an uninterrupted-access backup. The CDS server would try to activate them and hang.
4. Restart CDS.
5. If you defined a temporary clearinghouse to maintain an uninterrupted CDS access during the backup, use the CDS structure file you created with the `list_ch` command before the backup and redefine each replica set according to that structure file, thereby excluding the temporary clearinghouse.

   Note that in some cases it may seem unnecessary to run the `cdscp set dir to new epoch master` command and that CDS works correctly without excluding the temporary clearinghouse. However, the knowledge of this dummy clearinghouse is still there, and it may cause failures later on, for example, when you define an additional, new replica and want to redefine the corresponding replica set.
6. Synchronize every directory:

   ```
   # for DIR in `cdsli -rd`; do
   > echo "$DIR \c"
   > dcecp -c directory synchronize $DIR
   > done
   ```

   If you get errors on a directory because of changes that have been made since the backup, you need to decide which version of the directory you want to keep and redefine the replica set (a new epoch master) accordingly. Execute a `cdscp set dir to new epoch master` command, specifying the clearinghouse with the good replica of the directory as the master and *all* other existing replicas as `readonly` or `exclude`. In this way you could even designate a read-only replica as the new master if the original master is unrecoverable.
7. If you want to avoid time-outs on the clients, clean all clerk caches with `cleanup_cds_cache` (see 6.3.6, “Managing Caches on Client Machines” on page 181)

6.3.3 Backing Up DFS-Servers-Related Information

There are two types of entities that need backup considerations. The first consists of the two DFS databases, which are the fileset location database (FLDB) and the backup configuration database. The fileset data builds the second type, which is discussed in 6.3.4, “Backing Up and Restoring DFS Data” on page 177.

DFS provides a sophisticated backup subsystem that allows you to create full or incremental backups of filesets or aggregates. The backup database defines backup families, backup schedules, location of machines with one or multiple tape devices (tape coordinators), and which families use which tape coordinator.

The backup database is a distributed database that uses the same ubik algorithm to synchronize among its server processes, like the FLDB. The backup database can be backed up within the backup subsystem with bak savedb. For more details about the backup subsystem or the bak command suite, consult *The Distributed File System (DFS) for AIX/6000* redbook.

In the following subsections, we want to provide a short recapitulation on how the FLDB is backed up and restored.

6.3.3.1 Backing up the FLDB

The FLDB is a distributed database that stores fileset locations. Multiple FLDB servers are running in a cell, each of them controlling a database. Update requests go through a set of calls belonging to a library named ubik. The ubik routines designate one master server and update that database. The other servers, if there are any, become slaves, and their database is automatically updated by the ubik routines.

The set of information stored for each fileset in the FLDB can be created from information stored on the fileset itself, the fileset header. Under normal circumstances this redundant information is consistent or synchronized. The fts command suite offers options to synchronize FLDB and fileset header information in both directions.

The chances of losing the entire FLDB are minimal if this service is replicated as recommended. Furthermore, most of the FLDB information can be recreated from all fileset headers. Exceptions are non-LFS filesets, which do not have fileset headers, and replication information.

To be prepared for the worst, it might make sense to back up the FLDB in regular intervals, especially if you are using a lot of fileset replication or non-LFS filesets.

1. Keep the status of the FLDB database in a readable file for information:

   ```
   # date > /tmp/BACKUP/dfs/ls_fldb_output
   # fts lsfldb >> /tmp/BACKUP/dfs/ls_fldb_output
   ```

2. Stop the FLDB (flserver) service via the bos command suite:

   ```
   # /opt/dcelocal/bin/bos status /.:/hosts/ev8 -localauth
   Instance upserver, currently running normally.
   Instance flserver, currently running normally.
   Instance ftserver, currently running normally.
   # /opt/dcelocal/bin/bos stop /.:/hosts/ev8 flserver -localauth
   # /opt/dcelocal/bin/bos status /.:/hosts/ev8 -localauth
   Instance upserver, currently running normally.
   Instance flserver, disabled, currently shutdown.
   Instance ftserver, currently running normally.
   ```

3. Copy the FLDB files to the backup media:

   ```
   # di /opt/dcelocal/var/dfs/fldb*
   -rw-------   1 root  145472 Jun 10 14:24 /opt/dcelocal/var/dfs/fldb.DB0
   -rw-------   1 root      64 Jun 10 14:24 /opt/dcelocal/var/dfs/fldb.DBSYS1
   # cp  -v /opt/dcelocal/var/dfs/fldb* /tmp/BACKUP/dfs
   ```

4. Restart the FLDB service via the bos command suite:

```
# ps -ef | grep flserver
    root 11945  8382   4 14:47:13  pts/2  0:00 grep flserver
# /opt/dcelocal/bin/bos start /.:/hosts/ev8 flserver -localauth
# ps -ef | grep flserver
    root  6829  8382   3 14:47:40  pts/2  0:00 grep flserver
    root 11947 10700  35 14:47:35      -  0:00 /opt/dcelocal/bin/flserver
```

From this file, in case of reloading, it will be possible to repopulate a new FLDB server and from there to synchronize the cell file servers (ftserver processes) of the cell, if needed.

During this suspension of the FLDB activity, the DFS clients currently in communication with DFS servers are not affected. Only the new requests for the location of servers are delayed.

6.3.3.2 Restoring an FLDB

Please Note

The steps outlined here are a summary of ideas based on our experiences in this project and were not thoroughly tested due to lack of time. However, we wanted to include them to give you ideas on how to tackle this important issue.

FLDB databases have version numbers assigned to them. If any problem occurs and the ubik routines that manage the FLDB databases have elected a new synchronization site, this site checks all other FLDB servers and copies the database with the highest version number to its own machine. This will eventually become the master database from which the others are updated.

So if we want to restore an FLDB database, we have two problems:

1. We cannot predict which server will be elected to become the master.
2. Our backup might have a lower version number and therefore be overwritten by an existing one.

Therefore, we recommend trying one of the two following procedures. However, before you try this, be sure the effort to synchronize from the fileset headers is really too big and the backup of the FLDB is not outdated for too long a time.

Restoring the FLDB on All Servers: Try this procedure first:

1. Stop DFS on all servers with `dfs.clean`.
2. Restore the FLDB files on *all* flserver machines.
3. Start DFS on all servers with `rc.dfs`.

Restoring the FLDB on One Server by Removing the Others: If the first procedure does not work:

1. Remove all FLDB servers but the one on which you made the backup as outlined in 6.3.3.1, “Backing up the FLDB” on page 174; use `rmdfs dfs_fldb`, if this is still possible.
2. If this is not possible, try a local unconfiguration and manually delete the CDS entries for these servers:
   - `rmdfs -l dfs_fldb`
   - Delete the CDS entries of the form /.:/host/ev1/flserver.

- Remove the RPC group members from /.:/fs.

3. Restore the FLDB files on the one flserver machine that has been left over.
4. Start the flserver. If this does not work, remove and reconfigure the FLDB server, restore the files once again, and restart the flserver.
5. Reconfigure the other FLDB servers.

6.3.3.3 Recreating an FLDB

If the files of the FLDB servers are corrupted or accidentally destroyed, the flserver can be started anyway, but it produces the following error message:

```
# fts lsfldb -localauth
Could not access the FLDB for attributes
Error: FLDB: cannot create FLDB with read-only operation (dfs / vls)
```

By using the fts syncfldb for each fileset server in the cell, the needed entries are repopulated from information on each server. Here are the required steps when starting from an empty FLDB.

1. Establish the list of servers; for that purpose, it is a good habit to periodically list the contents of the FLDB into a file with fts lsfldb.
2. Create the server entries:

   The fts crserverentry command has to be executed for each server that has filesets which need to be known in the FLDB. The command requires the knowledge of the appropriate principal for getting the information from the server. This will be given by bos lsadmin.

   Example for server ev8:

   ```
   # /opt/dcelocal/bin/bos lsadmin /.:/hosts/ev8 admin.fl -localauth
   Admin Users are: user: hosts/ev8/dfs-server, group: subsys/dce/dfs-admin
   # fts crserverentry /.:/hosts/ev8 hosts/ev8/dfs-server -localauth
   # fts lsserverentry -all -localauth
   9.3.1.127 (2:0.0.9.3.1.127)
   FLDB quota: 0;  uses: 0;  principal='hosts/ev8/dfs-server';  owner=<nil>
   ```

3. Populate the FLDB database with the fileset entries from all the servers:

   ```
   # fts syncfldb -server /.:/hosts/ev8
   number of sites: 1
      server           flags     aggr   siteAge principal      owner
   9.3.1.127           RW       lvs.root 0:00:00 hosts/ev8/dfs-server<nil>

   -- done processing entry 6 of total 9 --
   Creating an entry for fileset 0,,7 in FLDB
           readWrite   ID 0,,7  valid
           readOnly    ID 0,,8  invalid
           backup      ID 0,,9  invalid
   number of sites: 1
      server           flags     aggr   siteAge principal      owner

   9.3.1.127           RW       lfsroot 0:00:00 hosts/ev8/dfs-server<nil>

   -- done processing entry 7 of total 9 --
   Creating an entry for fileset 0,,4 in FLDB
           readWrite   ID 0,,4  valid
           readOnly    ID 0,,5  invalid
           backup      ID 0,,6  invalid
   number of sites: 1
   ```

```
   server           flags     aggr   siteAge principal      owner

9.3.1.127           RW       lfsroot 0:00:00 hosts/ev8/dfs-server<nil>

-- done processing entry 8 of total 9 --
Creating an entry for fileset 0,,1 in FLDB
        readWrite   ID 0,,1  valid
        readOnly    ID 0,,2  invalid
        backup      ID 0,,3  invalid
number of sites: 1
   server           flags     aggr   siteAge principal      owner

9.3.1.127           RW       lfsroot 0:00:00 hosts/ev8/dfs-server<nil>

-- done processing entry 9 of total 9 --
FLDB synchronized with server /.:/hosts/ev8
```

The fts syncfldb command inspects filesets residing on the file server machine specified by -server. It checks either all of the filesets on -server or only the filesets on the optionally specified -aggregate. It then checks that the FLDB correctly records every fileset whose fileset header is marked on-line and changes any necessary FLDB entry to be consistent with the status of each fileset on the server.

This command also performs the following additional functions:

- If it finds a backup fileset whose read/write source no longer exists at the same site, it removes the backup from the site.
- If it finds a fileset ID number that is larger than the value of the counter used by the flserver when allocating fileset ID numbers, it records this ID number as the new value of the counter. The next fileset to be created receives a fileset ID number one greater than this number.
- If necessary, it increments or decrements the number of fileset entries recorded as residing on a file server machine in the FLDB entry for the server.

It is recommended that fts syncserv is run for all file server machines in a cell after fts syncfldb is run against all file server machines in the cell.

The fts syncfldb and fts syncserv commands cannot restore replication information that was lost when an entry for a DCE LFS fileset was removed from the FLDB. Replication information must be reconstructed with the fts setrepinfo and fts addsite commands.

You can use fts syncfldb against non-LFS filesets although they do not have fileset headers. If they are dfsexport'ed, syncfldb will create an entry called ″SYNCFLDB-GENERATED-0-<fileset ID>″. It makes up this name because it can't find a name from the header. The fts rename can then be used to rename the generated name back to the original name.

6.3.4 Backing Up and Restoring DFS Data

There are basically two methods to back up DFS data:

- Using the DFS backup subsystem with the bak command suite.

  As mentioned above in 6.3.3.1, “Backing up the FLDB” on page 174, the backup service allows for setting up sophisticated automated backup procedures. You define tape coordinator machines that have tape drives attached to them, and you define families of filesets and/or aggregates that

are going to be backed up together with the same schedules and to the same tape coordinator.

- Instantaneous backups with the `fts` command suite.

  The `fts dump` and `fts restore` commands let you create instantaneous backups to files or to media.

These are the only two options that preserve the DFS ACL information. All AIX backup/restore commands only consider owner and group IDs (UID/GID) and the UNIX mode bits derived from the user_obj, group_obj, mask_obj, and other_obj permissions.

For more information, consult *The Distributed File System (DFS) for AIX/6000* redbook or the AIX DFS documentation.

ADSM

The current release of the ADSTAR Distributed Storage Manager (ADSM) is not aware of DFS ACLs. Therefore, the only way to use ADSM and preserve the ACLs is to use the method described above as *Instantaneous Backup*.

This means you can create a backup file using the `fts dump` command and back up the backup file with ADSM.

6.3.5 Controlling Disk Space: System-Created Files

DCE creates a lot of files that it uses for its operation, such as:

- Database files
- Caches
- Credential files
- Socket special files

All these files can vary a lot in size and number. So they can use up all the disk space or i-nodes of a file system. If a file system becomes full for one of these reasons, the affected components will not work anymore, which may have an influence on the operation of other components.

DCE stores all these files underneath the /opt/dcelocal/var directory. In AIX this directory is a symbolic link to /var/dce in the /var file system. The /var file system is also used for other system files that vary in size and can grow very big, such as print spool or trace files. It is important to create separate file systems for DCE to decouple DCE from the operating system as advised in 3.2, “Preparing for DCE Configuration on AIX” on page 38.

6.3.5.1 Databases

Most DCE servers control a database which grows with the number of objects stored in it. The database files are in the following directories:

- Security server: /opt/dcelocal/var/security/rgy_data

  The files within this directory mainly grow with the number of principals and accounts at a rate of about 1 KB per user.

- CDS server: /opt/dcelocal/var/directory/cds

The database files mainly grow with the number of client workstations (approximately 1.4 KB per client) and also with the number of application servers.

- DCE daemon: /opt/dcelocal/var/dced

  The endpoint map file, *Ep.db,* grows with the number of application servers. The number of socket files in /opt/dcelocal/var/rpc/socket grows with the number of DCE client applications running on a server system, which corresponds to the number of concurrent local client/server connections.

- DFS FLDB: /opt/dcelocal/var/dfs

  The FLDB grows with the number of filesets. It may become large if there is a lot of users and each of them has their own fileset.

Observe these directories and increase the file system size(s) as necessary. See also the sizing guidelines in 2.3, “Sizing Guideline” on page 24 for information on disk space requirements.

6.3.5.2 Cache Files

Cache files can be found on every DCE client system. There are two caches in /opt/dcelocal/var/adm, and their size can be limited:

- CDS clerk cache: /opt/dcelocal/var/adm/directory/cds

  The size of the clerk cache can be limited if the cdsadv process is started with the -c flag.

- DFS cache: /opt/dcelocal/var/adm/dfs/cache

  This cache can be configured with the DFS client configuration or with the -block option to the dfsd command or within the file */opt/dcelocal/etc/CacheInfo*. It should not be set to a value larger than 85 percent of the logical volume size. For a new cache size to take effect, the system must be rebooted.

See also 6.3.6, “Managing Caches on Client Machines” on page 181, for more information about cache management.

6.3.5.3 Credential Files

A credential file is created, for instance, when a principal logs in. It contains all tickets that are granted to a principal as long as his Ticket Granting Ticket (TGT) is valid. The next time the same user logs in, he gets a new credential file. So the number of these files is increasing and may reach the maximum number of i-nodes defined for a file system or fill up the file system space. The rmxcred command should be used to remove stale credential files:

```
rmxcred -?
Usage: rmxcred [-h hours] [-d days] [-v] [-f | -p principal]

Default: all ticket caches totally expired are purged except
for the machine context, and except for those used by the
secd and cdsd programs. (To remove any of these explicitly,
specify -p and the name 'self', 'dce-rgy', or 'cds-server')

Use -d and -h options to only remove caches that have
been expired for the specified # of days or hours.  They can
be set separately or in combination.  Use -p to remove stale
caches for the specified principal only.  OK to specify xyz
for principals of form 'hosts/machine/xyz'.  Use -f option to
```

```
force removal of all stale caches, including special ones
'self', 'dce-rgy', and 'cds-server'.  -f ignored if -p is also
specified
```

The following example shows how the number of credential files is reduced:

```
# ls /opt/dcelocal/var/security/creds | wc -w
     17
# rmxcred -v
Principals with expired tickets:
         /.../old.itsc.austin.ibm.com/cell_admin
         /.../old.itsc.austin.ibm.com/cell_admin
         /.../old.itsc.austin.ibm.com/cell_admin
         /.../old.itsc.austin.ibm.com/cell_admin
# ls /opt/dcelocal/var/security/creds | wc -w
     13
```

Put the command rmxcred into crontab so that credentials are cleaned up at regular intervals:

1. Log in as root.
2. Call crontab -e, which opens up a vi editor session on your crontab file.
3. Insert, for instance, the following line which causes rmxcred to be run daily at 1:00 am:

   ```
   0 1 * * * /bin/rmxcred -h 10
   ```

   It will remove credential files that have been expired for 10 or more hours.
4. Save the file.
5. You can check the entry with crontab -l.

6.3.5.4 Socket Files for Local RPC

In AIX DCE 1.3, ncacn_unix_stream binding handles were available to represent RPC connections within the same machine. The protocol sequence was used to communicate over a local socket special file. Its file name was created with an embedded UUID, which guaranteed that the same name was never used over again. This could create a large number of socket files that need to be cleaned up. These ncacn_unix_stream bindings are no longer used in AIX DCE 2.1.

Socket files are created only for endpoints using the connection-oriented ncacn_ip_tcp protocol sequence. When a well-written DCE server application exits under normal conditions, it will unregister its endpoints from the RPC endpoint map, among other things, before it exits. This should also remove the socket file.

When for any reason these files are not properly cleaned up, you will get stale socket files, which occupy i-nodes. The following utility can be run to delete stale socket files:

```
rpc.clean -p /opt/dcelocal/var/rpc/socket
```

This command is also executed as part of the dce.clean script to stop DCE. In fact, once DCE is stopped, the files in /opt/dcelocal/var/rpc/socket could also be manually deleted.

6.3.6 Managing Caches on Client Machines

Client systems house two local caches, one for CDS and one for DFS. They speed up CDS lookups and DFS data access. However, they can cause DCE operations on the client to be either really slow, because many time-outs occur, or to completely hang if one of the following occurs:

- They point to a server that is not running.
- Information is outdated.
- Cache is corrupted or destroyed by a mistaken operator intervention.

This section covers the aspects of refreshing these caches. See also 6.3.5.2, “Cache Files” on page 179, for limiting the cache sizes.

6.3.6.1 Managing the CDS Clerk Cache

In order to bind to servers, DCE client programs need to contact the CDS for binding handles. All client requests go via the local cds_clerk. The cds_clerk caches all the information it gets from the CDS servers in a local cache. The cache is basically held in virtual memory. It is written to disk when the CDS clerk is disabled. It can be found in directory /opt/dcelocal/var/adm/directory/cds.

CDS access requests from client applications always go via a CDS clerk and its cache. If the cache gets messed up or contains outdated information because some server information has been changed in the cell, the application may experience time-outs or, in the worst case, fail because it receives invalid binding information. Since binding handles are usually received in a random order, one application may be lucky to get a good binding handle, another gets a stale handle, and the server call fails with a communication error. The default time-out is 30 seconds. The application can try the next handle and so on. So we must find a way to get rid of the stale entries.

The CDS server does not know what information the clients store; so it cannot notify the clients of any changes. The only information that is passed to CDS clients is the location of clearinghouses via advertisements. However, it is only a matter of time when CDS clerk cache entries are refreshed by client requests. The expiration age is usually between 8 and 12 hours. See “Expiration Age of CDS Clerk Cache Entries” on page 182, for more information.

Many of the administration tasks described in this publication state that it is recommended to refresh all client caches. Think twice before you do it, because:

- You have to do work on each client.
- It can affect network performance.
- If the client is connected via WAN, it has a cached CDS server entry that is wiped out and must be manually added again.

If only a minor change was made to CDS server information, you may decide to accept some infrequent time-outs on client machines for a few hours until they are refreshed by subsequent client requests. Before you wipe out the whole cache, you should consider forcing updates from the CDS server for specific CDS entries with one of the following commands:

```
# dcecp -c rpcentry show /.:/<some_entry_name>
# dcecp -c rpcgroup list /.:/<some_rpc_group>
old # rpccp show entry /.:/<some_entry_name> -u
```

Note that for the dcecp command you do not need an -u flag. The default is to always update the CDS clerk cache entry. If you did not want to contact the CDS server on every command, you would have to specify the -noupdate flag:

```
# dcecp -c rpcentry show /.:/<some_entry_name> -noupdate
```

The rest of this section covers the following topics:

1. An AIX DCE documentation excerpt about expiration age
2. A procedure to wipe out the CDS clerk cache
3. A procedure that wipes out caches, credentials, and endpoint maps
4. A procedure to define the cached server entry on a WAN-connected client

Expiration Age of CDS Clerk Cache Entries: It is the responsibility of the client side to have updated CDS information. Normally, the client program does not have to do anything; the RPC runtime takes care of checking the age of the cache entry that is being queried and triggers a refresh from the CDS server, if necessary. However, the application can set an expiration age. This has to be used with care because frequent refreshes may affect network performance.

The following description of the rpc_ns_mgmt_set_exp_age() routine is an excerpt out of the AIX DCE documentation:

When an application begins running, the RPC runtime specifies a random value of between 8 and 12 hours as the default expiration age. The default is global to the application. Normally, avoid using this routine; instead, rely on the default expiration age. The RPC NSI next operations, which read data from name service attributes, use an expiration age. A next operation normally starts by looking for a local copy of the attribute data that an application requests. In the absence of a local copy, the next operation creates one with fresh attribute data from the name service database. If a local copy already exists, the operation compares its actual age to the expiration age being used by the application. If the actual age exceeds the expiration age, the operation automatically tries to update the local copy with fresh attribute data from the name service database. If updating is impossible, the old local data remains in place and the next operation fails, returning the rpc_s_name_service_unavailable status code.

Setting the expiration age to a small value causes the RPC NSI next operations to frequently update local data for any name service attribute that your application requests. For example, setting the expiration age to 0 (zero) forces all next operations to update local data for the name service attribute that your application has requested. Therefore, setting small expiration ages can create performance problems for your application. Also, if your application is using a remote server with the name service database, a small expiration age can adversely affect network performance for all applications.

The cleanup_cds_cache Procedure: Binding handles are cached in each CDS clerk cache. If we want to avoid nasty time-outs, you should refresh the clerk caches on all systems when binding handles become invalid. The following procedure deletes the CDS clerk cache without interrupting DCE operation significantly:

```
#!/bin/ksh
#cds cache cleaning.
#Shell Script to remove the local CDS cache
#(It will be recreated, when cdsadv starts again)
#
# Tested with AIX DCE 2.1     01/19/96
```

```
#
CDS_SERVER=NO

ps -ef | grep cdsd | grep -v grep >/dev/null 2>&1
if [ $? = 0 ]
then
  CDS_SERVER=YES
  echo "Disabling CDS server and clerk"
  /etc/dce.clean cds
  # With OSF commands:
  #cdscp disable clerk    # Disable clerk first. Once the server is
  #cdscp disable server   # gone, the clerk cannot be disable anymore
else
  echo "Disabling CDS clerk"
  /etc/dce.clean cdsadv
  # With OSF commands:
  #cdscp disable clerk
fi

echo "Removing CDS clerk cache"
rm /opt/dcelocal/var/adm/directory/cds/cds_cache.*
rm /opt/dcelocal/var/adm/directory/cds/cdsclerk_*

if [ $CDS_SERVER = "YES" ]
then
  echo "Restarting CDS server and clerk"
  /etc/rc.dce cds
  # With OSF commands:
  #cdsadv
  #cdsd
else
  echo "Restarting CDS advertiser (and clerk)"
  /etc/rc.dce cdsadv
  # With OSF commands:
  #cdsadv
fi

echo "CDS clerk cache is cleared "
echo "You can look at it with \"cdscp dump clerk cache\"\n"
```

On an OS/2 client, you need to perform the same steps, which means stopping the CDS client, erasing the files and restarting the client:

```
[C:] dcestop cds_c
[C:\] cd \opt\dcelocal\var\adm\dir\cds
[C:\OPT\DCELOCAL\VAR\ADM\DIR\CDS] erase cdscache.*
[C:\OPT\DCELOCAL\VAR\ADM\DIR\CDS] cd \
[C:\] dcestart cds_c
```

If applications store binding information internally or have no sufficient recovery mechanism to handle communication errors, applications may need to be stopped and restarted. In such cases, the application would not read the refreshed CDS clerk cache and hence `cleanup_cds_cache` would not help.

The cleanup_cache Procedure: This is the full cache refresh procedure. This procedure should actually only be used in problem situations where `cleanup_cds_cache` does not help, for instance when the RPC endpoint map gets corrupted on a system that is a server of any kind.

Since cleanup_cache removes everything that DCE servers and clients need to communicate, you must also stop and restart all DCE applications running on that system. If DFS is running, the system must be rebooted.

The procedure first stops DFS and DCE, if they are running. Then it removes all the caches on that system as shown in the listing below. This ensures that a system does not use its invalid cache anymore. Under normal circumstances,

the CDS cache is kept while rebooting or restarting DCE. RPC endpoint maps and credentials are removed upon restart of DCE after a system reboot.

```
#!/bin/ksh
#general cache cleaning.
#Shell Script to remove the rpc config files and the credential files
#to make sure the cache and the endpoint mapper are cleaned up.
#
# Tested with AIX DCE 2.1     01/19/96
#

cat << EOI
If you want to continue, you will have to restart all your DCE Applications
and users have to login again, because this script removes all credentials
and RPC end points.

If you run DFS (client or server), you must reboot this system.

If you just want to clean the CDS clerk cache, use cleanup_cds_cache.

EOI
echo "Do you want to continue [y/n]: \c"
read a

if [ X$a != X"y" ]
then
  exit 1
fi

echo "Stopping DCE and DFS ..."
/etc/dfs.clean
/etc/dce.clean

echo "Removing socket files ...\n"
rm -f /opt/dcelocal/var/rpc/socket/*

echo "Removing the endpoint maps ...\n"
rm -f /opt/dcelocal/var/dced/Ep.db              # for DCE 2.1
rm -f /opt/dcelocal/var/rpc/rpcdep.dat          # for DCE 1.3
rm -f /opt/dcelocal/var/rpc/rpcdllb.dat         # for DCE 1.3

echo "Removing everyone's credentials ...\n"
ls /opt/dcelocal/var/security/creds | grep -v ffffffff | xargs -i rm -f {}

echo "Removing the CDS cache files ...\n"
rm -f /opt/dcelocal/var/adm/directory/cds/cds_cache.*
rm -f /opt/dcelocal/var/adm/directory/cds/cdsclerk_*

echo "All cache files erased.\n"
echo "Remember: DCE/DFS is currently stopped.\n"
```

On an OS/2 system, you need to perform the same steps, which means stopping the DCE and DFS, erasing the files and restarting the DCE and DFS:

```
[C:] dcestop
[C:\] erase \opt\dcelocal\var\dced\Ep.db
[C:\] erase \opt\dcelocal\var\security\creds\*
[C:\] erase \opt\dcelocal\var\adm\dir\cds\cdscache.*
```

Then shut down and reboot the system and call dcestart.

The create_cds_entry Procedure: When a client system cannot be reached via IP broadcasts from a CDS server or vice versa, it cannot find a CDS server. A cached server entry must be manually defined into the CDS clerk cache:

```
cdscp define cached server <ip_name> tower ncacn_ip_tcp:<ip_addr>
```

When you first install a DCE node that is not in the same network as the CDS server, you call mkdce with the -c flag. This instructs mkdce to set up a cdscp

defined cached server call and to put this call also into the /etc/rc.dce startup file.

The information is stored in the clerk cache. So when you wipe the cache out, you need to redefine the cached server. The following shell script create_cds_entry does that for you:

```
#!/bin/ksh
if [ "$1" != "-c" ]
then
  echo "\nUsage:     create_cds_entry -c <CDS_server_name>\n"
  echo "Purpose:   Sets a cached CDS server for the local CDS clerk\n"
  exit
fi

cat << EOI
Since CDS is not working you must be able to find the security
server via the pe_site file.
EOI
echo "Is /opt/dcelocal/etc/security/pe_site up to date? [y/n]: \c"
read a

if [ X$a != X"y" ]
then
  echo "\n>> Please edit /opt/dcelocal/etc/security/pe_site first;"
  echo ".. chpesite does not work at this point, it needs a working CDS."
  exit 1
fi

ip=`host $2 | cut -f3 -d' '`
BIND_PE_SITE=1; export BIND_PE_SITE
cdscp define cached server $2 tower ncacn_ip_tcp:$ip
unset BIND_PE_SITE
```

Then you need to edit the /etc/rc.dce file, look for the line that contains the CACHE_SRV environment variable, and enter the cdscp defined cached server command there.

6.3.6.2 Managing the DFS Client Cache

This topic is comprehensively described in *The Distributed File System (DFS) for AIX/6000* redbook and in the AIX DFS documentation. The cm command suite is used to manage the DFS client's cache manager. This section should serve as a short reminder of some of the most important cm subcommands:

cm flush	Forces the cache manager to discard data cached from specified files or directories and affects only data which has not been altered. Data that needs to be stored back to the server remains in the cache.
cm flushfileset	Forces the cache manager to discard data cached from filesets that contain specified files or directories. Again, this affects only unaltered data.
cm lsstores	Lists filesets that contain data the cache manager cannot write back to a file server machine.
cm resetstores	Cancels attempts by the cache manager to contact unavailable file server machines and discards all data the cache manager cannot store to such machines. You should run cm lsstores first to check which filesets are concerned.
cm checkfilesets	Forces the cache manager to update fileset-related information. It forces the cache manager to fetch the most recent information available about a fileset from the FLDB.
cm setpreferences	Sets the cache manager's preferences for file server machines. The cm setpreferences -fldb command sets preferences for FLDB server access.

There is no command to flush the entire cache. Should this ever be necessary, do not just delete the files as with CDS, but remove and reconfigure the DFS client.

6.4 Administering Users and Groups

DCE is designed to manage a large number of users in a cell. Administrators work on tasks such as adding, modifying, and deleting users, accounts, and groups in the DCE security registry. DCE provides some basic commands for the tasks. Managing single users is nicely supported by these commands or within SMIT. But current DCE implementations lack tools to perform these in a large scale. Not even a nice Administration GUI (graphical user interface) solves this problem, unless it provides some form of command-line interface to enable using scripts.

The second problem after mass DCE user management is that DCE login is not integrated into AIX login by default. However, AIX Version 4 now supports an authentication method that contacts the DCE registry and imports all necessary user information from there.

Tools for mass user-management and login integration are required to deploy DCE on a large scale. We can propose a solution for both issues:

- In 7.5, “Mass User/Group (and ACL) Management” on page 271, we describe a set of tools for user management which we have designed and implemented during this project.
- A summary description of the integrated login feature available on AIX DCE 2.1 is given.

Our user-management tools are designed to also manage user-related ACLs. This is important for supporting tasks such as migration from other environments to DCE or splitting/joining of existing DCE cells.

This section focuses on the user-management tools as well as some general user-management issues and explains how to configure the integrated login feature. The following items are discussed:

1. Using the `rgy_edit` command on both AIX and OS/2 to manage users
2. Using the `dcecp` command on both AIX and OS/2 to manage users
3. Operating system-dependent user-management tools
4. Mass user management on AIX (tools provided on the diskette of this book)
5. A test with a large number of users
6. Configuring integrated login

6.4.1 Managing Users With the rgy_edit Command on AIX and OS/2

The `rgy_edit` command has been provided since the first release of DCE. It is a tool for managing the registry database, such as adding, modifying, and deleting users, accounts, and groups. The `rgy_edit` command is supported on every DCE platform, such as AIX DCE 1.3, AIX DCE 2.1, DCE for OS/2 Warp, and so on. Typically, its subcommands are used from its interactive prompt (`rgy_edit=>`). The following simple examples show how to manage users by using the `rgy_edit` command. These commands are used in our user-management tools.

6.4.1.1 Adding Users

Make sure you are *cell_admin* or have *cell_admin* privileges; then use the following commands to create a principal, and view what you created:

```
# rgy_edit
Current site is: registry server at /.../itsc.austin.ibm.com/subsys/dce/sec/master
rgy_edit=> domain principal
Domain changed to: principal
rgy_edit=> add manabu 1001
rgy_edit=> view manabu -f
manabu                                          1001
  Uuid:       000003e9-60d1-21cf-9300-10005a4f4629
  Primary:  pr   Reserved:   --
  Quota: unlimited
```

A principal named *manabu* whose UID is *1001* was created. Then the principal you just created is added to an account. The account in this case has a password of *secret*, and it becomes a member of the group *none* and the organization *none*. The cell_admin password in this example is *-dce-*.

```
rgy_edit=> domain account
Domain changed to: account
rgy_edit=> add manabu -g none -o none -pw secret -mp -dce-
rgy_edit=> view manabu -f
manabu [none none]:*:1001:12::/::
  created by: /.../itsc.austin.ibm.com/cell_admin  1996/02/06.16:00
  changed by: /.../itsc.austin.ibm.com/cell_admin  1996/02/06.16:00
  password is: valid, was last changed: 1996/02/06.16:00
  Account expiration date: none
  Account MAY be a server principal
  Account MAY be a client principal
  Account is: valid
  Account CAN NOT get post-dated certificates
  Account CAN get forwardable certificates
  Certificates to this service account MAY be issued via TGT authentication
  Account CAN get renewable certificates
  Account CAN NOT get proxiable certificates
  Account CAN NOT have duplicate session keys
  Good since date: 1996/02/06.16:00
  Max certificate lifetime: default-policy
  Max renewable lifetime: default-policy
```

6.4.1.2 Adding Groups

Create a new *itsc* group with a GID of *20* and view it:

```
rgy_edit=> domain group
Domain changed to: group
rgy_edit=> add itsc 20
rgy_edit=> view itsc -f
itsc                                             20
  Uuid:       00000014-6193-21cf-9301-10005a4f4629
  Primary:  pr   Reserved:   --
  Project List:1
```

6.4.1.3 Modifying Users

Change the primary group of user *manabu* to the group you just created:

```
rgy_edit=> domain account
Domain changed to: account
rgy_edit=> change -p manabu -ng itsc
Change account "manabu none none" [y/n/g/q]? y
rgy_edit=> view manabu -f
manabu [itsc none]:*:1001:20::/::
.....
.....
```

Next, disable login for user *manabu* (modify the status to *not valid*):

```
rgy_edit=> domain account
Domain changed to: account
rgy_edit=> change -p manabu -anv
Change account "manabu none none" [y/n/g/q]?y
rgy_edit=> view manabu -f
.....
.....
Account is: NOT valid
.....
.....
```

6.4.1.4 Deleting Users

Delete principal *manabu*, and verify that it is really gone:

```
rgy_edit=> domain principal
rgy_edit=> delete manabu
Please confirm delete of name "manabu" [y/n]? (n) y
rgy_edit=> view manabu -f
?(rgy_edit) Cannot retrieve entry for manabu - Entry not found (Registry Edit
Kernel) (dce / sad)
rgy_edit=> exit
```

When you exit, all the changes you have made are committed. The account is deleted together with the principal.

6.4.1.5 Deleting Groups

Delete group *itsc*, and check with the view command to make sure it is deleted:

```
rgy_edit=> domain group
Domain changed to: group
rgy_edit=> del itsc
WARNING: any accounts for this group (itsc) will also be deleted.
Please confirm delete of name "itsc" [y/n]? (n) y
rgy_edit=> view itsc
?(rgy_edit) Cannot retrieve entry for itsc - Entry not found (Registry Edit
Kernel) (dce / sad)
```

Together with the group, all accounts with this group as their primary group will also be deleted, but not the principal associated with the account.

6.4.2 Managing Users With the dcecp Command on AIX and OS/2

The dcecp command is *the* general systems-management tool of DCE and has been supported since OSF DCE 1.1. So it is supported on OSF DCE 1.1-based systems, such as AIX DCE 2.1, DCE for OS/2 Warp, and so on. From a functional point of view, it is a command that integrates many of the existing DCE commands, such as rgy_edit, acl_edit, cdscp, and so on. Typically, we use its subcommands on its interactive prompt (dcecp>). The dcecp command is implemented as a superset of tclsh; consequently, we can use Tcl commands on dcecp as well.

The following sections show simple examples of how to manage users by using the dcecp command. These examples fully correspond to the examples described in 6.4.1, “Managing Users With the rgy_edit Command on AIX and OS/2” on page 186.

6.4.2.1 Adding Users

Create a user, *manabu*, using the dcecp command. You will immediately see how it is different from the nonintegrated commands:

```
# dcecp
dcecp> user create manabu -uid 1001 -mypwd -dce- -password secret \
-group none -org none
dcecp> user show manabu
{fullname {}}
{uid 1001}
{uuid 000003e9-616a-21cf-9300-10005a4f4629}
{alias no}
{quota unlimited}
{groups none}
{acctvalid yes}
{client yes}
{created /.../itsc.austin.ibm.com/cell_admin 1996-02-07-10:16:31.000-06:00I-----}
{description {}}
{dupkey no}
{expdate none}
{forwardabletkt yes}
{goodsince 1996-02-07-10:16:31.000-06:00I-----}
{group none}
{home /}
{lastchange /.../itsc.austin.ibm.com/cell_admin 1996-02-07-10:16:31.000-06:00I-----}
{organization none}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid yes}
{renewabletkt yes}
{server yes}
{shell {}}
{stdtgtauth yes}
```

Note: The dcecp command works on *objects*. Here, an object of *user* means a combination of a principal and an account. The only difference to the procedure that uses rgy_edit is that the user create subcommand of dcecp creates a CDS directory on the user’s behalf under */.:/users*. Check it using the cdsli command or the dcecp command:

```
# cdsli -woRld /.:
.....
d       /.:/users
d       /.:/users/manabu
.....
```

or

```
#dcecp
dcecp> directory list /.:/users -s
manabu ....
```

The concept of *user objects* was introduced in OSF DCE 1.1.

6.4.2.2 Adding Groups

Create an *itsc* group with a GID of *20*:

```
# dcecp
dcecp> group create itsc -gid 20
dcecp> group show itsc
{alias no}
{gid 20}
{uuid 00000014-6198-21cf-9301-10005a4f4629}
{inprojlist no}
{fullname {}}
```

6.4.2.3 Modifying Users

Change the primary group of user *manabu* to the group you just created:

```
dcecp> group add itsc -member manabu
dcecp> account modify manabu -group itsc
dcecp> user show manabu
.....
.....
{group itsc}
.....
.....
```

Disable *manabu*'s account (set it to status *invalid*):

```
dcecp> account modify manabu -acctvalid no
dcecp> user show manabu
.....
.....
{acctvalid no}
.....
.....
```

6.4.2.4 Deleting Users

Delete user *manabu*, and verify that it is really gone:

```
dcecp> delete user manabu
dcecp> user show manabu
Error: User "manabu" doesn't exist.
```

6.4.2.5 Deleting Groups

Delete group *itsc*, and check with the view command to make sure it is deleted:

```
dcecp> group delete itsc
dcecp> group show itsc
Error: Registry object not found
```

Together with the group, all accounts with this group as their primary group will also be deleted, but not the principal associated with the account.

6.4.3 Operating System-Dependent Management Tools

Since OSF does not provide an integrated, user-friendly administration tool for DCE, vendors usually create tools for the administration of DCE. On AIX, SMIT was enhanced to manage DCE. As for OS/2, a user-friendly GUI tool is provided.

6.4.3.1 SMIT on AIX

SMIT (System Management Interface Tool) is a nicely designed system-management tool on AIX. It provides GUI-based screens and character-based screens. Whichever you prefer can be used. The DCE management commands were integrated so that SMIT can be used to manage the DCE environment. Easy-to-use menus and integrated assistance allow you to manage a DCE environment without deep knowledge of commands. At the present time, however, SMIT is still calling the 'old' commands, such as `rgy_edit`.

6.4.3.2 GUI-Based DCE Configuration Tool on OS/2

DCE for OS/2 WARP offers a GUI-based DCE administration tool. The structure of the tool is similar to SMIT. Easy-to-use menus allow administration of the DCE environment without knowledge of DCE commands. See Figure 13 on page 56. In this window, click on the **DCE Administration** icon. It shows you an easy-to-understand submenu. You can manage users, groups, and accounts with this tool. The GUI is based on a new object-oriented class library, the MOCL (Managed Object Class Library), and is independent of OSF-provided management commands, such as `rgy_edit` or `dcecp`. The MOCL is based on IBM's System Object Model (SOM). For the time being, the MOCL is not made available to programmers.

6.4.4 Mass User-Management Tools on AIX

Our user-management tools provide a sophisticated solution to manage a large-scale DCE environment. The concepts and structure of the tools as well as the details about the commands are described in 7.5, "Mass User/Group (and ACL) Management" on page 271.

Make sure the management tools have been installed. See Appendix A, "Installing the Tools" on page 303, for instructions how to install the tools. Assume we have a directory, /umgt, which is to hold our user information database, and we have restored the shell scripts into this directory. In order to use the commands correctly, we need to make this directory the current directory. Also, make sure the PATH variable contains the current directory:

```
# cd /umgt
# PATH=$PATH::
```

Note: The tools use the old-style OSF commands as well as the Korn shell. Furthermore, they use some other UNIX-specific tools, such as `grep`, `sed`, `cut`,

and awk. The dcecp is based on the Tcl shell and is theoretically platform-independent. However, since we were using early code, we had some problems trying to port the tools to Tcl and eventually ran out of time. So, porting the tools would be the subject of a further project.

6.4.4.1 Adding Users

The concepts and the syntax of the commands used to add new users is described in 7.5.4, “Adding Users: add_users” on page 287, and in 7.5.5, “Enabling Users for DCE Login: rgy_enable_users” on page 292.

This section gives a few examples of the usage, such as:

1. Creating users from list of user names
2. Creating a user with specific UID

Adding Users from a List of Names: This is an example where the DCE registry automatically assigns the UIDs. It takes the next available UIDs.

1. Create a file with user names:

```
# echo "mary manabu mark" > /tmp/users
```

2. Call add_users to create the principals and a default account that will not be enabled for login yet:

```
# cd /umgt
# add_users /tmp/users
Checking to be sure you are cell_admin ...
 You must login as cell_admin first ... sorry
```

3. Log in as cell_admin and try again:

```
# add_users /tmp/users
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to add ...
  Checking mary ...ok      (file will be created)
  Checking manabu ...ok      (file will be created)
  Checking mark ...ok      (file will be created)

You are going to add       3 users
Starting to work with rgy_edit ...

Please provide your password:


   Adding principal mary ...ok
   Adding principal manabu ...ok
   Adding principal mark ...ok

*** Ended to add users in DCE
```

4. Check the principals that were created:

```
# dcecp
dcecp> principal cat -s
.....
mary
manabu
mark
```

5. Check the accounts that were created:

```
dcecp> account cat -s
.....
mary
manabu
mark
.....
dcecp> account show mary
{acctvalid no}
{client yes}
{created /.../itsc.austin.ibm.com/cell_admin 1996-01-30-09:54:15.000-06:00I-----}
{description {}}
{dupkey no}
{expdate none}
{forwardabletkt yes}
{goodsince 1996-01-30-00:00:00.000-06:00I-----}
{group none}
{home /}
{lastchange /.../itsc.austin.ibm.com/cell_admin 1996-01-30-09:54:15.000-06:00I-----}
{organization none}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid no}
{renewabletkt yes}
{server yes}
{shell {}}
{stdtgtauth yes}
```

6. The UDF of user mary looks as follows:

```
# cat dce_users/mary

# -- !!!!!!!!!!!!!!  Do NOT change manually the first part  !!!!!!
# --- Principal info:
uuid=0000015d-92c2-2e78-a100-02608c2fff91
uid=349
groups=none
# --- Account info:
group=none
org=none
valid=NO
gecos=Account for mary
homedir=/:/dfs_home/mary
size=
initprog=/bin/ksh
expir_date=97/01/30
good_since=96/01/30
# --- ACL_INI info:
# --- ACL info:
# --- State and last access:
state=SUSPENDED
last_time_access=Tue Jan 30 10:25:41 CST 1996 op=add_users
#!!
#!!!!!!!!!!!!!!!! Do NOT change manually above this line !!!!!!!!!!!!!
#!! Edit below (values that could not be applied):
#!! Edit below (values to be applied next time):
```

At this point, the user is added but not enabled for DCE login; the user's state is SUSPENDED and the account invalid. In this state, we could now change the parameters. Let's do it and change, for example, the GECOS field which was generated per default into *Mary White, Dept 99S*.

```
# echo "ADD_gecos=Mary White, Dept 99S" >> dce_users/mary
# cat dce_users/mary
```

7. Enable all three users for DCE login:

```
# rgy_enable_users /tmp/users
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to rgy_enable ...
  Checking mary ...ok
  Checking manabu ...ok
  Checking mark ...ok

You are going to rgy_enable       3 users
Starting to work with rgy_edit ...

mary manabu mark
   RGY-enabling principal mary ...  ok
   RGY-enabling principal manabu ...  ok
   RGY-enabling principal mark ...  ok

*** Ended to rgy_enable users in DCE
```

8. List the DCE accounts:

```
dcecp> account show mary
{acctvalid no}
{client yes}
{created /.../itsc.austin.ibm.com/cell_admin 1996-01-30-09:54:15.000-06:00I-----}
{description {Mary White, Dept 99S}}
{dupkey no}
{expdate 1997-01-30-00:00:00.000-06:00I-----}
{forwardabletkt yes}
{goodsince 1996-01-30-00:00:00.000-06:00I-----}
{group none}
{home /.../itsc.austin.ibm.com/fs/dfs_home/mary}
{lastchange /.../itsc.austin.ibm.com/cell_admin 1996-01-30-10:25:41.000-06:00I-----}
{organization none}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid no}
{renewabletkt yes}
{server yes}
{shell /bin/ksh}
{stdtgtauth yes}
```

   All the account information is now filled in. Note the GECOS field for user mary.

9. List the UDF file for mary again; her GECOS field is now changed, and the ADD_gecos has disappeared. The state is now RGY_ENABLED and her account valid:

```
# cat dce_users/mary

# -- !!!!!!!!!!!!!  Do NOT change manually the first part  !!!!!!
# --- Principal info:
uuid=0000015d-92c2-2e78-a100-02608c2fff91
uid=349
groups= none
# --- Account info:
group=none
org=none
```

```
valid=YES
gecos=Mary White, Dept 99S
homedir=/:/dfs_home/mary
size=
initprog=/bin/ksh
expir_date=97/01/30
good_since=96/01/30
# --- ACL_INI info:
# --- ACL info:
# --- State and last access:
state=RGY_ENABLED
last_time_access=Tue Jan 30 10:35:45 CST 1996 op=rgy_enable_users
#!!
#!!!!!!!!!!!!!!! Do NOT change manually above this line !!!!!!!!!!!!!
#!! Edit below (values that could not be applied):
```

Create a User with a Specific User ID: To accomplish this task, we first create an empty UDF, fill in the desired values, and then run the add_users command. This is also the procedure you may want to use when you create a new tool to migrate users from an existing base.

Let's assume we want to add a user named felix whose UID should be 1001 because he is defined as such in AIX:

1. Create an empty UDF:

```
# CR_EMTPY_UDF

 Usage:  CR_EMPTY_UDF udf-file udf-dir
         CR_EMPTY_UDF -h

         udf-file        = User definition file to read (=username)
         udf-dir         = Repository name
         -h      = Display more information

# CR_EMTPY_UDF felix dce_users
```

2. Add the instruction ADD_uid to the UDF, and check the UDF:

```
# echo  "ADD_uid=1001" >> dce_users/felix
# cat dce_users/felix
```

3. Create the DCE principal:

```
# add_users felix
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to add ...
  Checking felix ...ok

You are going to add       1 users
Starting to work with rgy_edit ...

Please provide your password:


   Adding principal felix ...ok

*** Ended to add users in DCE
```

4. Check the UDF again:

```
# cat dce_users/felix
```

The UID is set to 1001, and default account values, like *homedir*, have now been added to the UDF but not to the registry yet. Before you enable the account, you could now change other account attributes.

5. You can also check the DCE registry definition for the new principal and account with:

```
dcecp> principal show felix
{fullname {}}
{uid 1001}
{uuid 000003e9-5be6-21cf-9f00-10005a4f4629}
{alias no}
{quota unlimited}
{groups none}

dcecp> account show felix
{acctvalid no}
{client yes}
{created /.../itsc.austin.ibm.com/cell_admin 1996-01-31-09:48:55.000-06:00I-----}
{description {Account for felix}}
{dupkey no}
{expdate 1997-01-31-00:00:00.000-06:00I-----}
{forwardabletkt yes}
{goodsince 1996-01-31-00:00:00.000-06:00I-----}
{group none}
{home /.../itsc.austin.ibm.com/fs/dfs_home/felix}
{lastchange /.../itsc.austin.ibm.com/cell_admin 1996-01-31-11:16:44.000-06:00I-----}
{organization none}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid no}
{renewabletkt yes}
{server yes}
{shell /bin/ksh}
```

6. Enable the user for DCE login:

```
# rgy_enable_users felix
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to rgy_enable ...
  Checking felix ...ok

You are going to rgy_enable       1 users
Starting to work with rgy_edit ...

felix
   RGY-enabling principal felix ...ok

*** Ended to rgy_enable users in DCE
```

7. Check the DCE account:

```
dcecp> account show felix
{acctvalid yes}
{client yes}
{created /.../itsc.austin.ibm.com/cell_admin 1996-01-31-09:48:55.000-06:00I-----}
{description {Account for felix}}
{dupkey no}
{expdate 1997-01-31-00:00:00.000-06:00I-----}
{forwardabletkt yes}
{goodsince 1996-01-31-00:00:00.000-06:00I-----}
{group none}
```

```
{home /.../itsc.austin.ibm.com/fs/dfs_home/felix}
{lastchange /.../itsc.austin.ibm.com/cell_admin 1996-01-31-11:11:30.000-06:00I-----}
{organization none}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid no}
{renewabletkt yes}
{server yes}
{shell /bin/ksh}
{stdtgtauth yes}
```

8. Also check the UDF file. The account is valid now, and its state is RGY_ENABLED.

6.4.4.2 Modifying Users

Modifying users means changing some of their definitions in an already existing account. To modify their attributes, we have to bring the user into the SUSPENDED state. Suspending a user can be done from any of the states: *RGY_ENABLED, DFS_ENABLED, or FULL_ENABLED*.

We want to look at two examples:

1. Changing the primary group of user mary

   This could have been done right between the `add_user` and `rgy_enable` steps as well.

2. Adding ACLs for user felix's DFS home directory

   This step does not require you to suspend users first because they are not yet DFS-enabled.

Changing the Primary Group of a User: In order to assign user mary a new primary group, g7, we must perform the following steps

1. Check the DCE account information; the primary group is none:

   ```
   dcecp> account show mary
   ......
   ......
   {group none}
   ......
   ......
   ```

2. Suspend user mary:

   ```
   # susp_users mary
   Checking to be sure you are cell_admin ...ok
   Creating a candidate list of users to suspend ...
     Checking mary ...ok

   You are going to suspend       1 users
   Starting to work with rgy_edit ...

      Suspending account mary ...ok

   *** Ended to suspend users in DCE
   ```

3. Add the necessary instructions into her UDF:

   ```
   # echo "ADD_newgrp=g7" >> dce_users/mary
   ```

   If you had to do this for multiple users, you can write a short *for* loop:

```
# for user in `cat /tmp/users`; do
echo "ADD_newgrp=g7" >> dce_users/$user; done
```

4. Enable user mary again:

```
# rgy_enable_users mary
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to rgy_enable ...
  Checking mary ...ok

You are going to rgy_enable       1 users
Starting to work with rgy_edit ...

   RGY-enabling principal mary ...ok

*** Ended to rgy_enable users in DCE
```

5. Check the DCE account information:

```
dcecp> account show mary
.....
.....
{group g7}
.....
.....
```

6. Check the UDF and the principal definition of mary:

```
# cat dce_users/mary
.....
.....
groups= none, g7
.....
.....
group= g7
.....

# dcecp
dcecp> principal show mary
.....
.....
{groups none g7}
.....
.....
```

User mary was a member in the group none before, which was her primary group. Adding a new primary group does not delete her membership in group none. In order to achieve that, we need the instruction DEL_groups=none. We could have specified this in the same step where we added the instruction ADD_newgrp=g7.

```
# susp_users mary
.....

# echo "DEL_groups=none" >> dce_users/mary
# rgy_enable_users mary
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to rgy_enable ...
  Checking mary ...ok
You are going to rgy_enable        1 users
Starting to work with rgy_edit ...

   RGY-enabling principal mary ...ok
```

```
*** Ended to rgy_enable users in DCE

# dcecp
dcecp> principal show mary
.....
.....
{groups g7}
.....

# cat dce_users/mary
# -- !!!!!!!!!!!!!!  Do NOT change manually the first part  !!!!!!
# --- Principal info:
uuid=0000015d-92c2-2e78-a100-02608c2fff91
uid=349
groups= g7
.....
```

Creating DFS ACLs for a User's Home Directory: We must add ADD_ACL_INI instructions in the UDF. Since many users might start off with the same set of ACLs for their home directory and many of them would never change them thereafter, the same set of instructions would be added to many UDFs with a *for* loop.

In order for this step to succeed, the DFS directory must exist. However, if it does not exist, an error message appears, and the ADD_ACL_INI instructions remain in the UDF.

This step would typically be executed during a migration from NFS/NIS or the splitting/joining of a cell.

1. Preparation:

   The principal must be added, and the account must be in the RGY_ENABLED state. Make sure the directory exists, is accessible, and belongs to the correct principal. Finally, you should log in as cell_admin.

   Note: Other than in DFS 1.3, cell_admin does not have sufficient permission to manipulate files or ACLs per default. Make sure that the necessary ACLs are set on the home directories. See 4.5, "Defining Home Directories in DFS" on page 98, for information.

   However, dfs_enable_users will check all these prerequisites and fail with an appropriate message if something is wrong.

2. Add all the ADD_ACL_INI instructions to the UDF(s). We can also set the initial object ACLs and the initial container-creation ACLs:

   ```
   # cat << EOI >> dce_users/mary
   ADD_ACL_INI=dfs#/:/dfs_home/mary#mask_obj:r-x---
   ADD_ACL_INI=dfs#/:/dfs_home/mary#user_obj:rwxcid
   ADD_ACL_INI=dfs#/:/dfs_home/mary#group_obj:rwx---
   ADD_ACL_INI=dfs#/:/dfs_home/mary#other_obj:r-x---
   ADD_ACL_INI_OC=dfs#/:/dfs_home/mary#mask_obj:r-x---
   ADD_ACL_INI_OC=dfs#/:/dfs_home/mary#user_obj:rwxcid
   ADD_ACL_INI_OC=dfs#/:/dfs_home/mary#group_obj:rwx---
   ADD_ACL_INI_OC=dfs#/:/dfs_home/mary#other_obj:r-x---
   ADD_ACL_INI_CC=dfs#/:/dfs_home/mary#mask_obj:r-x---
   ADD_ACL_INI_CC=dfs#/:/dfs_home/mary#user_obj:rwxcid
   ADD_ACL_INI_CC=dfs#/:/dfs_home/mary#group_obj:rwx---
   ```

```
ADD_ACL_INI_CC=dfs#/:/dfs_home/mary#other_obj:r-x---
EOI
```

3. Enable user mary for DFS ACLs:

```
# dfs_enable_users mary
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to dfs_enable ...
  Checking mary ...ok

You are going to dfs_enable       1 users
Starting to work with rgy_edit ...

Checking Availability of DFS ...ok
   DFS-enabling principal mary ...ok

*** Ended to dfs_enable users in DCE
```

The same ADD_ACL_INI instructions would be used to change any of the existing ACL definitions. They would simply be overwritten. You are not limited to mask_obj, user_obj, group_obj, and other_obj. You could, for instance, also give the group *g99* access to the home directory of users (such as mary's in the example below) with the following set of entries:

```
ADD_ACL_INI=dfs#/:/dfs_home/mary#group:g99:rwx---
ADD_ACL_INI_OC=dfs#/:/dfs_home/mary#group:g99:rwx---
ADD_ACL_INI_CC=dfs#/:/dfs_home/mary#group:g99:rwx---
```

With instructions like DEL_ACL_INI, some of these entries can be deleted. The ACLs for owner, group, and others cannot be deleted, though.

Now, how do ACL_INI entries differ from the other category of ACL entries in the UDF?

The ACL_INI entries are related to a certain object and define all permissions for that object. The owner of the UDF that contains these object entries is the owner of the object and represented by the user_obj entry. Thus, if the user is deleted, the object will probably also be removed by the administrator later on. However, since removing the object makes it unnecessary to remove the ACLs, they will not be removed. The primary purpose of this entry is to provide support for managing home directories.

The other type of ACL entries in the UDFs pull all permissions together that a specific user has on any CDS and DFS object. This makes it possible to remove a user or group and to find and delete all their specific ACLs defined anywhere.

To define these other types of ACLs, you follow the same steps as outlined for the ACL_INI entries. The account has to be in state DFS_ENABLED, and the procedure to be used is acl_enable_users.

6.4.4.3 Deleting or Moving Users

Deleting an account involves suspending it and then actually removing it. Removing means first deleting all the ACL entries and group memberships that exist for the user and eventually removing it from the registry. The UDF will be copied to the cemetery directory. This is what the delete_users procedure does.

The objects that the deleted user owned are not automatically deleted; so now is the last chance, for instance, to back up the DFS files. The administrator can then remove these objects.

Since the UDF file is still around and reflects the last state and set of definitions the user had in this cell, the UDF can be used to redefine the user in another cell. So, this procedure can be used to split cells or to join cells.

Let's assume we want to delete all users that have ever been defined with our tools, which means users with a UDF in directory dce_users.

1. First suspend the users. This will set the account to invalid, and users cannot log in anymore. Already logged in users are not affected as long as their ticket is valid:

```
# susp_users all
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to suspend ...

You are going to suspend                4 users
Starting to work with rgy_edit ...

Suspending account manabu ...ok
Suspending account mark ...ok
Suspending account mary ...ok
Suspending account felix ...ok

*** Ended to suspend users in DCE
```

2. Then delete all users:

```
# del_users all
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to delete ...

You are going to delete 4 users
Starting to work with rgy_edit ...

Deleting manabu ...ACLs...Principal... ok
Deleting mary ...ACLs...Principal... ok
Deleting mark ...ACLs...Principal... ok
Deleting felix ...ACLs...Principal... ok

*** Ended to delete users in DCE
```

3. The files are now in directory cemetary_users:

```
# cat dce_users/mark
cat: 0652-050 Cannot open dce_users/mark.
# cat cemetary_users/mark
# ---!!!!!!!!!!!!!!  Do NOT change manually the first part  !!!!!!
# --- Principal info:
uuid=0000015f-92d4-2e78-a100-02608c2fff91
uid=351
groups=none
# --- Account info:
group=none
org=none
valid=NO
gecos=Account for mark
homedir=/:/dfs_home/mark
size=
```

```
initprog=/bin/ksh
expir_date=97/01/30
good_since=96/01/30
# --- ACL_INI info:
# --- ACL info:
# --- State and last access:
state=DELETED
last_time_access=Tue Jan 30 10:25:42 CST 1996 op=del_users
#!!
#!!!!!!!!!!!!!!! Do NOT change manually above this line !!!!!!!!!!!!!
#!! Edit below (values that could not be applied):
#!! Edit below (values to be applied next time):
```

4. The ACLs and principals are deleted.

```
dcecp> principal show mark
Error: Registry object not found
```

6.4.4.4 Users Aliases

Users can have aliases. In fact, you create a new principal name for the same UID and UUID. For the alias name, you can define a new account with new attributes, such as a new primary group and/or other group memberships.

Let's create a principal, pier, and an alias, sal, for pier:

```
dcecp> principal create pier -uid 1002
dcecp> principal show pier
{fullname {}}
{uid 1002}
{uuid 000003ea-5c0b-21cf-8000-10005a4f4629}
{alias no}
{quota unlimited}
dcecp> principal create sal -uid 1002 -alias yes
dcecp> principal show sal
{fullname {}}
{uid 1002}
{uuid 000003ea-5c0b-21cf-8000-10005a4f4629}
{alias yes}
{quota unlimited}
```

As you can see, once the principal is created, it has the same user identifier (UID) number and the same universal user identifier (UUID). The user now can be added as an account with a different group and organization. So, when user pier logs in as sal, he has other privileges than when he logs in with his original account.

Let's add pier as a member of the group staff:

```
dcecp> group add staff -member pier
```

User sal can now be added as a member of group dev:

```
dcecp> group add dev -member sal
```

Now the only difference is the group ID. Let's modify an ACL for the object /.:/fs:

```
dcecp> acl modify /.:/fs -entry -add {user pier r}
dcecp> acl modify /.:/fs -entry -add {user sal rw}
dcecp> acl show /.:/fs -entry
.....
{user sal rw---}
.....
```

Since pier and sal are the same principal, the ACL does not reflect both names but only the last change, which was for sal. Different access permissions for the two accounts can only be achieved via group memberships. Add an ACL entry for the two groups that the users sal and pier belong to:

```
dcecp> acl modify /.:/fs -entry -add {group staff rw}
dcecp> acl modify /.:/fs -entry -add {group dev r}
dcecp> acl show /.:/fs -entry
.....
{group staff rw---}
{group dev r----}
.....
```

The staff group has read and write permissions, while the dev group has only read permission. When the user is logged in with the account pier, it can access the object /.:/fs for read and write because it belongs to the group staff. When he is logged in as account sal, he has only read access to the same object because the group he belongs to has only read permission.

6.4.5 A Test with a Large Number of Users

The POSIX standard defines the least common denominator for the different UNIX implementations. The number of users is limited by the UID (User identifier), which is a signed 2-byte integer. So by default, when you configure a cell, the maximum number (MAXIMUM possible UNIX ID) is set to 32767. This number is not a limit for DCE; it can be dynamically changed to the number you want within the signed 32-bit number. The theoretical number of users is actually 2147483647 (2 Giga). We have experimented successfully at setting the UID number to 2147483647. There is no incompatibility between AIX DCE 2.1 and AIX 4.1; this number is supported in both environments. As such, mapping AIX users to DCE users (or vice versa) is a straightforward task, and DCE login integration with AIX 4.1 is easy.

6.4.5.1 How to Prepare For Setting a Larger Number of Users

As a matter of fact, even if the theoretically allowable number of users is high, there are other limits you have to consider. Before starting to add many users in your cell, you must solve at least the following issues:

- The amount of RAM memory available on your system

  This feature is very important on the machine where the security server is running, be it a master security server or a replicated security server. Actually, the whole registry database is held in memory. If you have limited memory, then enlarge your paging space. According to our experience, when you add a user, it will take approximately 1.5 KB.

- Paging space

  As stated above, the paging space needs to be large enough to hold a whole registry database. Normally, the paging space is configured with twice the RAM size. But this size may not be sufficient; so care must be taken when adding many users.

- Disk space /var/dce

  The directory /var/dce/security is the place where the registry database is located. Care must also be taken. It is strongly recommended that you create a separate file system at least for /var/dce. Then you can increase the space dynamically as needed.

- Maximum allowable User ID

  This is a software setting. You might increase the maximum number to the limit you want.

6.4.5.2 How to Set and Use the Maximum Number of Users

This number can be changed dynamically to the threshold number needed. Show the current limit in the registry:

```
#dce_login cell_admin
Enter Password:

#dcecp
dcecp> registry show
{deftktlife +0-10:00:00.000I-----}
{hidepwd yes}
{maxuid 32767}
{mingid 100}
{minorgid 100}
{mintktlife +0-00:05:00.000I-----}
{minuid 100}
{version secd.dce.1.1}
dcecp>
```

Then try to add a user with a big UID number

```
dcecp> user create brice -uid 2000000 -gr none -org none -passw <brice_passwd>
-mypwd <cell_admin_passwd>

Error: Invalid data record
dcecp>
```

The system responds by issuing the message *Error: Invalid data record*. Actually, the current maximum UID is set to 32767. To be able to add a user with a number bigger than 32767, you have to set the *maxuid* to the threshold you need. Let's set this number to 3000000 (three million). Set the maximum number of users (*maxuid*) and verify:

```
#dcecp
dcecp>registry modify -uid 3000000
dcecp> registry show
{deftktlife +0-10:00:00.000I-----}
{hidepwd yes}
{maxuid 3000000}
{mingid 100}
{minorgid 100}
{mintktlife +0-00:05:00.000I-----}
{minuid 100}
{version secd.dce.1.1}
dcecp>
```

We can notice above that the *maxuid* is effectively set to 3000000. From now on, we can add up to 3000000 users. Try again to create a user with a big number:

```
#dcecp
dcecp>user create brice -uid 2000000 -gr none -org none -home /:/home/brice\
>-passw <brice_passwd> -mypwd <cell_admin_passwd>
dcecp>
dcecp> user show brice
{fullname {}}
{uid 2000000}
{uuid 00000bb8-630b-21cf-a700-10005a4f4629}
{alias no}
{quota 0}
{groups daemon}
{acctvalid no}
{client yes}
{created /.../itso/cell_admin 1996-02-09-12:00:51.000-06:00I-----}
{description {}}
{dupkey no}
{expdate none}
{forwardabletkt yes}
{goodsince 1996-02-09-12:00:41.000-06:00I-----}
{group daemon}
{home /:/home/brice}
{lastchange /.../itso/cell_admin 1996-02-09-12:00:51.000-06:00I-----}
{organization none}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid no}
{renewabletkt yes}
{server yes}
{shell /usr/bin/ksh}
{stdtgtauth yes}
No policy
```

Log on to AIX and DCE at the same time. For this, we assume that the DCE login is integrated with AIX according to 7.4, “Integrated Login AIX and DCE” on page 265.

```
AIX Version 4
(C) Copyrights by IBM and by others 1982, 1994.
login: brice
brice's Password:

ev2::/-> id
uid=2000000(brice) gid=12(none)

ev2::/-> touch /tmp/myfile
ev2::/-> ls -l /tmp/myfile
-rw-r--r--   1 brice    none           0 Feb 09 15:23 /tmp/myfile
```

Note that user brice has no local account and that the system displays the coherent owner of the file that user brice created. This is thanks to the dceunixd process, which maintains synchronization with AIX. The command below also shows that user brice receives his credentials (and tickets) after his (single) login to AIX. This is shown by the principal identity of the klist command.

```
ev2::/-> klist
DCE Identity Information:
        Global Principal: /.../itso/brice
        Cell:      001c7122-b6fd-111a-bf77-10005a4f52c2 /.../itso
        Principal: 001e8480-6311-21cf-a700-10005a4f4629 brice
        Group:     0000000c-b6fd-211a-bf01-10005a4f52c2 none
```

```
        Local Groups:
                0000000c-b6fd-211a-bf01-10005a4f52c2 none

Identity Info Expires: 96/02/09:23:15:10
Account Expires:       never
Passwd Expires:        never

Kerberos Ticket Information:
Ticket cache: /opt/dcelocal/var/security/creds/dcecred_470c5f00
Default principal: brice@itso
Server: krbtgt/itso@itso
        valid 96/02/09:15:15:10 to 96/02/09:23:15:10
Server: dce-rgy@itso
        valid 96/02/09:15:15:10 to 96/02/09:23:15:10
Server: dce-ptgt@itso
        valid 96/02/09:15:15:11 to 96/02/09:17:15:11
Client: dce-ptgt@itso   Server: krbtgt/itso@itso
        valid 96/02/09:15:15:11 to 96/02/09:17:15:11
Client: dce-ptgt@itso   Server: hosts/ev2/self@itso
        valid 96/02/09:15:15:11 to 96/02/09:17:15:11
Client: dce-ptgt@itso   Server: dce-rgy@itso
        valid 96/02/09:15:17:08 to 96/02/09:17:15:11
```

6.4.6 Configuring Integrated Login

Use the following short path to configure a system for integrated security operations (for details, see 7.4.3, “Configuring a System for Integrated Security” on page 270):

1. Ensure that the module /usr/lib/security/DCE is installed on the machine.
2. Edit the /etc/security/login.cfg file to include the following lines:

   ```
   DCE:
       program = /usr/lib/security/DCE
   ```

   This defines the DCE authentication method to the system.
3. Ensure that the dceunixd daemon is running on the machine; if not, start this program:

   ```
   # dceunixd
   ```

   This daemon communicates to the DCE servers *secd* and *dced* on behalf of the AIX commands. It should be added to the /etc/inittab file as `dceunixd -d 1`.
4. Edit the /etc/security/user stanza file to allow and/or deny DCE access for users. See 7.4.1, “AIX 4.1+ Authentication Parameters” on page 265, for instructions on editing this file.
5. Create or edit the /opt/dcelocal/etc/passwd_override and /opt/dcelocal/etc/group_override files to explicitly prevent DCE access by certain users. See 7.4.2, “User Synchronization Between AIX 4.1+ and DCE” on page 267, for instructions on editing this file.

6.5 Managing the cell_admin Account

The *cell_admin* account is by default the omnipotent DCE account that has the necessary rights to configure all aspects of DCE. If the cell_admin password or the entire cell_admin account gets lost, specific steps have to be followed to restore the lost information.

This section focuses on the following tasks:

1. What to do if the cell_admin password is lost
2. What to do if the cell_admin account has accidentally been removed
3. Adding a new cell administrator account

This last procedure also shows you where cell_admin needs to be defined to have all the necessary rights. If you do not like the fact that one single account is omnipotent, you can assign the rights to several other special accounts.

6.5.1 Restoring the Password for the Cell Administrator

This is the procedure to follow when the cell_admin password is lost or forgotten for any reason:

6.5.1.1 Restoring cell_admin's Password on AIX 4.1

Log in as root on the security server machine, and perform the following steps:

1. Kill the security daemon secd:

   ```
   # /etc/dce.clean secd
   ```

2. Call the secd command in maintenance mode as follows:

   ```
   # secd -locksmith cell_admin -lockpw
   Enter password for locksmith account: <NEW PASSWORD: not-echoed password>
   Reenter password to verify: <NEW PASSWORD: not-echoed password>
   ```

3. After this step, the command hangs; so either press Ctrl-Z followed by the bg command to put it in the background, or open another window and continue in the new window.

4. Log in to DCE with the new password:

   ```
   dce_login cell_admin
   Enter Password: <NEW PASSWORD: not-echoed password>
   ```

5. You must stop secd, which is still running in the background:

   ```
   # dcecp
   dcecp> registry stop /.:/subsys/dce/sec/master
   dcecp> quit
   #
   ```

6. Restart the security server with /etc/rc.dce secd or by simply calling secd from the command line.

7. Log in to DCE again with cell_admin, using the new password.

6.5.1.2 Restoring cell_admin's Password on OS/2 Warp

The following steps show you how to restore the password on OS/2 Warp:

1. Stop the security daemon secd:

   Go to the secd window by clicking on the **Window list**. Stop the secd window.

2. Call the secd command in maintenance mode as follows:

   ```
   c:>secd -locksmith cell_admin -lockpw
   Enter password for locksmith account: <NEW PASSWORD: not-echoed password>
   Reenter password to verify: <NEW PASSWORD: not-echoed password>
   ```

3. After this step, the command hangs; open another window to start the dcelogin session.

4. Log in to DCE with the new password:

   ```
   c:>dcelogin cell_admin
   Enter Password: <NEW PASSWORD: not-echoed password>
   ```

5. You must stop secd, which is still running in the background:

   ```
   c:>dcecp
   dcecp> registry stop /.:/subsys/dce/sec/master
   dcecp>quit
   c:>
   ```

6. Restart secd from the DCE Admin window by clicking on the **DCE Start** icon.

7. Log in to DCE again as cell_admin, using the new password.

6.5.2 Cell Administrator Accidentally Removed

Accidentally, a cell administrator might remove their own user ID:

```
# dce_login cell_admin
Password:
#dcecp
dcecp> user del cell_admin
```

From now on, every time the cell administrator tries to log in, the following message is displayed:

```
# dce_login cell_admin
Sorry.
User Identification Failure. - Registry object not found (dce / sec)
```

Let's first remember how the cell_admin principal and account was set up. Each cell administrator is created initially with the following principal, account, group, and ACL information. That information must be recreated now:

```
# dcecp
dcecp> user show cell_admin
{fullname {}}
{uid 100}
{uuid 00000064-5da3-21cf-bf00-10005aa8cff8}
{alias no}
{quota unlimited}
{groups none acct-admin subsys/dce/sec-admin subsys/dce/cds-admin
subsys/dce/dts-admin subsys/dce/audit-admin subsys/dce/dfs-admin}
{acctvalid yes}
{client yes}
{created /.../brice_cell/cell_admin 1996-02-02-14:49:59.000-06:00I-----}
{description {}}
{dupkey no}
{expdate none}
{forwardabletkt yes}
{goodsince 1996-02-02-14:49:59.000-06:00I-----}
{group none}
{home /}
```

```
{lastchange /.../brice_cell/cell_admin 1996-02-02-14:49:59.000-06:00I-----}
{organization none}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid no}
{renewabletkt yes}
{server yes}
{shell {}}
{stdtgtauth yes}
No policy
```

User cell_admin has an ACL entry user:cell_admin:rwdtc in the following objects and directories:

```
/.:/cell-profile
/.:/fs
/.:/lan-profile
/.:/sec
/.:/sec-v1
/.:/hosts
/.:/users
/.:/users/username
/.:/hosts/hostname
/.:/hosts/hostname/cds-clerk
/.:/hosts/hostname/cds-server
/.:/hosts/hostname/profile
/.:/hosts/hostname/self
/.:/subsys
/.:/subsys/dce
/.:/subsys/dce/dfs
/.:/subsys/dce/dfs/bak
/.:/subsys/dce/sec
```

There may be more CDS objects on which cell_admin has such an ACL entry, depending on the exact configuration of the cell. To find out all the current rights of cell_admin, you can run the get_info_user cell_admin command. See 7.5, “Mass User/Group (and ACL) Management” on page 271, for more information about the user-management tools. The user definition file (UDF) created by the get_info_user command can be also used to recreate the cell_admin account.

6.5.2.1 Recovering cell_admin’s Account on AIX 4.1

Log in as root on the security server machine, and perform the following steps to recreate the cell_admin account:

1. Kill the security daemon:

   ```
   # /etc/dce.clean secd
   ```

2. Start the security daemon secd in maintenance mode with the locksmith option:

   ```
   # secd -locksmith cell_admin
   Account for cell_admin doesn’t exist. Create it [y/n]? (y) y
   Enter password for locksmith account: <NEW PASSWORD: not-echoed password>
   Reenter password to verify: <NEW PASSWORD: not-echoed password>
   ```

3. At this point, the command hangs; so either type in Ctrl-Z followed by the bg command to put it in the background, or open another window and continue in the new window.

4. Run the dce_login command for the cell_admin user:

```
# dce_login cell_admin
Enter Password:  <NEW PASSWORD: not-echoed password>
#
```

5. Run the dcecp command, and make cell_admin a member of the groups listed below:

```
#dcecp
dcecp> group add acc-admin -member cell_admin
dcecp> group add subsys/dce/sec-admin -member cell_admin
dcecp> group add subsys/dce/cds-admin -member cell_admin
dcecp> group add subsys/dce/dts-admin -member cell_admin
dcecp> group add subsys/dce/dfs-admin -member cell_admin
dcecp> quit
```

6. Exit from your current session, and log in again as cell_admin:

```
# exit
# dce_login cell_admin
Password:
#
```

7. Update the ACL entries for all the directories and objects with the following command:

```
# dcecp
dcecp> set  x [exec cdsli -oRdc]
dcecp> foreach i $x {acl modify -entry $i -add {user cell_admin rwdtc}}
```

8. Run the dcecp command with the registry stop subcommand to stop the security daemon which is still running in maintenance mode in the other window or in the background:

```
# dcecp
dcecp> registry stop /.:/subsys/dce/sec/master
dcecp> quit
```

9. Start up the security daemon again, and start working in normal mode:

```
# rc.dce secd
Starting DCE daemons:
        starting secd
```

6.5.2.2 Recovering cell_admin's Account on OS/2 Warp

Perform the following steps to recreate the cell_admin account:

1. Stop the security daemon secd:

   Go to the secd window by clicking on the **Window list**. Stop the secd window.

2. Call secd command in maintenance mode as follows:

```
c:> secd -locksmith cell_admin
Account for cell_admin doesn't exist. Create it [y/n]? (y) y
Enter password for locksmith account: <NEW PASSWORD: not-echoed password>
Reenter password to verify: <NEW PASSWORD: not-echoed password>
```

3. After this step, the command hangs; open another window to start the dcelogin session.

4. Log in to DCE with the new password:

```
c:> dcelogin cell_admin
Enter Password: <NEW PASSWORD: not-echoed password>
```

5. Run the dcecp command, and make cell_admin a member of the groups listed below:

```
c:> dcecp
dcecp> group add acc-admin -member cell_admin
dcecp> group add subsys/dce/sec-admin -member cell_admin
dcecp> group add subsys/dce/cds-admin -member cell_admin
dcecp> group add subsys/dce/dts-admin -member cell_admin
dcecp> group add subsys/dce/dfs-admin -member cell_admin
dcecp> quit
```

6. Exit from your current session, and log in again as cell_admin:

```
c:> exit
c:> dce_login cell_admin
Password:
c:>
```

7. Update the ACL entries for all the directories and objects with the following command:

```
c:> dcecp
dcecp> set  x [exec cdsli -oRdc]
dcecp> foreach i $x {acl modify -entry $i -add {user cell_admin rwdtc}}
```

8. Run the dcecp command with the registry stop subcommand to stop the security daemon, which is still running in maintenance mode in the other window:

```
c:> dcecp
dcecp> registry stop /.:/subsys/dce/sec/master
dcecp> quit
c:>
```

9. Restart secd from the DCE Admin window by clicking on the **DCE Start** icon.

6.5.3 Adding a New Cell Administrator

A new cell administrator can be added to the system with the same rights as the original one. You may want to have two administrators or delete the old one afterwards.

6.5.2, “Cell Administrator Accidentally Removed” on page 208, shows how cell_admin is defined when it is first created. More definitions or permissions might be there depending on the complexity of the distributed environment.

To find out all the current rights of cell_admin, you can run the get_info_user cell_admin command. See 7.5, “Mass User/Group (and ACL) Management” on page 271, for more information about the user-management tools. The user definition file (UDF) created by the get_info_user command can also be used to create the *new_admin* account.

6.5.3.1 Creating a New cell_admin Account on AIX 4.1

To add a new cell administrator user on AIX 4.1, do the following:

1. Log in to DCE as cell_admin.
2. Add the *new_admin* user:

```
#dcecp
dcecp> user create new_admin -uid 1000 -gro none -org none \
> -passw secret -mypwd cel_admin_passwd
```

3. Run the dcecp command, and make *new_admin* a member of the groups listed below:

```
#dcecp
dcecp> group add acc-admin -member new_admin
dcecp> group add subsys/dce/sec-admin -member new_admin
dcecp> group add subsys/dce/cds-admin -member new_admin
dcecp> group add subsys/dce/dts-admin -member new_admin
dcecp> group add subsys/dce/dfs-admin -member new_admin
dcecp quit
```

4. Create ACL entries for the following CDS objects:

```
#dcecp
dcecp>s et x [exec cdsli -oRdc]
dcecp> foreach i $x {acl modify -entry $i -add {user cell_admin -rwdtc}}
dcecp>quit
```

From now on, *new_admin* has the same rights as *cell_admin*.

6.5.3.2 Creating a New cell_admin Account on OS/2 WARP

To add a new cell administrator user on OS/2 Warp, do the following:

1. Log in as *cell_admin*:

```
c:>dcelogin cell_admin
```

2. Add the *new_admin* user:

```
c:> dcecp
dcecp> user create new_admin -uid 1000 -gro none -org none \
> -passw secret -mypwd cel_admin_passwd
```

3. Run the dcecp command, and make *new_admin* a member of the groups listed below:

```
c:> dcecp
dcecp> group add acc-admin -member new_admin
dcecp> group add subsys/dce/sec-admin -member new_admin
dcecp> group add subsys/dce/cds-admin -member new_admin
dcecp> group add subsys/dce/dts-admin -member new_admin
dcecp> group add subsys/dce/dfs-admin -member new_admin
dcecp quit
```

4. Create ACL entries for the following CDS objects:

```
c:> dcecp
dcecp> set x [exec cdsli -oRdc]
dcecp> foreach i $x {acl modify -entry $i -add {user cell_admin -rwdtc}}
dcecp> quit
```

From now on, *new_admin* user has the same rights as *cell_admin*.

6.6 Integrating an NFS/NIS Environment

The Network Information System (NIS) and Network File System (NFS) are network services that were developed and introduced in 1985 by Sun Microsystems. NIS provides a distributed database system for common configuration files. NIS servers manage copies of the database files and NIS clients request information from the server instead of looking them up in their local copies of the files. For example, /etc/hosts can be managed by NIS. NIS servers manage copies of the information contained in the /etc/hosts file. All

NIS clients ask these servers for TCP/IP address information instead of consulting their local /etc/hosts file.

NFS is a distributed file system. An NFS server has one or more file systems exported that may be mounted by clients. To the NFS client, these file systems look like local file systems. Although NFS works with NIS, it can also be used separately. DCE/DFS serves the same purpose as NFS/NIS, but has many advantages. However, DFS is relatively new and is just about to establish itself as a standard platform for file sharing in client/server (C/S) environments. Many customer installations today use NFS/NIS to store common configuration files or to build RPC-based client/server applications or a distributed file system.

In order to convince customers to purchase DCE, we must show them one or both of the following ways to deal with the established NFS/NIS environment:

- How to integrate NFS/NIS into DCE/DFS
- How to migrate from NFS/NIS to DCE/DFS

The purpose of this section is to discuss these two issues. In most of the cases where DCE/DFS is introduced, we will see both steps. DCE/DFS-capable platforms could be migrated, whereas other platforms would continue to use NFS but with transparent access to DFS. This scenario determines the logical sequence of our subsections:

1. Migrating from NIS Domains to DCE cells
2. Migrating users from NIS to DCE
3. Migrating NFS file systems to DFS
4. Configuring NFS to DFS access

6.6.1 Migrating from NIS Domains to DCE Cells

NIS can centrally manage configuration files usually needed on each single system. Files like, for instance, an /etc/passwd file are present on each system, but they contain an escape sequence that directs the lookup call to a central file. These configuration files managed by NIS are converted into keyword and value pair tables called maps. You can look up these maps with the command `ypcat`. If you enter, for example, `ypcat hosts`, you concatenate your local /etc/hosts file with the database information about hosts and display them as if you were displaying a regular /etc/hosts file.

If you are using NIS within your environment, you may want to migrate NIS information over to DCE. Considerations for this migration are discussed in this chapter.

First you must evaluate the maps administrated by NIS. Following is a list of possible maps:

- /etc/groups
- /etc/passwd
- /etc/aliases
- /etc/hosts
- /etc/protocols
- /etc/services
- /etc/rpc
- and possibly more, specific to each customer

Once you have this list, decide what network information is important to have commonly available within your cell. There are maps which have to be treated differently. The following list gives you some ideas how to manage them:

- /etc/hosts

  The map of /etc/hosts should be migrated to the Domain Name Service (DNS) standard of the Internet. This allows you also to go for intercell communication later on.

- /etc/passwd and /etc/group

  The password and the group file information is managed in the DCE security registry, one of the core pieces of DCE. This version of DCE (AIX/DCE 2.1) supports the single login. It means that DCE login and AIX login can be integrated together. You need to log in to DCE *and* AIX in one step, giving the DCE password. As with NIS, users need not be defined in the passwd files of the AIX machines, and before machines can be integrated, their file ownerships must be adjusted to the global NIS or DCE user IDs. See 7.4.2.3, “Synchronized Users” on page 268, for a discussion about issues of defining users for integrated login.

  In 6.6.2, “Migrating Users from NIS to DCE” on page 215, we describe how to populate the DCE user registry database from NIS maps.

- Other configuration files such as /etc/services, /etc/rpc and others

  All the other configuration files listed above can be managed either with DCE/DFS or with objects in the namespace. This is described in the rest of this section.

In a network, there is always data which must be consistent and well known among all the connected systems. The /etc/services or /etc/protocols files are two examples of such files. Within DCE, there are several ways to provide the network consistency of such files.

Since these files build part of the definitions for TCP/IP, which itself is an enabling layer for DCE, the files cannot simply be in DFS. They would not be available when TCP/IP needs them. What this means is they need to be local. Our approach is to find a way with DCE to keep them synchronized on all the systems.

6.6.1.1 Distribution via Binary Distribution Machine (BDM)

BDM is a feature provided by DFS to update common files within the network. A BDM server machine running an upserver process is listening for client machines running upclient processes. The upclients pull files from the BDM. The files need to have the exact same full-path name on all systems. If this is not the case for certain files, another family of upserver/upclients can be defined.

Files like /etc/services or /etc/protocols can be maintained on a central system running a BDM. By default, the upclient process on each involved machine checks its BDM for new (or different) versions of certain predefined files every five minutes; if it finds new versions, it automatically copies the files to its local machine.

For more detailed information, consult *The Distributed File System (DFS) for AIX/6000* redbook or the AIX DFS documentation.

6.6.1.2 Distribution via DFS Namespace

You can maintain the files on one machine and copy them to the DFS namespace. All other systems use the same mechanism to copy the files from DFS to their local file system.

The copy procedure can compare the versions by checking the modification time and copy only if something has changed. This procedure would have to be executed in regular intervals controlled by cron.

This is a very simple but effective way of being consistent within the network.

6.6.1.3 Distribution via CDS Namespace

Although it is not recommended to store data in the CDS namespace, there is the possibility of keeping information that has to be consistent throughout the cell in a CDS object. Following is an example for the file /etc/services:

1. Add a new attribute to the file /opt/dcelocal/etc/cds_attribute:

   ```
   echo "1.3.22.1.3.60  CDS_TCPIP_SERVICES  char" >> /opt/dcelocal/etc/cds_attribute
   ```

   This defines the attribute CDS_TCPIP_SERVICES of *character* type to be used within your namespace. The /opt/dcelocal/etc/cds_attribute is also a file which needs to be consistent on all the systems within the cell.

2. Now you are ready to enter the information to be distributed into the object. For example, the following entries of /etc/services:

   ```
   domain    53/tcp   namserver  # domain name server
   domain    53/udp   namserver
   ```

   would be entered into CDS with

   ```
   # dcecp> object create /.:/services -attribute {
   >{CDS_TCPIP_SERVICES domaine:53/tcp:nameserver} \
   >{CDS_TCPIP_SERVICES domaine:53/udp:nameserver}}
   ```

   The colons (:) are used as field delimiters. Process each entry of /etc/services in this way.

3. Once you have filled in all these attribute entries into the object, they are available for every system in your cell. The systems now need a script which must frequently check the content of the object for updates and if necessary update the local /etc/services. Use cron to execute this check at regular intervals.

We cannot tell you which method is the best for you. Each case needs to be analyzed separately. However, you must always keep in mind that you should not fill up your CDS database with too much non-DCE-relevant data. Clearinghouses are not designed to be used as general purpose databases but to provide important binding information to your network application.

6.6.2 Migrating Users from NIS to DCE

As explained above, NIS users are centrally managed in a passwd map. The passwd map can be looked up by entering the command:

```
# ypcat passwd
```

The output of this command has the same format as if you were displaying a local /etc/passwd file

```
# cat /etc/passwd
```

To migrate users from NIS to the DCE registry database, we use mainly our user-management tools as described in 7.5, “Mass User/Group (and ACL) Management” on page 271.

All we must have is a small shell script `nis2dce_users` that reads the information from `ypcat passwd` and transforms the entries into UDF format (user definition file) for use by our `add_users` and `enable_users` procedures. For explanations on how this script works, we include a listing in 6.6.2.3, “The nis2dce_users Procedure” on page 217.

There is one important issue when migrating from any environment to DCE: unique user IDs and group IDs (UIDs/GIDs). Most likely you will have to unify UIDs and GIDs when introducing DCE in a previously unorganized environment of single workstations. Even in an NFS/NIS environment, it might be necessary to unify UIDs/GIDs first when multiple NIS domains are migrated into one DCE cell or when NIS and DCE have to be merged because a DCE cell is already there. So we have to look at two cases:

1. Unifying UIDs/GIDs and adjusting all their properties before the migration
2. Moving user accounts and groups straight into DCE

6.6.2.1 Unifying UIDs/GIDs and Adjusting File Ownerships

As mentioned above, it might be necessary to make UIDs unique before they can be entered into the DCE registry. Before you can reassign UIDs/GIDs to existing users/groups, you must find out what resources they own or have access rights to. Remember the user and group names are just for the user’s or administrator’s convenience. Internally, everything is based only on the UID and GID, simply called ID hereafter. Candidates to look at are, for example:

- Files and directories
- Databases
- Configuration files that define access rights, such as:
  - .rhosts
  - /etc/exports

  These particular examples of configuration files use user or group names. As long as the IDs are consistently changed on all systems, these files need not be changed because the user names still have the desired effect.

There might be more subsystems in your environment that will be affected by a global ID change. We pick the most common case and show how ownership of files and directories need to be changed. It is a somewhat tedious task, but it has to be done sooner or later. Otherwise, you will become very confused when you start to deal with DFS Access Control Lists and DCE IDs do not match AIX IDs.

The following is the generalized procedure to perform global ID changes on files and directories:

1. Find out which subsystems will be affected as outlined above.
2. Create a list of existing UIDs/GIDs in DCE and all other repositories.
3. Create a cross reference list that shows which existing IDs need to be changed into what target DCE ID.

4. Check whether on each individual system one of the target IDs is already in use by another user. If this is the case, the other user needs to be moved away from that target ID first.

   This means: Sort the cross reference list so that you do not inadvertently lump together files of different owners to one UID. You might have to create intermediate UIDs if there are too many mutual dependencies.

5. Start global changes from the top of your sorted cross-reference list, one at a time on each involved system:

   ```
   find / -user <old_UID> -print | xargs chown <new_UID>
   ```

   Caution: be sure that <new_UID> is not in use (anymore)!

The same steps need to be followed for the group IDs (GIDs). Then create the DCE user accounts as outlined below.

6.6.2.2 Moving User Accounts and Groups into DCE

Migrating users and groups from an existing environment means extracting their existing user account and group information and put them into UDF and GDF format files so they can be treated with our DCE user-management tools. See 7.5, “Mass User/Group (and ACL) Management” on page 271, for explanations on UDF/GDF and the tools.

Assuming you installed the user-management scripts in directory /umgt and directory *temp_users* does not yet exist, you perform the following steps:

```
# cd /umgt
# mkdir dce_users
# nis2dce_users temp_users
# nis2dce_groups temp_groups
```

Since you have checked all UIDs/GIDs for duplications and fixed possible problems beforehand, you can now copy all the files from the temporary repository to the DCE repositories dce_users and dce_groups and add the users and groups:

```
# dce_login cell_admin <passwd>
# add_groups all
# add_users all
# rgy_enable_users all
```

The nis2dce_users script is shown below. It could easily be modified for use with other environments. A procedure, pwd2dce, is also provided with this publication.

6.6.2.3 The nis2dce_users Procedure

This procedure reads the passwd NIS map and writes a user definition file (UDF) for each user. It uses the READ_UDF and WRITE_UDF scripts. See 7.5, “Mass User/Group (and ACL) Management” on page 271, for information about the user-management tools.

In READ_UDF, you will see what variables can be set. READ_UDF assigns the default values to the new user file. Then WRITE_UDF writes the upper part of the file and eventually writes the extracted values to the end of the file as ADD instructions.

You might want to change the script before you run it, for instance to exclude old home directory information. You should inspect all UDFs before you move them to the repository dce_users. They might contain information that you want to change, add, or remove.

```
#!/bin/ksh
# This script extracts NIS records and write a UDF file
# for each user
# $1 must be a new repository

dir=$1

if [ $# = 0 ]
then
   echo "Usage:  nis2dce_users <new_repository>"
   exit
fi

if [ -d "$dir" ]
then
   echo "Directory $dir already exists, specify a new directory"
   exit
fi
mkdir $dir


# Read the NIS passwd map
ypcat passwd  |  {
   while read input
   do
       user=`echo $input | cut -f1 -d:`
       echo "Writing UDF for user $user ..."
       . READ_UDF $user $dir
       . WRITE_UDF $user $dir NEW define_udf
       echo "ADD_uid=`echo $input | cut -f3 -d:`" >> $dir/$user
       GROUPID=`echo $input | cut -f4 -d:`
       GROUP=`ypcat group | grep :$GROUPID: |cut -f1 -d:`
       echo "ADD_newgrp=$GROUP"  >> $dir/$user
       echo "ADD_gecos=`echo $input | cut -f5 -d:`" >>  $dir/$user
       echo "ADD_homedir=`echo $input | cut -f6 -d:`" >>  $dir/$user
       echo "ADD_initprog=`echo $input | cut -f7 -d:`" >>  $dir/$user

       for p in `ypcat group | grep $user |grep -v :$GROUPID:` ; do
          MEMBER="$MEMBER`echo $p | cut -f1 -d:` "
       done
       if [ -n "$MEMBER" ]
       then
         echo "ADD_groups=$MEMBER"   >> $dir/$user
       fi
       MEMBER=""
   done
}

echo "All files are created in directory $dir"
echo "Inspect them before you move them into the repository dce_users\n"
```

6.6.3 Migrating NFS Files to DCE/DFS

Before you move any files into DFS, make sure you have unique UIDs/GIDs. If you had to make them unique, be sure to have also changed the ownership of files and directories to the new UIDs/GIDs as outlined in the previous chapter.

Note

If you continue with mismatches between AIX/NFS UIDs and DCE UIDs, you will always become confused about actual file ownerships and access permissions.

Even though it might be a very tedious job to unify UIDs/GIDs, do it now to save you a lot of trouble later on.

Before you can move the files to DFS, the framework of directories with mount points for filesets should be in place. This step actually needs a lot of design work because, for performance reasons, it probably makes sense to create the filesets in different aggregates and on different file servers. Please refer to Chapter 5, “Implementing Various LAN/WAN Scenarios” on page 105, for tips and guidelines on designing and implementing DFS.

Moving files from AIX or NFS to DCE is a simple task, but you have to be aware of how ACLs are assigned or inherited when the files are created. We would like to give a short description on DCE ACL inheritance before we show how to copy the files.

6.6.3.1 DFS ACL Inheritance

ACL inheritance is the method by which a file or a directory is given an ACL when it is created. Certain values, we call them Initial Creation ACLs, are defined on the parent directory and are passed on to new files and directories within that directory. For files, the values that are passed are the Initial Object Creation (IOC) ACL. Directories receive the Initial Container Creation (ICC) ACLs for themselves, plus they store the IOC and ICC to further pass them to files or directories underneath.

DFS considers three things when it creates the ACLs for a new object or directory:

1. The AIX mode bits specified with the system call that creates a file or directory.

   For example, a command such as `touch` or redirection of output into a file uses the open() system call with permissions `rw-rw-rw-` when creating a new file without a umask set. If the file is already there, permissions are not changed. Commands like `cp` or `tar` use the permission of the source file if they are creating a new file. If the file is already there, permissions are not changed.

2. The Initial Creation ACL set for the parent directory.
3. The umask attribute of the process creating the file.

If Initial Creation ACLs are set, then items 1 and 2 will be used for the new object. On the entries user_obj, group_obj, mask_obj, and other_obj, which are directly related to the user/group/others permission sets in AIX, the more restrictive set is applied (AND operation).

If the parent directory does not have Initial Creation ACLs defined, the umask is used to possibly further restrict the permission bits as explained in item 1. In this case, an ACL for the newly created file or directory will not be created, and the protection will be only by AIX mode bits.

Note: It may seem that the Initial Creation ACLs are always set when you list them with the `acl_edit` or `dcecp` command. However, if you have not explicitly set them, they are not there, and what you see is the umask which is interpreted by `acl_edit` or `dcecp`.

But how do we know whether the Initial Creation ACLs are set or not?

The following procedure shows a way to test whether ACLs are on the directory /:/testdir:

```
ev2 (/) # cd /:
ev2 (/:) # dcecp -c acl show -io testdir
{user_obj rwxc--}
{group_obj r-x---}
{other_obj r-x---}

ev2 (/:) # umask
022
```

The umask 022 means that write permissions for group and others are masked out. So the above ACL corresponds to the umask, which makes it likely that ACLs have not been set. To verify this, we need to change the umask with the `umask` command and list the ACL again:

```
ev2 (/:) # umask 0
ev2 (/:) # umask
00
ev2 (/:) # touch testdir/t
ev2 (/:) # rm testdir/t
ev2 (/:) # dcecp -c acl show -io testdir
{user_obj rwxc--}
{group_obj rwx---}
{other_obj rwx---}
```

The fact the Initial Object Creation ACL is changing along with the umask indicates that the ACL has not been set. Touching a file seems to be necessary for `dcecp acl` to reflect the new umask value.

We recommend setting the Initial Creation ACLs on the directory that will be the top directory for the new file subtree. Furthermore, you should set ACLs on all underlying fileset mount points because ACL inheritance does not go across fileset boundaries. In this way, you make sure all new files and directories created in the future will also receive ACLs.

If you do not want to set the ACL, you should check whether ACLs are already set or whether the umask will be in effect. If the latter is true, no ACLs will be set, and the umask will also in the future be used when new files or directories are created.

6.6.3.2 Moving the Files

Before you copy the files over to DFS, you should have:

- Unified UIDs/GIDs
- Adjusted file ownerships to the new UIDs/GIDs at the old location
- Created the DFS fileset framework
- Set Initial Creation ACLs on all filesets

This is just a summary of the steps explained above. Now that you are prepared, you can use any copy method that allows preserving of file ownership and permissions, for example:

- `cp -pr`
- `rcp -pr`
- `tar -xpvf`

Assume we want to move the /home file system with all users' home directories to /:/dfshome. To be able to better control the users' resources and quotas, we want to give each user their own fileset. The following generalized example outlines the necessary steps:

1. Log in as root.
2. Run `nis2dce_user` and `nis2dce_groups` and check whether you need to adjust any IDs.
3. Now is the last opportunity to adjust UIDs/GIDs and file ownerships.
4. Create the archive:

   ```
   # cd /home
   # tar -cvf/dev/rmt0 .
   ```

5. Log in to DCE as cell_admin.
6. Add the new groups to DCE with `add_groups`.
7. Add the new users to DCE with `add_users`.
8. Now is the last opportunity to create all the necessary filesets and target directories that will serve as mount points.
9. Check all DFS filesets for availability.

   ```
   # cd /:/dfshome
   # cd /:/dfshome/user1
   # cd /:/dfshome/user2
   ```

10. Check, or better, actually set the Initial Creation ACLs on *all* involved filesets such as /:/dfshome/user1 and so on. Remember that ACLs are not passed across mount points.
11. Restore the files:

    ```
    # cd /:/dfshome
    # tar -xpvf/dev/rmt0
    ```

It is necessary to be root and cell_admin in order to preserve file ownership upon creating the new DFS files. The `-p` flag overrides the umask or Initial Creation ACLs that might be in use and sets the mode bits as they were defined on the old files.

Note: It is nevertheless important to set the Initial Creation ACLs because we want the IOCs and ICCs to be passed on to subdirectories. They are not overwritten by the `-p` flag. Also, all explicitly defined ACLs, other than user_obj, group_obj, other_obj, and mask_obj, are applied if defined.

6.6.4 Configuring DFS Access from NFS Clients

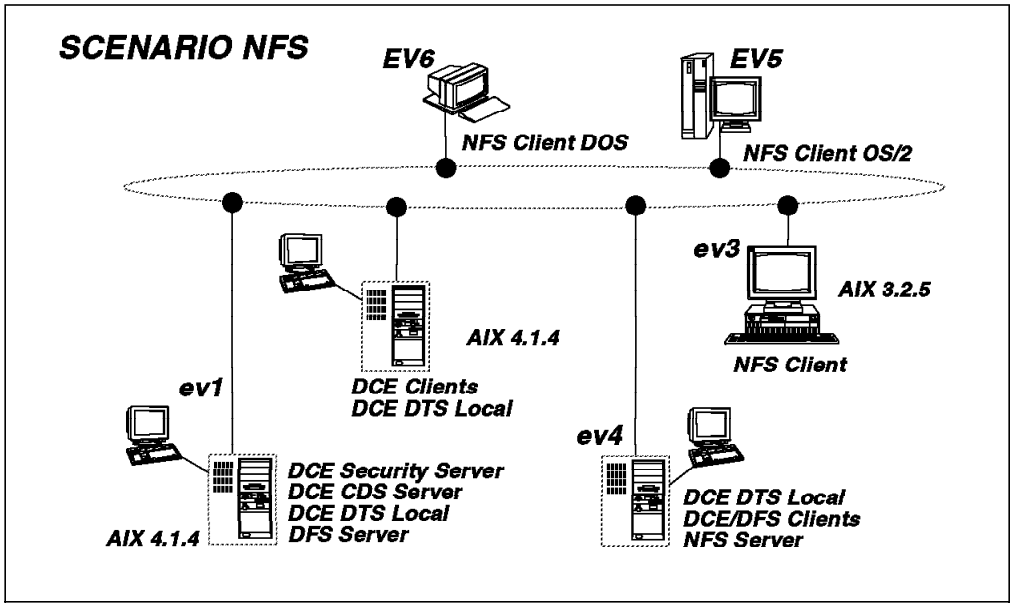


Figure 52. Scenario with Coexistence of NFS Clients and DCE/DFS

This is the task to configure step-by-step the NFS/DFS Translator and how to access the DFS file space remotely from NFS client machines. Before starting to configure DFS access from NFS clients, we suggest you read 7.3, “NFS-to-DFS Authenticating Gateway” on page 257, to understand the basics about the translator.

According to Figure 52, machine *ev4* is a DFS client machine and also houses the NFS/DFS Translator. The machine *ev3* is an NFS UNIX machine, *ev5* is an NFS OS/2 machine, and *ev6* is an NFS DOS/Windows machine. Suppose we want to export a DFS directory, */:/dfshome/brice*, to the NFS server and that the user account *brice* exists on *ev4* (in /etc/passwd) and in the DCE registry database. The directory /:/dfshome/brice is a mount point for fileset *hbrice.ft* and is the home directory for user brice.

See 4.5, “Defining Home Directories in DFS” on page 98, for tips how to define a user’s home directory in DFS.

Here is an overview over the steps we will perform and describe in this section:

- Applying ACLs to a directory before exporting
- Configuring and starting the NFS/DFS Translator on a DFS client machine, on *ev4* in our case
- Exporting a directory
- Registering the authentication mappings
- Mounting a DFS remote directory via NFS to a local one

6.6.4.1 Preparation Steps

See 3.2, “Preparing for DCE Configuration on AIX” on page 38. To install the DCE cell follow the configuration steps exactly in 3.4, “Configuring the Initial DCE Servers and Clients on AIX” on page 43, for machines *ev1, ev2,* and *ev4*.

Install and test TCP/IP and NFS on the OS/2 and the DOS/Windows machines. Please follow the appropriate system documentation.

6.6.4.2 Creating and Mounting the Filesets for Home Directories

1. Create a logical volume /dev/dfshome with five blocks of 4 MB:

   ```
   # mklv -y'dfshome' rootvg 5
   ```

2. Create an aggregate on the /dev/dfshome:

   ```
   # newaggr -aggreg /dev/dfshome -bl 8192 -fr 1024
   ```

3. Export the aggregate:

   ```
   # mkdfslfs -d /dev/dfshome -n dfshome
   ```

4. Log in to DCE as cell_admin.

5. Create the dfshome fileset with mount point:

   ```
   # mkdfslfs -f dfshome.ft -m /:/dfshome -n dfshome
   # chmod 0775 /:/dfshome
   # dcecp -c acl modify /:/dfshome -add {user cell_admin rwxcid}
   ```

   Adding the ACL entry for cell_admin effectively creates the ACL, and with it a mask_obj entry that is derived from the group permission set of /:/dfshome. In order for cell_admin to be able to insert new objects (including fileset mount points) underneath /:/dfshome, the mask_obj must contain insert permission. This had been achieved by giving *group* write access with chmod 0775. Had we used chmod 0755, would cell_admin not get insert permission because the mask_obj would restrict its permission set.

6. Create Brice's fileset with a mount point in the same aggregate:

   ```
   # mkdfslfs -f hbrice.ft -m /:/dfshome/brice -n dfshome
   # chmod 0755 /:/dfshome/brice
   ```

   Here, there is no need for *group* having write permission, unless we want to give other users write access to Brice's home directory. Another way to allow for that would be to modify the mask_obj directly after is has been created.

7. See if the filesets are correctly exported:

   ```
   # fts lsfldb
   ```

8. Make Brice the owner of the new fileset:

   ```
   # chown brice.staff /:/dfshome/brice
   ```

6.6.4.3 Applying ACLs to a Directory

To make sure that this directory is protected, user brice has to apply ACLs on it if it is not already done. Normally, this step is to be done by the owner of the directory. However, since most users won't ever touch ACLs, a DFS administrator or cell_admin has to define the initial ACLs for the home directories.

1. Log in as DCE brice principal on *ev4*:

   ```
   $ dce_login brice brice_passwd
   ```

2. Apply (add or modify) ACLs on the directory /:/dfshome/brice to the values you want to be automatically set for every new file or directory:

   ```
   $dcecp
   >dcecp acl modify /:/dfshome/brice -add { ... }
   ```

   Inherited rights when you create another subdirectory:

   ```
   $dcecp
   >dcecp acl modify -io /:/dfshome/brice -add { ... }
   ```

   Inherited rights when you create files underneath this point in the file tree:

   ```
   $dcecp
   >dcecp acl modify -ic /:/dfshome/brice -add { ... }
   ```

   The job of user brice stops here for the moment. He has to wait for the NFS/DFS Translator to be started and for the directory to be exported to NFS.

6.6.4.4 Configuring and Starting the NFS/DFS Translator on ev4

This step has to be executed by a UNIX system administrator. Be sure that:

- The DFS client is configured and running on the machine
- The NFS server is running on the machine

Start the NFS/DFS Authenticating Translator via SMIT:

```
# smitty dce
   -> Configure DCE/DFS
      -> NFS/DFS Authenticating Translator Administration
         -> Start NFS/DFS Authenticating Translator
            (fastpath = dfsnfs)
```

Press **Enter**. Then the following message is shown:

```
The NFS to DFS Authenticating Translator has been started successfully.
```

Instead of using SMIT, you can also simply start the NFS/DFS authentication from the command line:

```
# /etc/rc.dfsnfs
```

If you want to automatically start the NFS/DFS Authentication Translator, you have to put this command into the /etc/inittab file.

6.6.4.5 Exporting the DFS Directory to NFS

This task needs to be done by a UNIX super user.

Call SMIT and follow the indicated path, or call SMIT with the fastpath name:

```
# smit nfs_menus
   ->  Network File System (NFS)
      -> Add a Directory to Exports List
         (fastpath = dfsnfsexp)
```

```
                        Add a Directory to Exports List

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                [Entry Fields]
* PATHNAME of directory to export               [/:/dfshome/brice]
* MODE to export directory                       read/write              +
  HOSTNAME list. If exported read-mostly        []
  Anonymous UID                                 [-2]
  HOSTS allowed root access                     []
  HOSTS & NETGROUPS allowed client access       [ev3,ev5,ev6]
  Use SECURE option?                             no                     +
* EXPORT directory now, system restart or both   both                   +
  PATHNAME of Exports file if using HA-NFS      []
```

Check if the directory is exported correctly by using exportfs command:

```
# exportfs
```

At this point, the directory is exported to the NFS clients. Nevertheless, NFS client users cannot yet access this DFS fileset until user brice has registered his authentication mapping on the NFS/DFS Translator site.

6.6.4.6 Registering Your Authentication Mappings on ev4

As we previously explained: Before a directory can be mounted for authenticated DFS access by an NFS client, a DCE principal must register his authentication mapping on the NFS/DFS Translator site. This task can be done either by user brice who has a UNIX account and a DCE account or by a UNIX super user who knows brice's DCE password. We suppose here that user brice is doing that himself. User brice is still logged into AIX and DCE on *ev4*.

The command to use on the NFS/DFS Translator site is:dfsiauth, for example:

```
$ dfsiauth -add -r ev3 -i 107 -u brice -p brice_passwd
$ dfsiauth -add -r ev5 -i 107 -u brice -p brice_passwd
$ dfsiauth -add -r ev6 -i 107 -u brice -p brice_passwd
```

Check if the registering is done correctly:

```
$ dfsiauth -list

Host   Uid     Principal     @sys    @host     Expiration
----  ------  ----------     -----    -----     ----------
ev3     107      brice                       5/27/94  00:30
ev5     107      brice                       5/27/94  00:30
ev6     107      brice                       5/27/94  00:30
```

6.6.4.7 Mounting the Directory

Before trying to mount the remote directory, make sure that TCP/IP and NFS are running on all machines.

1. On a UNIX machine (*ev3*), enter:

   ```
   #mount -v nfs -n ev4 /:/dfshome/brice /u/brice
   ```

2. On an OS/2 system (*ev5*), enter:

   ```
   c>mount E:  ev4:/:/dfshome/brice
   UID: 107
   GID: 100
   ```

3. On a DOS/Windows system (*ev6*), enter:

   ```
   c>mount E: ev4:/:/dfshome/brice
   ```

   We assume the disk unit *E:* is defined on the DOS system.

6.7 Managing Remote Servers

In the DCE cell, many platforms and machines are tied together. Some have server roles (replica, master, secondary); others are only clients. However, the preallocation of roles discussed in this book so far is only valid for the *core* services, where the functions have been distributed by the cell administrator during the cell configuration. The *functional applications* that also use the client/server pattern are freely distributed among the different machines. *Application servers* could reside on machines that have been configured as DCE (core) clients, and vice versa.

In this section, we will discuss:

1. A functional overview on RPC and the involved DCE components
2. The DCE daemon and its managed objects
3. How the availability of remote services can be checked
4. How remote services can be started and stopped
5. The management of hostdata objects
6. The management of keytab objects

6.7.1 DCE RCP Applications: Functional Overview

DCE offers a high degree of flexibility to create client/server applications. An application can be called a DCE client/server application when it uses DCE RPC (remote procedure calls). Although we consider *Security* and *Directory* Services an integral part of the DCE functions, it is, in some circumstances, possible to have a client/server connection for applications without those services. This situation should be avoided because these services are the primary values you get when using DCE, but in a testing situation, this could be acceptable.

When an application client contacts an application server, it goes through a *binding process*. Depending on the way this binding is realized, the impact of the core services will be more or less important.

- Using a *string binding*, the client can use a server without contacting the CDS and even without security authentication. The client has to know the binding elements to compose the *binding handle* from a string. The elements to know are the IP address and the network transport protocol. Using this *explicit binding* method, you do not even have to define a DCE cell. Of course, the explicit binding method can also use the services of CDS and security, which is usually the case.
- Another method of binding is to import the binding elements, usually a *partial binding handle*, from the CDS server. Obviously, in this case, the DCE core services have to be available. This can be done completely automatically and under the covers so that a client program does not even know it is using a binding handle. This is called *automatic binding*. Accessing the CDS information also requires some authentication of the client. This authentication can be implicit or explicit.

The (binding) information needed by an RPC client includes the following:

1. The interface UUID and version number obtained from the client stub

2. The IP address and protocol obtained in one of the following ways:
    - Hard-coded in the program
    - As a parameter passed to the program at startup
    - Extracted from the CDS if the server was exported using the `rpc_ns_binding_export()` call.
3. An optional object UUID obtained in one of the following ways:
    - Hard-coded in the program (but that seems rather painful)
    - Extracted from the exported information in CDS

The DCE daemon (dced) on the server machine maintains an *endpoint map*. Application servers may register their interface(s) with the endpoint mapper function of the dced and get an endpoint assigned. The dced acts as a *portmapper* and listens to the well-known port 135. Figure 53 shows the components involved for a successful server binding.

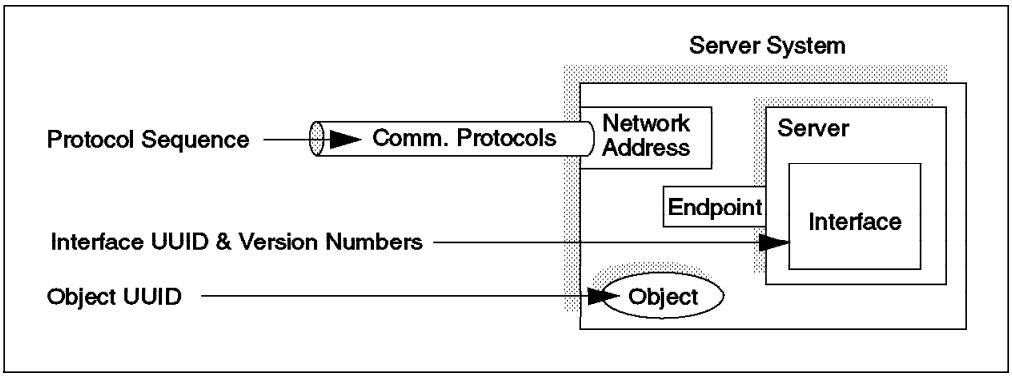


Figure 53. Information Used to Identify and Access a Compatible Server

Usually, the first call of a client comes in with a *partially bound handle*, meaning it does not have an endpoint. Such a call gets to the dced, which completes the handle by adding the correct endpoint. The client can then directly call the server process with a *fully bound handle*.

If a client knows the endpoint and can create a string binding that includes the endpoint, not even the dced is necessary on the server side. Of course, that would be very unusual.

Usually, an operational client/server application needs to be secured. There are different levels of security available with DCE:

- Authentication of the client and server principals
- Access control
- RPC data security ranging from none to full data encryption

It is not the purpose of this section to discuss these options. The point here is that servers need to run under an authenticated principal if you want security between client and server. An interactive principal (a user) usually authenticates itself with the Security Service by providing its name and password. A server process, however, needs to be started automatically, without user intervention.

But how can a server provide a password?

A keytab object is maintained by every DCE daemon. The server can retrieve its (encrypted) password from the keytab of the machine it is running on. See 6.7.2,

"The DCE Host Daemon (dced)" on page 228, for information about the dced and 6.7.6, "Managing the Keytab" on page 236, for more details about the keytab and its management.

6.7.2 The DCE Host Daemon (dced)

The DCED daemon plays a more and more important role in the DCE cell. Known as the RPC daemon (rpcd), its function was strictly confined to the *endpoint mapping*, whereas now it includes the *sec_cl* (Security Service client) function that maintains the machine principal's credentials. It also manages and monitors a series of databases and tables, both static and dynamic, on each DCE server and client machine.

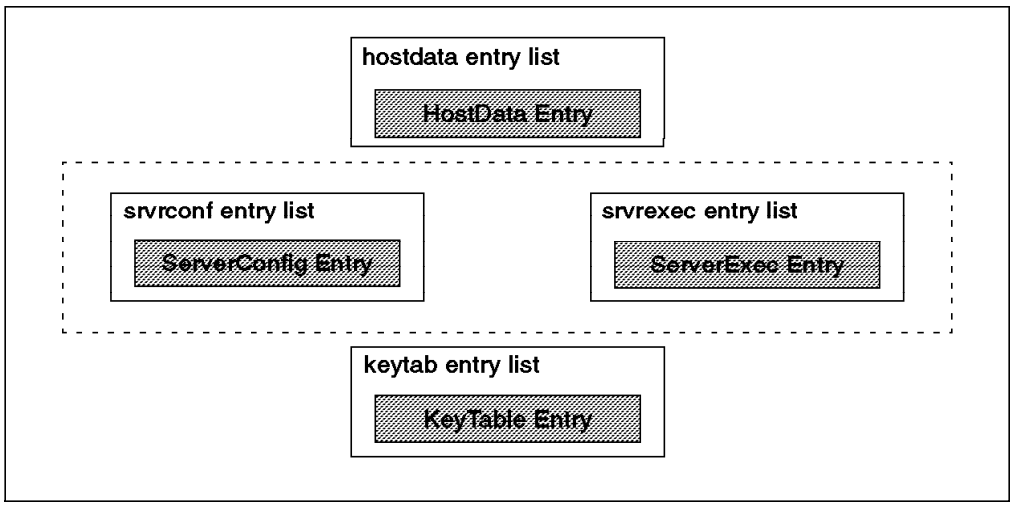


Figure 54. Objects Maintained by dced

Figure 54 shows the objects which are maintained by the DCE daemon. They currently are:

- **HostData** — HostName, cell name, and much more (see 6.7.5, "Working with the Hostdata" on page 235)
- **SrvrConf** — Information about all configured servers, including DCE core service daemons and user application servers if they have been entered in this database
- **SrvrExec** — Information about running servers, including both core and user servers
- **Keytab Info —** Keytab entries with information about the private keys (passwords) of application server principals

These dced objects keep status and can be queried and updated. The DCE daemon maintains its own junction in the CDS namespace so that all these objects have a representation in the CDS namespace and can be manipulated from remote sites. The junction is */.:/hosts/<hostname>/config*.

Depending on the site from where you access these objects, the service name has to be qualified:

- On the local host **—** *service*@hosts/*host* (Only from programming API)
- In the local namespace — /.:/hosts/*host*/config/*service*
- Global name — /.../*cell_name*/hosts/*host*/config/*service*

Programs can be written with a binding to all those different services in the dced. A dced binding is special; it is not just a binding to the dced but also to a well-defined, specific service maintained by the dced. A lot of manipulation can also be done via the dcecp shell. For example, to get a list of services enter:

```
$ dcecp
dcecp> host show /.:/hosts/EV5
{time_provider running}
{cdsadv running}
{dfsd running}
{dtsd running server}
```

Note: This command did not return correct results from AIX DCE systems at the time we were testing. It just returned an empty list.

6.7.3 Checking Availability of Remote Services

An adequately functioning DCE cell requires an activated presence of several layers of functions. Although we could split it up in many layers, for the sake of simplicity, we decided to limit it to four well distinct layers, which are shown in Figure 55 below.

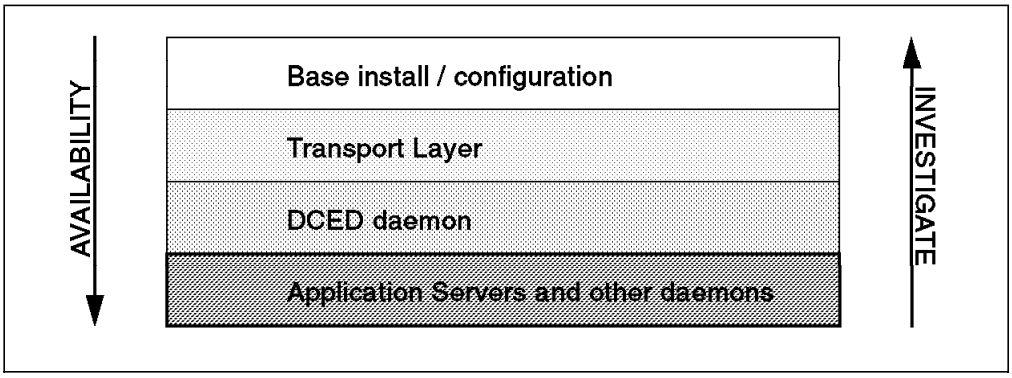


Figure 55. Availability Layers for DCE Applications

Availability of the DCE applications requires the layers in a *top/down* way, while the *investigation* of a problem should be tackled *bottom/up.* In this section, we discuss how the availability of these layers can be checked.

6.7.3.1 The Platform Base

Before any DCE task can be executed, DCE has to be installed and configured. To verify whether this task has been executed correctly, we have a few tools at our disposal.

The investigation can be done remotely if a remote shell (telnet) can be opened, or has to be done locally if this is not possible.

1. The customization of an AIX platform could be verified by commands like:
   - `lsdce` — Listing of the current state of the local DCE configuration
   - `lsdfs` — Current state of the DFS configuration
   - `ps -ef` — Listing of processes.

     To list all process with the search term *dce*, issue the following command:

```
ev1::/-> ps -ef | grep dce
root  7440     1   0 15:06:37      -  0:00 /opt/dcelocal/bin/cdsadv
root  8774     1   0 15:06:55      -  0:02 /opt/dcelocal/bin/dtsd -s
root 17094     1   0 15:05:57      -  0:57 /opt/dcelocal/bin/dced -b
root 17812  8130   0 15:14:01  pts/1  0:03 dcecp
root 18214     1   0 15:06:42      -  0:12 /opt/dcelocal/bin/cdsd
root 18658     1   0 15:06:08      -  0:07 /opt/dcelocal/bin/secd
root 21760     1   0 15:57:14      -  0:00 /opt/dcelocal/bin/bosserver
root 22018  7440   0 15:57:18      -  0:11 /opt/dcelocal/bin/cdsclerk -w
root 22276 21760   0 15:57:23      -  0:00 /opt/dcelocal/bin/upserver
root 22534 21760   0 15:57:23      -  0:02 /opt/dcelocal/bin/flserver
root 22792 21760   0 15:57:23      -  0:00 /opt/dcelocal/bin/ftserver
root 23050 21760   0 15:57:24      -  0:02 /opt/dcelocal/bin/repserver
root 23590     1   0 15:57:34      -  0:00 /opt/dcelocal/bin/dfsbind
root 23906     1   0 15:58:00  pts/0  0:00 /opt/dcelocal/bin/fxd
root 26744     1   0 15:58:00  pts/0  0:00 /opt/dcelocal/bin/fxd
root 27544     1   0 15:58:05  pts/0  0:00 /opt/dcelocal/bin/dfsd
```

To list all processes with search term *dfs*, enter:

```
ev1::/-> ps -ef | grep dfs
root 21416 18796   0 10:42:24      -  0:00 /opt/dcelocal/bin/upserver
root 21672     1   0 11:14:14      -  0:10 dfsbind
root 24008     1   0 11:14:22  pts/1  0:00 fxd -mainprocs 7 -admingroup
 subsys/dce/dfs-admin
root 26846     1   0 11:14:23  pts/1  0:00 fxd -mainprocs 7 -admingroup
 subsys/dce/dfs-admin
root 28264     1   0 11:16:57  pts/1  0:00 dfsd
```

- Other commands that could be used for DFS are:

  `/opt/dcelocal/bin/bos status -server servername`

2. On OS/2 Warp, the verification can de done with the following command:

   `showcfg.exe`

   This returns the status of configured and running components.

6.7.3.2 The Transport Layer

Between AIX and OS/2 Warp platforms, all DCE transport relies on TCP/IP or on AnyNet (using TCP/IP). Without a functioning transport layer, there is no successful functioning of a DCE cell.

It is quite easy to verify this layer with the `ping` command. Other tools that can be used to verify the correctness of the network definitions are, for instance, the `netstat` command or the `host` command to check proper name resolving. These and others have been covered sufficiently in previous chapters.

The point that we want to make here is that once you have *transport connectivity*, you will probably be able to open a *telnet* session into the other platform where you suspect some malfunctioning or a lack of configuration. With a look into the remote platform, you can verify the following points.

- Is the platform fully configured for its functions?
- Are the required daemons up (especially dced)?
- Are the required application servers running?

If the transport connectivity cannot be realized, we have to return to local platform investigations, as explained in the previous paragraph. To establish a

telnet session, a running telnetd daemon is required on the target platform, and we must be allowed to activate this remote shell into the target machine.

Although very convenient, a telnet session creates a security exposure because the password is submitted in clear text. So, if this is an issue, consider the options that the DCE daemon offers first, which are explained below.

6.7.3.3 The DCE Daemon

The DCE daemon is explained in more detail in 6.7.2, "The DCE Host Daemon (dced)" on page 228. This section concentrates on some commands you can use to check the availability of the DCE daemon and its services.

- A view of the endpoint mappings of the local hosts can be obtained with:

```
dcecp> endpoint show
{{object 032070a1-600e-11cf-b924-08005aceea02}
 {interface {006e0b04-0538-1d34-8366-0000c09ce054 1.0}}
 {binding {ncacn_ip_tcp 9.3.1.124 135}}
 {annotation {Server Execution}}}

{{object 032070a1-600e-11cf-b924-08005aceea02}
 {interface {006e0b04-0538-1d34-8366-0000c09ce054 1.0}}
 {binding {ncadg_ip_udp 9.3.1.124 135}}
 {annotation {Server Execution}}}

{{object 019ee420-682d-11c9-a607-08002b0dea7a}
 {interface {019ee420-682d-11c9-a607-08002b0dea7a 1.0}}
 {binding {ncacn_ip_tcp 9.3.1.68 2135}}
 {annotation {Time Service}}}
```

- List the services controlled by the dced of machine EV5:

```
dcecp> host show /.:/hosts/EV5
{time_provider running}
{cdsadv running}
{dfsd running}
{dtsd running server}
```

  Host EV5 has been DCE-configured with security, CDS, DFS clients, and a DTS server function. This information is kept in the *srvrconf.db,* while the *srvrexec.db* contains information about the currently running processes, if they have been exported to dced. By comparison of both tables, the contacted daemon on a particular host is able to return the information as indicated.

  Note: This command did not return correct results from an AIX DCE system at the time we were testing; it just returned an empty list. But it worked for querying OS/2 Warp systems.

- List the configured servers on a particular host (including user application servers):

```
dcecp> server cat /.:/hosts/EV5
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/time_provider
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/cdsadv
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/dfsd
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/dtsd
```

  Note the complete name of a configured server. The *srvrconf* refers to the static server configuration portion for an installed DCE server.

Note: This command did not work correctly for AIX DCE systems at the time we were testing; it just returned an empty list. But it worked for querying OS/2 Warp systems.

The information related to the so-called host services is kept in containers. The access to those containers is protected by the DCE Security Service with ACLs on the container objects. You can show, and eventually change, the access rights. The following `acl show` command displays the current ACL status.

```
dcecp> acl show /.:/hosts/EV5/config/srvrexec
{unauthenticated criI}
{any_other criI}

dcecp> acl show /.:/hosts/EV5/config/srvrconf
{unauthenticated criI}
{any_other criI}
```

It is important that the ACLs are correctly set to allow you the desired access to dced objects.

6.7.3.4 Application Servers

As mentioned in 6.7.3.3, “The DCE Daemon” on page 231 above, we need to define the user applications as dced objects, and then they can be monitored via the `server cat`, `server show`, `server ping` subcommands of dcecp from anywhere in the cell. These commands contact the remote dced.

6.7.4 Controlling Remote Core and Application Servers

In previous versions of DCE, a server had to be started manually on a platform. In DCE 2.1, additional starting/stopping capabilities are available:

1. The starting/stopping of servers can be remotely controlled if the server has been defined in the *srvrconf.db* of the host.
2. The start of servers can be automatically triggered by the client if the adequate definition has been created.

A client with partial binding arriving at the DCE daemon in search of a compatible server will first initiate a lookup of the dynamic table *srvrexec.db*, which contains a view of all active and exported servers/daemons. If a compatible server is found, dced can redirect the client’s request to that particular server.

If an active server is not found, the lookup extends into the static *srvrconf.db* table. If a compatible description is found here and it allows for automatic start (triggering), the server will be started, and again redirection can take place.

6.7.4.1 Entering Binding Information into CDS

Normally, a well-written application server exports its interface(s) into CDS (`rpc_ns_binding_export()` call) so that clients can use automatic binding methods and look up compatible servers in the directory. However, if we want to trigger the startup of a server by a client request, then the server has no chance to export anything beforehand. We must manually create the CDS entry:

```
DCECP> rpcentry create /.:/subsys/servers/SVRAPPL1
```

and complete it afterwards with the next command:

```
DCECP> rpcentry export /.:/subsys/servers/SVRAPPL1
-interface { d58ab008-b3c6-11ca-891c-c9c2d4ff3b52 1.0 } \
-binding [ncadg_ip_udp:9.3.1.124 }
```

To check the created CDS entry, run the following command:

```
dcecp> rpcentry show /.:/subsys/servers/SVRAPPL1
{d58ab008-b3c6-11ca-891c-c9c2d4ff3b52 1.0
  {ncadg_ip_udp 9.3.1.124}}
noobjects
```

6.7.4.2 Defining a Server to the DCE Daemon

To allow for application servers to be remotely started and stopped, the following two conditions must be met:

1. The server entry has to be created as in the following command (example on OS/2):

   ```
   DCECP-> server create
   /.:/hosts/EV5/config/srvrconf/SVRAPPL1 \
   -program {\util\server\bin\svrappl1.exe} \
   -principal {SVRAPPL1} \
   -entryname {/.:/subsys/servers/SVRAPPL1} \
   -starton {explicit failure auto } \
   -services {{annotation { automatically started server 1}} \
   {interface {d58ab008-b3c6-11ca-891c-c9c2d4ff3b52 1.0}}}
   ```

   Alternative Options for Command Entry

   Instead of using the interactive dcecp shell, the command could be entered from the command line with the dcecp -c command. It worked when we tried it, but it is error prone. We had rather enter it by a Tcl procedure, but in both AIX and OS/2 we got the following results:

   ```
   Error: msgid=0x1131f009 Too many arguments specified in command
   ```

   This was a bug in the release we were using, and it should be fixed in the release-level code.

2. The EP (endpoint, listening port) of an application server must be exported by the server's *server code* with the following call:

   ```
   rpc_ep_register( )
   ```

A server cat shows the presence of the new application server:

```
dcecp> server cat /.:/hosts/EV5
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/cdsadv
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/dtsd
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/time_provider
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/dfsd
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/SVRAPPLA
/.../itsc.austin.ibm.com/hosts/EV5/config/srvrconf/SVRAPPL1
```

Note: This command did not work correctly for AIX DCE at the time we were testing; it just returned an empty list. But it worked for querying OS/2 Warp.

After creating the application entry in the *srvrconf.db*, the entry can be queried with:

```
dcecp> server show /.:/hosts/EV5/config/srvrconf/SVRAPPL1
{uuid 939a4af4-65a2-11cf-a1f9-10005a4f4629}
{program c:\util\servers\bin\SVRAPPL1}
{arguments {}}
{prerequisites {}}
{keytabs {}}
{entryname /.:/subsys/servers/SVRAPPL1}
{services
 {{ifname {}}
  {annotation { automatic started server 1}}
  {interface {d58ab008-b3c6-11ca-891c-c9c2d4ff3b52 1.0}}
  {bindings {}}
  {objects {}}
  {flags {}}
  {entryname {}}}}
{principals /.../itsc.austin.ibm.com/SVRAPPL1}
{starton auto failure}
{uid 0}
{gid 0}
{dir {}}
```

The application server's *interface UUID* and *version* have to correspond with the *UUID* and *version* in the DCE RPC stubs. Note also the *starton* option set to *auto*, which allows for a *client-triggered startup* of this server.

Eventually, this entry can be deleted with:

```
DCECP>server delete /.:/hosts/EV5/config/srvrconf/SVRAPPL1
```

6.7.4.3 Starting and Stopping Remote Servers

Once defined as dced objects, all servers can be remotely started and stopped. If set up for client-triggered start-up, the first RPC request will start the server process. With following command, you can explicitly start a remote server:

```
DCECP>server start /.:/hosts/EV5/config/srvrconf/SVRAPPL1
```

A host show command will indicate the updated situation, including the application servers. This information is collected from a combination of the *srvrconf.db* and *srvrexec.db* containers:

```
dcecp> host show /.:/hosts/EV5
{dtsd running}
{time_provider running}
{SVRAPPLA notrunning}
{cdsadv running}
{dfsd running}
{SVRAPPL1 running}
```

We find *SVRAPPL1* running. To stop the same server, enter:

```
DCECP>server stop /.:/hosts/EV5/config/srvrexec/SVRAPPL1
```

Errors encountered

In the releases we were testing, we were not able to activate a server through a client request although we defined the `-starton auto` option.

When trying to stop *SVRAPPL1*, we received the following error message:

```
dcecp> server stop /.:/hosts/EV5/config/srvrexec/SVRAPPL1
Error: No Bindings
```

6.7.5 Working with the Hostdata

Another resource list that can be manipulated, and is located on each host, is the *hostdata* entry list. To have an idea of the contents of those entries, and also to know the object names as available through the dced junction in the CDS namespace, you can issue the following command:

```
dcecp> hostdata cat /.:/hosts/EV5
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/passwd_override
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/cds_globalnames
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/host_name
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/cell_name
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/cds_attributes
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/pe_site
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/cds-cache-info
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/krb.conf
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/dfs-cache-info
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/group_override
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/post_processors
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/cds.conf
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/cell_aliases
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/dce_cf.db
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/clock-synch-info
/.../itsc.austin.ibm.com/hosts/EV5/config/hostdata/svc_routing
```

The catalog of the *hostdata* shows the object names that can be used as in the following show command for the *pe_site* object.

```
dcecp> hostdata show /.:/hosts/EV5/config/hostdata/pe_site
{uuid 4bed6d84-c0d1-11cd-975f-0000c09ce054}
{annotation  {PE Site file}}
{storage C:\opt\dcelocal/etc/security/pe_site}
{hostdata/data
 {/.../itsc.austin.ibm.com dd1d96e4-654e-11cf-819a-10005a4f4629@ncacn_ip_tcp:9.3.1.68[]}
 {/.../itsc.austin.ibm.com dd1d96e4-654e-11cf-819a-10005a4f4629@ncadg_ip_udp:9.3.1.68[]}}
```

Here follows the result of the show command for the *dce_cf.db* object:

```
dcecp> hostdata show /.:/hosts/EV5/config/hostdata/dce_cf.db
{uuid 002c7cdf-4301-1d77-af44-0000c09ce054}
{annotation  {DCE name config file}}
{storage C:\opt\dcelocal/dce_cf.db}
{hostdata/data
 {cellname /.../itsc.austin.ibm.com}
 {hostname hosts/EV5}}
```

6.7.6 Managing the Keytab

A *keytab* file contains the keys for one or more server principals and must be located on the same host as the application servers. Keys for servers are analogous to passwords for human users. Keys play a major role in the authentication procedures. Server keys and user passwords have to follow the same change policy. This means that a server periodically has to generate a new key. For a human user, the password is memorized, for a server it is stored in a *keytab* file. This file must have restricted access by the local entities.

It is more complex for a server to change keys than it is for a human user. A server may have to maintain a history of its keys with version numbers. The passwords (keys) are maintained at two locations: the Security Server Registry and the keytab. A client having obtained a ticket to a server holds an encoded version of the server's key. If in the mean time the server changes its key, at the next contact between client and server, the key (stored in the ticket) would be outdated, except if a certain history of the keys is kept in the *keytab*. New tickets will contain the new updated secret key.

Between the two locations where the secret key is located (keytab, registry), the key has to be synchronized. If the key is specified in *plain* text, this synchronization could be done manually, but if the key is automatically generated (at random), the generation process has to update both sites.

6.7.6.1 DCED and the Keytab File

A series of dcecp commands can be used to maintain the keytab files:

`keytab create`	Creates keytab files and their entries
`keytab delete`	Deletes keytab files and their entries
`keytab add`	Adds key entries to an existing keytab file
`keytab remove`	Removes key entries from keytab files

In DCE 2.1, the keytab is also registered as an object in the CDS name space and can be manipulated remotely. Local access is still available.

First, we need the name of the *keytab* object. This can be obtained by the following command. On ev1, execute the `keytab catalog` command for two hosts.

```
dcecp> keytab cat
/.../itsc.austin.ibm.com/hosts/ev1/config/keytab/self
dcecp> keytab cat /.:/hosts/EV5
/.../itsc.austin.ibm.com/hosts/EV5/config/keytab/self
```

We can `show` the contents of the object and find out whether a disk file is already linked to this object. The keytab information (principal-password pairs) is not contained in the object itself but in a local file linked by the object. The keytab file and its format is still the same as in previous DCE releases, but now, it can be accessed through a dced object.

```
dcecp> keytab show /.:/hosts/ev1/config/keytab/self
{uuid 00096565-4301-1d77-9108-0000c09ce054}
{annotation  {Host Principal Keytab}}
{storage /krb5/v5srvtab}
{/.../itsc.austin.ibm.com/hosts/ev1/self des 1}
{/.../itsc.austin.ibm.com/hosts/ev1/cds-server des 1}
{/.../itsc.austin.ibm.com/hosts/ev1/cds-server des 2}
```

```
{/.../itsc.austin.ibm.com/hosts/ev1/dfs-server des 1}
{/.../itsc.austin.ibm.com/hosts/ev1/dfs-server des 2}
```

A keytab show command on the local host could look like:

```
dcecp> keytab show self
```

6.7.6.2 Step-by-step Instructions to Create a Keytab

The commands discussed in this section were tested with the simple setup shown in Figure 56 below. We created a small cell composed of two OS/2 Warp machines. *EV5* contains the master servers. In this scenario, we will create a keytab file on *EV6* and populate it with some users, all from *EV5*.

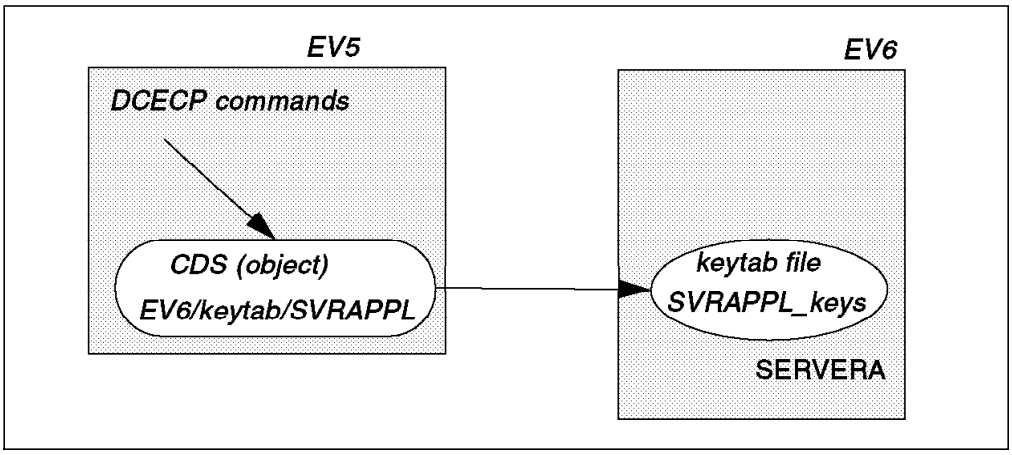


Figure 56. Remote Keytab Creation from EV5 to EV6

On *EV6*, no keytab file is yet registered. The first action would be the creation of the keytab on *EV6*. This can be done locally or remotely from another system, but some authorizations are required. We will try to do it remotely.

From the *EV5* system, we enter the following command. The user, *SERVERA,* does currently not exist in the registry:

```
DCECP>keytab create /.:/hosts/EV6/config/keytab/SVRAPPL
-attr {{storage /opt/dcelocal/key/SVRAPPL_keys } \
{data { SERVERA plain 3 itso}}}
```

The previous command did the following:

- Created an object /.:/hosts/EV6/config/keytab/SVRAPPL
- Created the keytab file /opt/dcelocal/key/SVRAPPL_keys
- Added in this keytab file one entry for account SERVERA

It seems impossible to create an empty keytab file; at least one entry has to be specified as in the previous example.

The keytab cat command verifies the just-created keytab:

```
keytab cat /.:/hosts/EV6
/.../itsc.warp5.ibm.com/hosts/EV6/config/keytab/self
/.../itsc.warp5.ibm.com/hosts/EV6/config/keytab/SVRAPPL
```

A keytab show of the object gives the following result:

```
dcecp> keytab show /.:/hosts/EV6/config/keytab/SVRAPPL
{uuid bb9ae380-67b7-11cf-a2dd-08005a49f2f8}
{annotation  {}}
{storage /opt/dcelocal/key/SVRAPPL_keys}
{/.../itsc.warp5.ibm.com/userb des 3}
```

For the principal, *SERVERA*, additional actions have to be taken:

- The account *SERVERA* has to be registered in the security database.
- The password has to be synchronized between keytab and registry.

SERVERA is added into the registry by a series of commands:

- group create creates a group, *SVRAPPL*, in the registry.
- org create creates an organization, *IBM*, in the registry.
- user create creates principal *SERVERA*, account *SERVERA* belonging to the above group and organization. The password to specify *must* be the same as during the keytab create action.

With the server account now present in the keytab and in the registry, and synchronized with respect to the password, we are able to switch to a *randomized* key pattern. This can be done in the following way.

```
DCECP> keytab add /.:/hosts/EV6/config/keytab/SVRAPPL -member SERVERA
-registry -random
```

This command has been executed remotely from EV5 to EV6.

Additional users can now be defined in the registry, and in the keytab, but to activate a randomized key for an account, a two-step procedure will always be required as outlined in the previous example. Additional users can be added to an existing keytab file and randomized with the following two command:

```
DCECP> keytab add /.:/hosts/EV6/config/keytab/SVRAPPL -member SERVERB
-key itso -version 1
DCECP> keytab add /.:/hosts/EV6/config/keytab/SVRAPPL -member SERVERB \
-registry -random
```

From a central point, *keytab* files can be maintained on remote target platforms. Additional members can be added; keys can be regenerated periodically. For these protected actions, permissions are required that are controlled by the ACL on the keytab object in CDS. Here is the result of the show command.

```
dcecp> acl show /.:/hosts/EV6/config/keytab/SVRAPPL
acl show /.:/hosts/EV5/config/keytab/SVRAPPL
{unauthenticated acdepr}
{any_other acdepr}
```

Long-running server applications usually spawn a thread that checks the password expiration date, goes to sleep, wakes up shortly before the password expires, and updates the password with another randomly generated password. The dced process does this for the password of the machine principal.

6.8 Running DCE Authenticated Batch Jobs

Due to the nature of DCE and its principles and standards, every kind of user, no matter whether this is a human user, a program, a system, or anything else, needs to be authenticated in order to make use of the functionality DCE provides. This authentication is done by providing an ID and a password, which the DCE Security Service compares against its registry database.

Let's verify this with a human user. He (or she) logs into the operating system and then into DCE by using the `dce_login` command, providing his (her) user ID and password. Of course, this double login could also happen in one step using AIX 4.1.3+ or other integrated single-login techniques. DCE then provides the user with the credentials that all processes subsequently started by this user inherit. Without such credentials, neither the user nor any launched non-DCE application could ever benefit from any DCE services.

This can cause some headache if a system administrator wants to automate certain tasks that need DCE authentication and have them started automatically, for example by cron. A simple solution would be to have a `dce_login` done within a shell script. Although technically simple and possible, this would require the password to be stored somewhere in a file. Moreover, if that password would be passed to that shell script as a parameter, anyone could see it with a `ps` command.

DCE server applications also need to provide a password (here called a *key*) in order to register successfully with DCE. They usually do it the same way as all DCE components by storing an encrypted version of their keys in a so-called *keytab* file. All DCE components, such as a DFS or CDS server, use the file /krb5/v5srvtab. The contents of this file can be viewed and managed with the `rgy_edit` command. Its subcommand, `ktlist`, lists all the names of the principals that have keys stored in this file (for security reasons, you cannot list the encrypted keys themselves). For more details on keytab entries and management, see 6.7.6, "Managing the Keytab" on page 236.

6.8.1 Running Batch Jobs Using start_batch

If any program is to be started with appropriate DCE credentials, but lacks the functionality of doing a DCE login on its own, like a simple shell script, someone else has to do the login. This would be a system administrator in most cases, starting the programs from his user environment.

This would not work if a program is to be launched off-line from cron or from inittab during system startup.

We provide a tool called `start_batch` on the diskette that comes with this book. It does basically the same as described above what DCE components and other DCE server processes do:

1. It accesses the /krb5/v5srvtab file to get the key for a given principal.
2. It does a DCE login with that principal name and the key.
3. Upon success, it starts a program with the DCE credentials.

The `start_batch` command needs two mandatory parameters:

```
start_batch <principal> <command-to-be-executed>
```

The given principal must be defined and have a valid key in the keytab file on the machine where it is supposed to run. This one-time definition must be done by the administrator. The command can be any command that could also have been entered via a command line, for example:

```
# start_batch backup_principal DFS_backup
```

An equivalent of a dce_login is performed with the principal name *backup_principal* and its key obtained from /krb5/v5srvtab. Then the command *DFS_backup* is executed. In this way, any program can be run in a DCE authenticated environment.

Caution: Although such a tool is desirable or even necessary in almost any DCE installation, it may lead to some questions about security. We strongly recommend the following:

1. Do not use the principal *cell_admin* for this purpose. This is not only a security issue. Whenever the password of *cell_admin* changes, one would also have to change the key(s) in the keytab file(s). Add another principal with the appropriate privileges for this purpose.
2. Only root should have access to this tool. It should have restricted file permissions: 700 (-rwx------).
3. You may even make it part of the Trusted Computing Base (TCB).

If you add a new principal and account for this tool by using SMIT, then you should specify that this account is going to be an RPC server. This way, SMIT will do all the entries in /krb5/v5srvtab.

`start_batch` is a simple implementation of such a tool. It does not alter the key in the keytab file automatically after expiration, and it only allows you to use the default keytab file, /krb5/v5srvtab.

Chapter 7. Miscellaneous Tools and Technologies

This chapter is a collection of additional technologies or tools for DCE administrators. The level of detail given here would have overloaded the previous how-to chapters. We cover the following topics in this chapter:

1. DCE for AIX release history
2. DFS file server replication
3. NFS/DFS authentication translator
4. DCE Web
5. Login integration
6. A mass user/group management tool

In the previous chapters, we showed how to use all these features without explaining the details.

7.1 DCE for AIX Release History

In this section, we will provide a brief overview of the most important new features and functions that came with DCE Version 1.3 and DCE Version 2.1. Some of them are IBM-exclusive.

7.1.1 AIX DCE 1.3 New Features Overview

The purpose of this chapter is to list and explain the most important new features of AIX DCE 1.3. Some of the enhancements are performance oriented, some provide extended functionality.

Two of the most important new features, DFS replicated file server and the NFS to DFS Authenticating Gateway, are covered in more detail in separate sections. See 7.2, “DFS Replication” on page 252 and 7.3, “NFS-to-DFS Authenticating Gateway” on page 257.

These are the new features that are explained in the rest of this overview section:

- Security server replication
- DFS fileset replication - see 7.2, “DFS Replication” on page 252
- NFS to DFS Authenticating Gateway - see 7.3, “NFS-to-DFS Authenticating Gateway” on page 257
- Split configuration
- Local RPCs
- Environment variable RPC_UNSUPPORTED_NETIFS
- Monitoring function in IBM NetView for AIX
- Exportable data encryption facility (Common Data Masking Facility, CDMF)
- Stub-size reduction
- Preferred file server for DFS clients

7.1.1.1 Security Server Replication

This feature is actually already part of OSF DCE 1.0.2, but was added later in DCE for AIX. It has been delivered for AIX DCE 1.2 now as PTF#U431018. See 3.7.2, “Replicating the Security Server” on page 67, for an example on how to configure it. See also the release notes that come with the PTF and with AIX DCE 1.3 for various considerations and detailed explanations of the Security Service in general.

The security registry database is copied as a whole to all defined replication servers, where it is read-only. This is sufficient for getting tickets for `dce_login` or DCE server access. The only time write access is needed is when a new principal, account, or group is added/changed/deleted.

Once you have installed one or more security replication servers, you can indicate any of them when you issue `mkdce -s <sec_server_name> <DCE_component>` to install new components. If you specify a replica server, a connection is made to this server, and its binding information is put into the pe_site file in the first position:

```
# cat /etc/dce/security/pe_site
/.../itsc.austin.ibm.com 007abb9c-8a15-1df3-b3c0-10005aa8c755@ncacn_ip_tcp:9.3.1.127[]
/.../itsc.austin.ibm.com 007abb9c-8a15-1df3-b3c0-10005aa8c755@ncadg_ip_udp:9.3.1.127[]
```

Since configuring a new component requires write access to the registry, a connection is automatically made to the master security server. However, you usually specify the master security server with the `mkdce` command.

If there are multiple security servers in a cell or after a new security replication server has been added, the pe_site file can be updated to contain a list of all available security servers:

```
# chpesite
```

This is an IBM-provided tool. It must be run on all systems in order to update their pe_site file.

7.1.1.2 Split Configuration

This feature, which also includes a split unconfiguration, separates configuration of DCE client machines into a central DCE administrator part and a local system administrator part.

Up to now, we only had the full client configuration method. The administrator who configured the client had to be logged in as the local root user and had to provide the DCE cell_admin password to the configuration routine. This was because entries had to be made for the new client in the security registry and the CDS namespace, which required write access.

These entries are still necessary, but can now be preconfigured by cell_admin from any machine already configured in the cell. This is what the admin part of the split configuration is all about.

The owner of a workstation who wants to be part of a DCE cell can now request configuration from a central DCE administrator. Once the workstation is preconfigured, the user can configure their machine as a DCE client without having to know cell_admin’s password. This is very convenient for very large DCE cells.

The `mkdce` command has some new flags to allow for split client configuration:

```
mkdce -o configtype -h dce_hostname -i ip_identity
```

configtype	Specifies what type of DCE client configuration is used:	
	full	This is the default. It is the old-style configuration where everything is done on the client to be installed.
	admin	This is the admin part of the split configuration where the cell_admin password has to be known.
	local	This is the local part of the split configuration to be executed on the client to be installed.
dce_hostname	This is a specifically selected name for the new client under which they will be known in DCE. This affects the machine principal name in the security registry and the hosts entry in CDS. It can be the same name as the TCP/IP name, which is somewhat long, though, when it contains the domain name.	
ip_identity	This is either the IP address or the TCP/IP hostname of the machine for which a DCE client is being preconfigured.	

7.1.1.3 Local RPCs

If DCE client and server applications are running on the same system, communication now goes through a local UNIX socket rather than through a network interface and the network layers. This prevents a server connection from being established over the network either to another or to their own machine when the service is available on their own machine. This brings performance improvements for client programs that run on a server machine.

If a client uses CDS to obtain binding handles, it will always get the local sockets first, even with calls such as rpc_ns_import_binding_next(), which returns bindings in a random order.

This binding information is not stored in CDS because all clients would get these local sockets. It is rather the client's RPC runtime which realizes that some of the handles it receives contain an IP address that corresponds to the client's own system. It creates the local binding handles and returns them to the caller in the first place, before those from CDS.

The following is an example of a local RPC socket compared to regular TCP and UPD sockets. The example is the pe_site file of a system that runs a security server. It could also be the response to consecutive rpc_ns_import_binding_next() calls, which return partly bound handles without endpoints:

```
# cat /etc/dce/security/pe_site
/.../jacques.itsc.austin.ibm.com 007abb9c-8a15-1df3-b3c0-
10005aa8c755@ncacn_unix_stream:[]
/.../jacques.itsc.austin.ibm.com 007abb9c-8a15-1df3-b3c0-
10005aa8c755@ncacn_ip_tcp:9.3.1.127[]
/.../jacques.itsc.austin.ibm.com 007abb9c-8a15-1df3-b3c0-
10005aa8c755@ncadg_ip_udp:9.3.1.127[]
```

The endpoint for a ncacn_unix_stream binding handle is represented as full path names to a UNIX socket file. A unique socket file is used for each association established between a client and server process. By default, these socket files are opened in the directory /opt/dcelocal/var/rpc/socket. Also by default, the name for each socket file is an object UUID, which ensures the uniqueness of

each file name. This means there is no chance that a socket file will ever be used over again on two invocations of a DCE application.

Here is an example of a ncacn_unix_stream string binding:

```
ncacn_unix_stream:[/var/dce/rpc/socket/0063980e-357b-1e07-878b-10005a4f3bce]
```

When a well-written DCE server application exits under normal conditions, it will unregister its endpoints from the RPC endpoint map, among other things, before it exits. The ncacn_unix_stream endpoints are user-space files. Over time, these socket files will accumulate. They are zero-length files, but each one occupies an i-node entry in the file system that it was created in. It is necessary to have some means of cleaning up these stale socket files. This is done by the RPC endpoint map.

By building their own string bindings, applications can define other file names for the socket files. However, this incurs additional overhead, and the stale socket files are not removed automatically. The `rpc.clean` command has to be called together with the directory name that contains user-created socket files.

7.1.1.4 Environment Variable RPC_UNSUPPORTED_NETIFS

This variable excludes a list of network interfaces from being used in DCE binding handles. The following examples excludes xt0 and sl0:

```
export RPC_UNSUPPORTED_NETIFS=xt0:sl0
```

This environment variable should be set in the /etc/environment file before DCE is configured. It prevents services from exporting their interfaces into CDS. This is a very important instrument in the performance and availability considerations for the DCE cell layout. This is discussed in more detail in Chapter 5, “Implementing Various LAN/WAN Scenarios” on page 105, and in 3.2, “Preparing for DCE Configuration on AIX” on page 38.

It is also required and used in HACMP/6000 configurations with DCE because HACMP/6000 has redundant interfaces, and some of them must not be used by regular applications. They need to be idle in the HACMP/6000 configuration such that they are ready for takeovers.

7.1.1.5 DCE Manager for AIX

IBM NetView for AIX is the platform for network management based on the simple network management protocol (SNMP). It can be used for monitoring any kind of TCP/IP devices and for managing devices that are able to support the SNMP protocol. An SNMP agent is the interface to and from such SNMP-capable devices. SNMP agents have a database of system control variables that are standardized. It is called the Management Information Base (MIB). An SNMP management node can manage an SNMP agent node by changing MIB values. SNMP agents can be configured to send traps or alerts to the manager node when certain events happen or when thresholds are exceeded.

IBM NetView for AIX provides a GUI to display network topologies so that users will be alerted and can respond to abnormal conditions in their network. APIs on both sides, the manager and the agent side, provide an opportunity to integrate new applications into this management interface.

Applications integrated on the management side usually do not communicate via SNMP protocol to the other nodes. They just run on the central node and have their own method or protocol to get information from their clients. They may

even be able to manage their clients through the use of this proprietary method or protocol.

Applications implemented on the SNMP agent side are called subagents. They basically function like the agent itself. They support MIB values and generate traps. They implement an extended MIB that allows the manager node to actually manage these subagents via a regular SNMP methods.

What does IBM NetView for AIX now do for DCE?

The DCE monitoring function (the DCE Manager for AIX) is implemented on the management side and does not support SNMP. It uses traditional DCE tools to look up, for example, CDS entries or RPC endpoint maps or to test availability of services. Even though it cannot manage any DCE services or databases, it gives significant value to customers, such as:

- DCE topology view providing location, function, and role of DCE services in the network. This information is dynamic and responds to changes in the DCE or network configuration.
- Monitoring the basic state of the DCE services in the network and indicating fault conditions to the network administrator.

7.1.1.6 Exportable Data Encryption Facility CDMF

The Common Data Masking Facility (CDMF) is an exportable user data encryption facility that can replace the DES algorithm. DES may be exported within a product that does not export any interface to DES routines. So, DCE internally continues to use DES, but as an option, users can install CDMF if they need access to encryption routines.

CDMF is an IBM-patented encryption algorithm that utilizes the underlying DES support in DCE without exposing it directly to the application. It uses the DES algorithm but exposes a weaker 40-bit key to the application in contrast to the full 52-bit DES key.

7.1.1.7 Stub-Size Reduction

This performance enhancement produces smaller RPC stubs, thus improving throughput and reducing memory allocation needs. Since all DCE services and applications are based on RPC, this should significantly improve overall performance in the cell.

7.1.1.8 Preferred File Server for DFS Clients

The Cache Manager maintains ranks for file server machines. A file server machine's rank determines the Cache Manager's preference for electing to access replicas that reside on the file server machine over replicas that reside on other file server machines. You can specify preferences for file server machines to bias the Cache Manager's selection process.

This is an important feature for cells having WAN connections. It may help reduce network traffic that could have occurred with random server selection.

7.1.2 IBM DCE 2.1 New Features Overview

The most important new items of the this release are highlighted here. It is well known that the DCE 2.1 release implemented on AIX 4.1 and OS/2 Warp corresponds to OSF DCE 1.1. It is important to mention that the OSF DCE 1.1 implementation also is available on MVS 5.2.2, currently with the exception of the Directory Services.

Most of the new features came as part of the new OSF DCE 1.1 release, such as:

- Integration Services
- DCE Control Program
- DCE Host Daemon
- Cell Aliasing
- Security Delegation
- Auditing
- ACL Manager Library
- Password Management API
- Internationalized Interfaces
- Character Code Set Interoperability
- IDL Compiler

In addition to these OSF features, IBM provided some enhancements to their IBM DCE 2.1 for AIX and OS/2 Warp, such as:

- Event Management Service
- Automatic local RPCs
- Configurable preferences for FLDB server access
- Support for a CD-ROM file system in DFS

7.1.2.1 Integration Services

Within this category, we mention all the items that are part of this new release but which could be used without the Remote Procedure Call and which could be very useful for other client/server approaches like APPC/CPI-C and MQSeries products.

Encoding Services: The IDL Encoding Services provide client and server applications with a method for encoding data types of input parameters into a byte stream and decoding data types in output parameters from a byte stream. Encoding and decoding are analogous to marshalling and unmarshalling, except that the data is stored in a local buffer and is not transmitted over the DCE network.

Encoding flattens complex data (structures) into a byte stream in a local buffer or restores the complex data from the flattened data in the buffer. Encoded data can be written to a file or forwarded by a *message-passing system*. This is a way to exchange complex data types (even binary data) between different platforms, regardless of their data type size and endianness. The only DCE service needed is an RPC runtime on both the encoding and the decoding end.

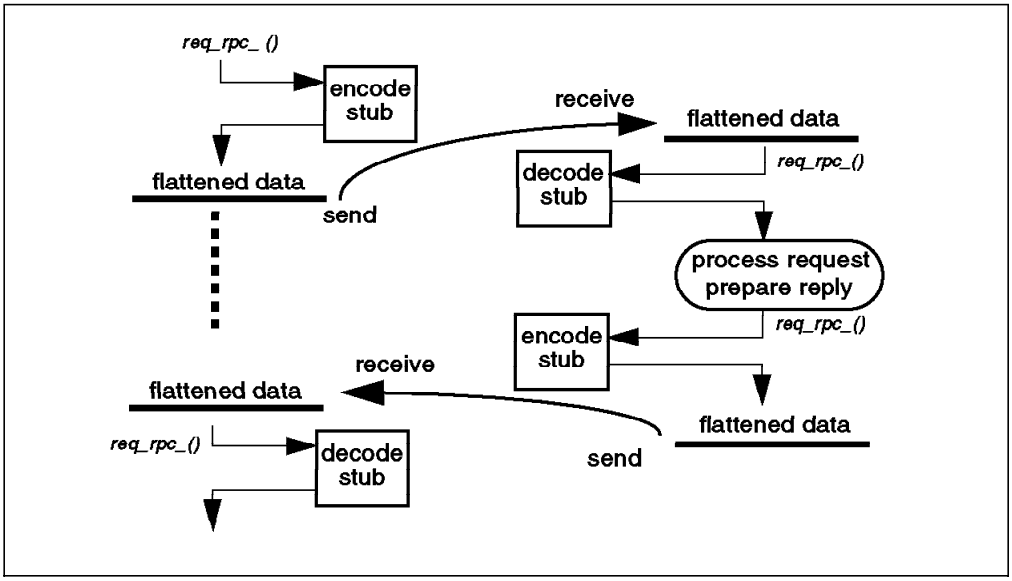


Figure 57. Encoding Services

The *encode* and *decode* part both use the same API offered by IDL-generated stubs. So in the above case, two stubs have to be generated: one for the request and one for the response. For additional details, look into the *IBM DCE 2.1 for AIX Application Development Guide - Core Components* or equivalent OS/2 Warp and MVS publications.

GSS-API - Generic Security Services API: The Generic Security Service (GSS) provides an alternative way of securing distributed applications that handle network communications by themselves. With the GSS-API, applications can establish secure connections and act like DCE RPC servers.

The GSS-API is a standard API for interfacing with security services, as defined by the IETF RFCs 1508 and 1509. It allows flexible use of the DCE security by programs, even if they don't use DCE RPC to communicate (see Figure 58). Because the GSS-API is product neutral, you can define your security policy and implement it using the generic API.

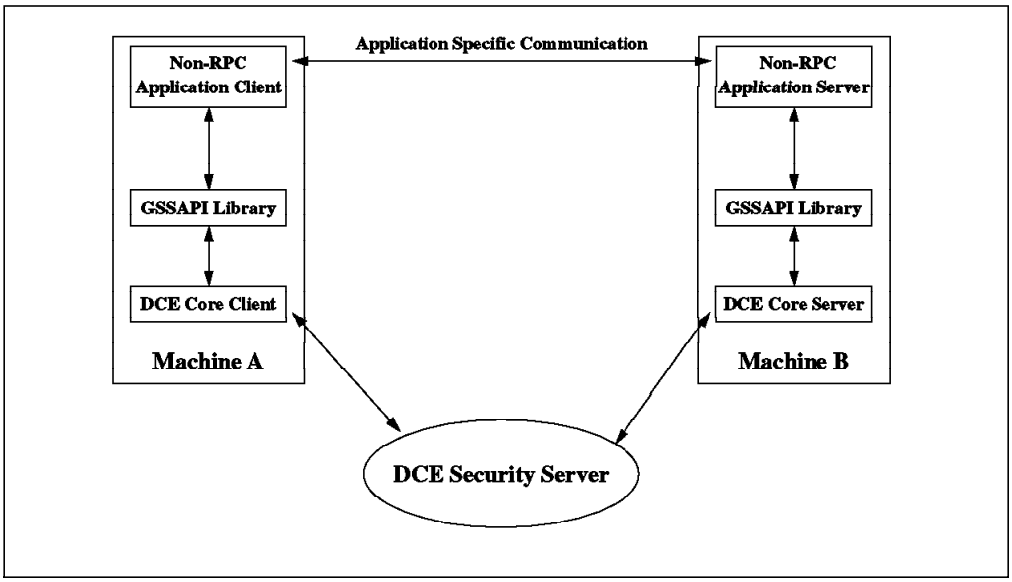


Figure 58. Generic Security Service API (GSS-API)

The GSS available with DCE includes the standard GSS-API routines (Internet RFC 1509) as well as OSF DCE extensions to the GSS-API routines. These extensions are routines that enable an application to use the DCE Security Service. However, if applications make use of the DCE extensions, they will not be portable to other security mechanisms because they will not understand, for instance, EPACs.

A GSS-API caller (usually the application client) accepts tokens provided to it by its local GSS-API implementation and transfers the tokens to a peer (usually the application server) on a remote system. That peer then passes the received tokens to its local GSS-API implementation for processing.

Clients as well as servers first have to authenticate themselves with the security server (network login).

Extended Registry Attributes: The registry stores specific information about principals, groups, organizations, and accounts. The kind of information stored in the registry database is defined in a registry schema, which is essentially a catalog of the kinds of data stored in the database. There is a schema entry definition for each type of attribute that can be associated with a registry object. Using the Extended Registry Attribute (ERA), you can add your own schema entries. These attributes are called extended attributes. Once those extended attributes have been defined, they can be attached to a registry object (principal, group...) with dcecp commands.

A useful implementation of this feature can be the *Identity Mapping* between the DCE principal and the MVS Resource Access Control Facility (RACF) user. The RACF user is the user ID against which the access lists stored in the RACF database are checked.

If a principal has extended attributes, these attributes are carried with the Extended Privilege Attribute Certificate (EPAC) obtained when the principal has been authenticated. More information about this subject can be found in the redbook *Understanding OSF DCE 1.1 for AIX and OS/2*.

7.1.2.2 DCE Control Program

The DCE Control Program (dcecp) provides consistent, portable, and secure access to nearly all DCE administration functions from any point in a DCE cell.

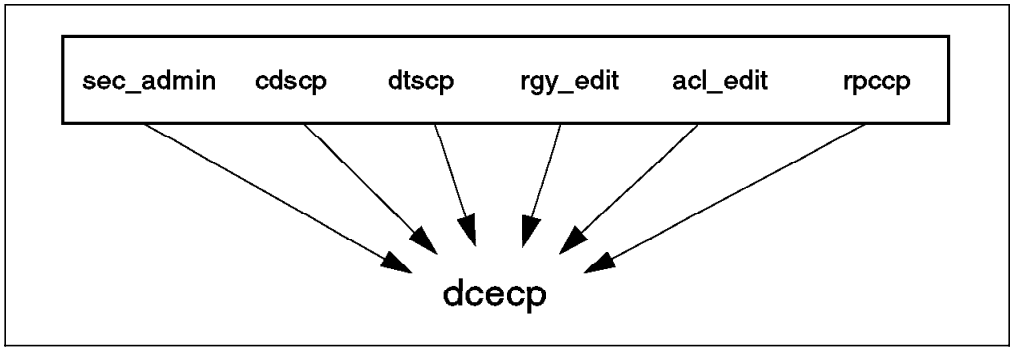


Figure 59. DCE Control Program

As Figure 59 above shows, dcecp incorporates into a single command interface most of the operations previously performed by using various components' control programs. In cooperation with the DCE host daemon (dced), it allows for remote administration of the DCE machines.

It also includes the Tool Command Language (Tcl), a powerful and portable scripting language. Tcl is platform-independent and should run on each platform where DCE Version 1.1 is installed. Portable extensions to dcecp can be written in Tcl to simplify administrative tasks.

7.1.2.3 DCE Host Daemon

The DCE host daemon (*dced*) enables complete remote administration of DCE services and other applications. It incorporates the functions of the previous RPC daemon and the security client daemon.

This function not only allows the administration of the DCE-related servers, such as CDS and security, but also provides the opportunity to manage some remote objects that are local to the individual platforms, such as servers, hostdata, and keytabs.

Another interesting capability offered by the new *dced* is the possibility to get an application server started by an incoming client. This assumes that the server has been registered in the `srvrconf.db` and eventually that its interface has been manually exported into the CDS directory.

7.1.2.4 Cell Aliasing

This feature permits a cell to have multiple names and eventually allows the primary name of the cell to be changed.

The first step consists in creating the *cell_aliases* object in dcecp, which is an alias name of the cellname. Afterwards, this alias name can be set to the primary name. The *cellalias* operation is very powerful because it also tries to change data in the *hostdata* container of all machines in the cell.

However, although it may seem that this dcecp object is there and can be used, cell aliasing is not yet working. None of the DCE vendors received working code from OSF to support cell aliasing or hierarchical cells.

7.1.2.5 Security Delegation

This feature allows intermediary servers to act on behalf of an initiating client while preserving the client's and server's identities and access control attributes across chained RPC operations.

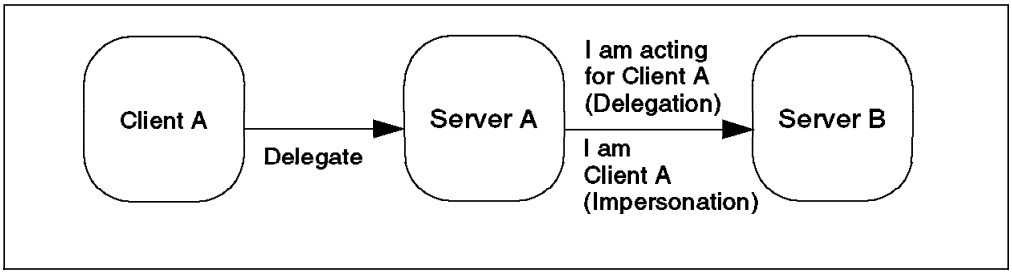


Figure 60. Delegation

The DCE delegation model requires the extension of two components:

1. Extended privilege attributes certificates (EPACs)
2. ACL model

The EPACs need to contain all involved identities. The ACL contains more entry types, such as *user_obj_delegate*, *group_obj_delegate*, *user_delegate,* and so on.

The object being protected by ACLs has to set up the according permissions if it wants to allow for delegated access.

An application server can also impersonate the client, meaning it can temporarily adopt the identity of the client.

7.1.2.6 Auditing

This allows administrators to track security-related events within a DCE's trusted computing base. An API is included and permits the development of servers that record audit events. The data structures and API functions involved in this auditing capability are derived from those of POSIX 1003.6. The features of the DCE Audit Service are the following.

1. An audit daemon (auditd) that may run on all hosts in the cell.
2. An API that can be used to record audit events. This can also be used to craft tools that analyze the audit records.
3. An administrative interface to the audit daemon.
4. An event classification mechanism that allows the logical grouping of a set of events.
5. Audit records can be directed to the log or to the console.

7.1.2.7 ACL Manager Library

The ACL Library provides simple and practical access to the ACL Manager Interface and the ACL Network Interface for the convenience of programmers who are writing ACL managers called on by the DCE servers. The library provides DCE developers with a set of convenience functions so that they can implement ACL managers more easily.

The ACL API consists of three parts.

1. The *sec_acl...()* API is used by clients that need browse and edit ACLs. The `acl_edit` or `dcecp acl` commands are built using this interface.
2. The network interface *rdacl..()*. ACL managers must implement the code to the predefined calls of this interface. A working ACL manager must be able to respond to all these calls. Callers can be clients that use the *sec_acl* interface.
3. The ACL manager library *dce_acl...()* enables the server code to perform the DCE-conformant authorization lookups at run-time.

7.1.2.8 Password Management API

User passwords are the weakest link in the chain of the DCE security. Easily memorized passwords are also easy for attackers to crack. The Password Management facility is intended to reduce this risk by providing the tools necessary to develop customized password-management servers and to call them from the client password change programs. It enables enforcement of:

1. Stricter constraints on user passwords.
2. Automatic generation of user passwords.

7.1.2.9 Internationalized Interfaces

This allows you to use a message catalog for all user-visible messages. It is now possible to localize DCE programs by supplying DCE messages in other languages.

7.1.2.10 Character Code-Set Interoperability

This is another aspect of the internationalization. It allows the development of RPC applications that automatically convert character data from one code set to another, and it enables the preservation of the character data integrity.

7.1.2.11 IDL Compiler

The compiler generates smaller and cleaner RPC stub code. It also supports new IDL constructs such as unique pointers, user exceptions, and node deletions.

As already mentioned, the Encoding Service (part of the marshalling) can be used as a stand-alone function. This functionality is requested by an appropriate IDL in combination with an ACF definition, specifying *encode* and *decode*.

7.1.2.12 DFS Enhancements

DFS now supports exporting an AIX CD-ROM file system and file system sizes greater than 2 GB.

The AIX authentication service for PC-NFS clients is now integrated with the NFS-to-DFS gateway.

7.1.2.13 Event Management Service

The DCE Event Management Service (EMS) supports asynchronous event management for use by system management applications. EMS uses the concepts of event suppliers and event consumers and sets up an event channel between them to support asynchronous communication. In the context of DCE, event suppliers are any DCE core service or DCE-based application (client or server), and event consumers can be any application with an interest in receiving asynchronous events from one or more DCE processes. The transmission of events between suppliers and consumers is uncoupled by routing events via EMS which is the implementation of an event channel. EMS also provides a filtering mechanism to allow administrators and consumers control over which events EMS will send. EMS provides integration for DCE clients and servers using the DCE Serviceability (SVC) interface. DCE applications can use the APIs offered in SVC to become event suppliers.

A Simple Network Management Protocol (SNMP) subagent for DCE gets its information, besides other sources, from EMS by registering as an event consumer. The SNMP traps and events are then sent by the SNMP agent on the local machine to an SNMP manager, such as the IBM SystemView.

These features are available on OS/2 Warp but not in this release of AIX DCE.

7.1.2.14 Automatic Local RPCs

The UNIX protocol sequences introduced with AIX DCE 1.3 and described in7.1.1.3, “Local RPCs” on page 243, are no longer needed. The RPC runtime handles client/server connections on the same machine now ″under the covers″. It uses UNIX domain sockets and still creates socket files for local RPC connections, but the DCE applications always see regular binding handles as if a network were involved.

7.2 DFS Replication

This subject is already fully covered in theory by the *The Distributed File System (DFS) for AIX/6000* redbook. However, since this feature was not available when the DFS redbook was published, we tested it and give a general overview and guideline. The reader should be familiar with DFS in general before reading this section.

For a step-by-step instruction on how to set up DFS replication see 4.4, “Replicating Filesets on AIX” on page 89.

7.2.1 Overview

DFS replication is the ability to have one or more read-only copies (replicas) of a read/write DCE LFS (Local File System) fileset. The different copies are hosted on multiple file servers. Therefore, if one server machine goes down, you can still access the information from other available servers.

Replication is supported only for DCE LFS filesets, not for non-LFS filesets. An example of a non-LFS fileset is the AIX Journaled File System (JFS). The AIX JFS can be exported into the DFS file space, but it lacks the ability to be replicated.

A read-only fileset is an exact copy, or replica, of all data contained in a read/write source fileset. The read-only fileset receives a *.readonly* extension to the fileset name. It is updated when changes are made to the read/write fileset. The frequency of updates depends on the type of replication. Two types of replication are available with DCE LFS filesets:

- Release replication
- Scheduled replication

With release replication, you manually propagate the update from the read/write fileset server to the read-only fileset server(s) at the frequency you want. This type of replication is useful if the fileset seldom changes or if you need to closely monitor the replication process.

With scheduled replication, you specify replication parameters that dictate how often DFS is to automatically update replicated filesets with new versions of source read/write filesets. This type of replication is useful if you prefer to automate the process and do not need to track exactly when releases are made.

Both types of replication produce the same result: Changes to the read/write source filesets are copied to different server machines. It is the duty of the system administrator to choose which type of replication to use with each fileset. The next sections give a summary of the DFS replication concepts.

7.2.2 Why Fileset Replication?

One of the advantages of the DCE DFS over another distributed file system is its ability to replicate a fileset on multiple machines. Actually, when you replicate your fileset, you can benefit from higher availability and load balancing.

- Availability

  Replication minimizes the effects of machine outages. If one machine housing a DCE LFS fileset is unavailable, replicated versions of the filesets are still available from other machines. To achieve this goal, it is not

sufficient to have just that one fileset replicated; all filesets higher up in the access path must also be replicated.

- Load balancing

  Requests for files of popular or frequently used DCE filesets are then spread across different machines, preventing any one machine from becoming overburdened with data requests.

These advantages are of course only valid for files which are accessed mostly for reading (see below).

7.2.3 Which Files to Replicate?

Even though we may replicate any fileset, care should be used in deciding which filesets would benefit most from replication. If the type of access is read-only, replication works perfectly, whereas filesets with frequent write accesses or updates should not be replicated. They would cause either a lot of network traffic, because they would have to be updated frequently, or even worse, the DFS clients would access outdated files. If the filesets are accessed read-mostly, it is your decision how often writes occur and how important it is for clients to always read the most current data. If that is not critical, you may decide to replicate for availability and performance (load balancing).

The DFS namespace should be planned accordingly; so you should put read/write files into read/write filesets and read-only files into read-only filesets. The fileset candidates for replication should also be put at a high level in the cell's file tree (for example root.dfs and its direct subdirectories). The reason for this is the fact that the decision algorithm of the *Cache Manager* to use read/write or read-only filesets depends on the availability of the replica on all intermediate steps in the search path. This point is further explained in 7.2.5, "Mount Points" on page 255.

7.2.4 Prerequisites for Replication

To allow for replication, all machines participating in the fileset exports must have been configured with the appropriate machine roles; this also includes the *replication server*.

It is also required to have a replication of the *root.dfs* fileset because it is the top-level fileset in any DFS full-path name, and to enable access to any read-only fileset, all higher level filesets traversed must also be read-only. Although it is sufficient to have a replication in the same aggregate as the read/write replica, from an availability point of view, additional replicas should also be created on other machines.

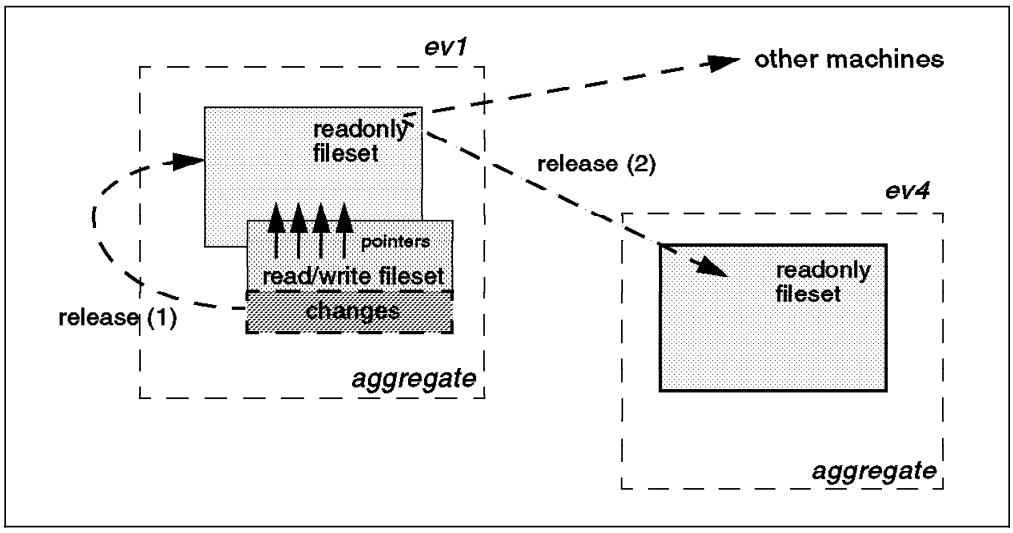


Figure 61. Replicating from ev1 to ev4

When a fileset is selected for replication on another platform, the following steps have to be executed.

- On the target platform, we have to prepare the *aggregate* and export it.
- We select the replication method via the `fts setrepinfo` command.
- With the `fts addsite` command, we indicate which sites we want to hold a read-only copy of the read/write fileset. This has to include a mandatory add `site` for the platform that houses the read/write fileset (ev1).
- Finally, depending on the selected replication method:
  - We issue the `fts release` command to activate a replication for the *release* replication.
  - We issue an `fts update` command to request an immediate update of the replicas in case we defined *scheduled* replication.

When the read-only fileset is created in the same aggregate as the read/write fileset (same server and aggregate), DFS attempts to save disk space by having the filesets share data that is the same across the different types of filesets. As shown in Figure 61, this is accomplished in the following way.

- When the read-only fileset is created, the new fileset is filled with an array of pointers to the data housed by the read/write source.
- Then the identities of the read/write and the read-only are exchanged so that the current read-only becomes the read/write source and vice versa. As long as the read-only remains identical to the read/write fileset, this one stays small. However, as changes are made to the data in the read/write fileset, the amount of storage for this set will increase.

When the `fts release` command is issued, the read-only copy on the same file server machine as the source is updated. The *repserver* on each machine that houses a replica of the read-only fileset then updates its replica to match the read-only replica on the source platform. The read-only will not change until there is a new `fts release`.

7.2.5 Mount Points

In order to become accessible by DFS clients, filesets need to be mounted. Mount points have to be created in the DFS filespace. Figure 62 on page 256 shows several such mount points. Mount points show up as directory names. Directories and files within a fileset can be accessed by specifying their full path name. A full path name contains one or more directory names that are fileset mount points.

There are two types of mount points that play an important role in the decision whether the read/write or the read-only fileset is going to be accessed:

- Regular mount point

  This is the usual type, to which any type of fileset can be mounted. If the read/write fileset name is mounted there, the Cache Manager will decide which fileset to access based upon criteria explained below.

- Read/write mount point

  Only read/write filesets can be mounted and accessed via this type of mount point.

The Cache Manager running in each DFS client system interprets the path name. When it encounters a fileset mount point, it looks up information about the fileset in the FLDB. Once it traverses a read/write type mount point, it only accesses read/write filesets, even if the underlying mount points are regular mount points associated with replicated filesets. As long as the Cache Manager traverses regular mount points, it accesses read-only filesets if they exist; if a read-only fileset does not exist, it accesses the read/write fileset. Once it encounters a read/write fileset that is not replicated, any underlying mount points will also access the read/write fileset even if a read-only fileset exists.

If the Cache Manager does not find the fileset type it looks for, it returns an error. In other words, for example, it never accesses the read/write fileset as a fallback variant when none of the replicas are available.

Before starting to replicate a fileset, we previously have to replicate the root.dfs fileset. If we use release replication, the first replica must be on the same machine that physically houses this root.dfs fileset. In order to use replication for any other fileset, we must create read-only versions (replicas) of all filesets mounted above it at higher levels in the file system. This means that we must create read-only copies of the filesets that contain their parent directories.

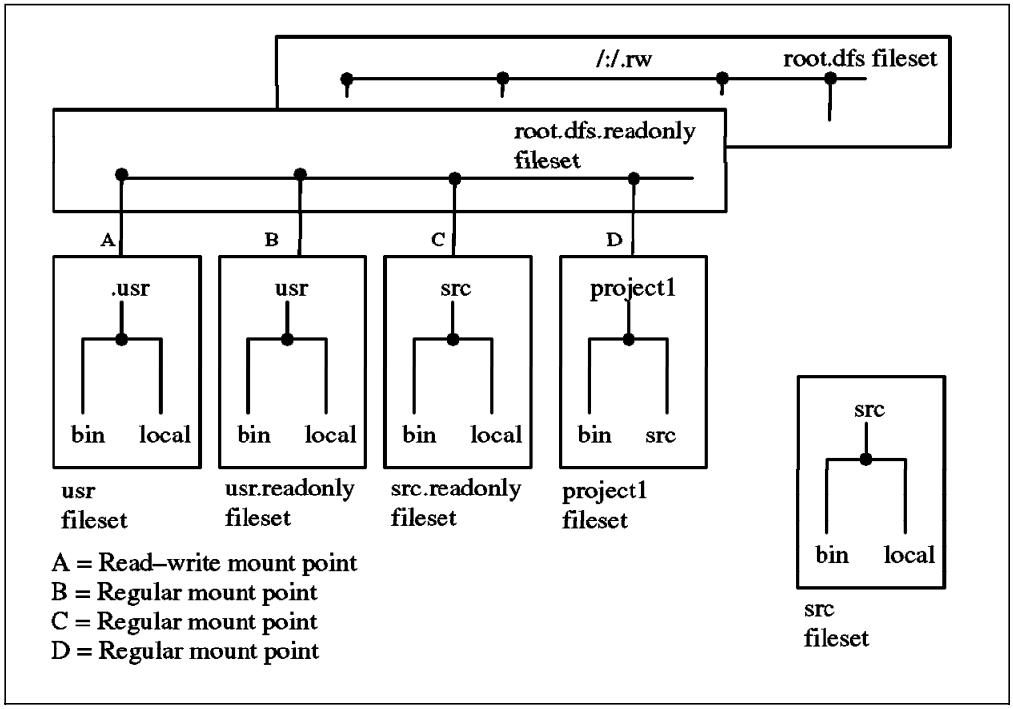


Figure 62. DFS Hierarchy File System

We can see in Figure 62 that the normal root directory mount point for the root.dfs read/write fileset is `/:/.rw` when the replication is active. /: becomes the access path to the root.dfs.readonly fileset. The two directories, /: and /.rw, are identical; they show the same contents. The difference is, however, that when you specify /:/.rw, you deliberately choose to access the read/write version of /: and *all underlying* directories and files, even though you might have other regular mount points and read-only filesets below the /: directory. An administrator may want to unmount the /:/.rw for daily use so that only the read-only fileset is accessible, and no changes can be made in the top directory. Unmounting a fileset actually means deleting its mount point.

Note in Figure 62:

- /:/project1 is mounted via a regular mount point and is not replicated. This read/write fileset is always accessible in read/write mode via /: or /:/.rw.
- /:/usr is a regular mount point for the usr fileset. Since usr is replicated, only the usr.readonly fileset is accessed via the /:/usr path.
- /:/.usr is a read/write mount point for the usr fileset. The read/write fileset can now be accessed by specifying the path name, /:/.usr, or via /:/.rw/usr or /:/.rw/.usr. It is especially useful when /:/.rw is deleted for daily use.
- /:/src is a regular mount point for the src fileset. It accesses the src.readonly fileset only because it is a replicated fileset, and the administrator decided to not create a read/write mount point. If changes need to be applied in the src fileset, a read/write mount point needs to be temporarily created, or access has to go via /:/.rw.

7.2.6 DFS Clients

DFS clients can be located on AIX (UNIX) platforms and OS/2 Warp platforms. These machines need to be configured as DCE and DFS clients.

DFS clients are not only caching fileset data but also fileset information, which means path names to fileset associations. Assume having had a read/write fileset mounted at a regular mount point for a while and DFS clients accessing it. Then you decide to replicate it with the intention to force further access to the read-only fileset. Since DFS clients are caching, they will continue to access the read/write fileset after you have created the replica as long as they work in that directory and the cache information is still valid. The cache information expires after one hour if the directory is not the working directory.

To make a replica available on a DFS client, the following steps need to be performed:

1. Force an update of read-only fileset containing the parent directory with `fts release` or `fts update`. If the regular mount for a new replica had existed before, you need not do that; the read-only fileset containing the mount point should be up-to-date.
2. Change the working directory to a directory outside of that fileset if it was accessible before the replication.
3. Refresh the cache's fileset information with `cm checkfilesets`.

The problem that DFS clients are able to access a read/write fileset via a regular mount point even though it is replicated can be avoided if you create the replica before you define the mount point. For details about configuration steps, see 4.4, “Replicating Filesets on AIX” on page 89.

7.3 NFS-to-DFS Authenticating Gateway

This section provides the description of the NFS to DFS Authenticating Gateway, sometimes also called the NFS/DFS Translator. This new functionality provides DFS the ability to interoperate with NFS (Network File System). We do not explain NFS; users are supposed to be familiar with it.

For a step-by-step configuration example of authenticated DFS access, see 6.6.4, “Configuring DFS Access from NFS Clients” on page 222.

7.3.1 Introduction

We know that many customers are currently using NFS as a technology to distribute file systems across the network. Many of their NFS client systems are OS/2 clients or even more often DOS/Windows workstations. For these platforms, DFS is not available yet. These customers need a transition period within which they can access the DFS filespace from their non-DFS or even non-DCE workstations. This has actually been possible since DFS was released. DFS clients can export their directories to NFS, but since NFS users are not authenticated, they obtained very limited access rights; they are considered *unauthenticated* users.

The functionality known as NFS/DFS Authenticating Translator or NFS/DFS Translator effectively provides NFS client users access to the DFS filespace. The NFS/DFS Translator provides a mechanism for establishing a bridge between the diverse authentication information by allowing a mapping to be established

between an NFS client and an authenticated DCE principal. See Figure 63 on page 258.

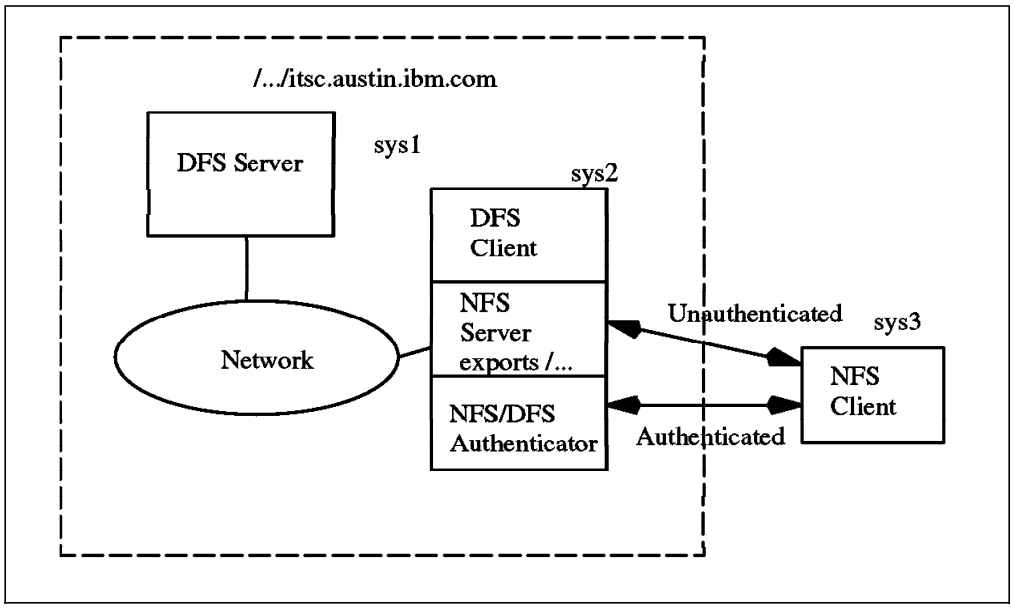


Figure 63. DFS/NFS Translator Architecture

Two scenarios are possible for *sys3*: it can be part of the cell, but DFS is not available on it, or it is not part of the DCE cell, because not even DCE is available on it.

7.3.2 Scope of Service

The primary function of the NFS/DFS Translator is to provide authenticated DFS access from NFS clients. The NFS client views the same DFS namespace, with the same file system hierarchy, as the DFS client, and we can, for example, export parts of the file systems to NFS. This allows NFS client users who do not have DFS ports for their hardware platform to participate in DFS file-sharing.

The NFS/DFS Translator does not provide complete DCE services from the NFS client side. DCE services, such as directory services, security services, and RPC services, are not available. Tools to modify ACLs or DFS administrative commands are also not available to NFS clients.

The real goal of the NFS/DFS Translator is to provide authenticated access to the DFS filespace from NFS client machines that do not need any extra software.

7.3.3 Concept

The standard DFS technology provides NFS access to the DFS filespace by allowing an NFS server to export the DFS client's view of the global filespace to an NFS client machine. This is achieved by exporting the root directory, /..., or any underlying directory, such as /.../<mycellname>/fs/mydirectory, to NFS. However, this functionality is limited to allowing only unauthenticated access to the DFS global filespace because NFS is unaware of DCE Kerberos-based authentication. As a result, anyone who is not identified as a DCE principal and makes an NFS request to the DFS filespace is treated as an anonymous user. In order to provide authenticated access to the DFS filespace from NFS, an additional agent is necessary to map the NFS-provided authentication information into DCE authentication information suitable for issuing an

authenticated DFS request. In other words, an NFS user needs to be mapped to a DCE principal.

The role of the translator is to map an incoming NFS client request credential into a credential representing a DCE principal. The NFS server then makes the file system request through the DFS client's virtual file system with the mapped credential so that the request looks to the DFS client as if it were made by a DCE authenticated process.

7.3.3.1 Functionality and Implementation

NFS and DFS both provide service from inside the kernel. Since the NFS/DFS Translator needs to reference NFS and DFS services that are not exported to user processes, the translator must also reside inside the kernel. On AIX, this is achieved by adding a kernel extension.

DFS, like NFS, is a layer underneath the VFS (Virtual File System). VFS is an abstraction of a physical file system implementation. It provides a consistent interface to multiple file systems, both local and remote. A virtual node (v-node) represents access to an object within a virtual file system. Associated with each v-node is a vector of procedures (read, write, create, remove), the vnodeops. The NFS server performs the service indicated in a received request by calling the vnodeop operations. Since the DFS client is integrated into the VFS model, it also provides a full set of vnodeops. These vnodeops are used by the NFS server to export the file system as seen from the DFS client.

7.3.3.2 NFS/DFS Translator Administration Model

DFS uses DCE Kerberos-based authentication. The role of the translator is to map an incoming NFS client request credential into a credential representing a DCE principal. NFS/DFS Translator administration requires commands or services to administer authenticated mappings in order to:

- Authenticate a DCE principal to be associated with an NFS host IP address/ UID pair
- Register NFS/DFS translation mappings on the NFS/DFS Translator site, also known as a translation point
- Query registered authentication mappings
- Remove authentication mappings from the translator

The `dfsiauth` command is the tool for administering these functions. It is part of the translator and resides therefore only on the machine running the translator. To access this command, the DCE user who exports their files must log in to the translator machine. They can do so on a local terminal or via a remote login utility (`telnet, rlogin`). The command can be executed, and then the session can be terminated. After this session, the authentication mapping is active, and NFS client users can mount the exported directory to a free local directory and then begin to access the DFS filespace.

Credentials can expire after a certain time (10 hours by default). When DCE credentials expire, the translator user will not be able to use the `kinit` command to renew his credentials. From the user's point of view, when their DCE credentials expire, they will start to experience access permission errors because they become an authenticated user. To re-authenticate, the user must use the translator registration command once again to register the mapping. Then they can continue to access the DFS filespace.

On the NFS client machine, you don't need to unmount and remount the file system.

7.3.4 Administration Tasks for the System Administrator

Administration and configuration of the NFS/DFS Translator involves steps to be performed by the AIX system administrator and by the DCE users who are willing to make their DCE authentication available to the NFS users.

This section covers the administration tasks to be executed by the system administrator on the translator machine. For the administration tasks of the DFS users who make their data accessible from NFS consult 7.3.5, "Administration Tasks for the DFS User" on page 262.

Eventually, to provide file access to the end users, a system administrator on the NFS client machine must mount the exported DFS directory. This is described in 7.3.6.1, "Using the Translator from a UNIX NFS Client Machine" on page 264.

The following tasks of a system administrator are performed on the NFS/DFS Translator site:

- Installing and starting the NFS/DFS Translator
- Exporting DFS to NFS
- Managing expired authentication mappings
- NFS anonymous mappings
- Local ID differences

7.3.4.1 Installing and Starting NFS/DFS Translator

The first step is to install the package *dcedfsnfs.obj,* which includes:

- The AIX kernel extension for the NFS/DFS Translator
- The `dfsiauth` command to register(add), delete, and list authentication mappings
- The libdceiauth.a user library
- The dcedfs/dceiauthapps.h include file for application development
- The System Management Interface Tool (SMIT) screens for NFS/DFS Translator management

Then provide (start) NFS/DFS Translator service by:

1. Verifying DCE/DFS and NFS are configured and running on your system.
2. Loading dfsiauth.ext (the kernel extension) by running the /etc/rc.dfsnfs script file.

The translator can also be started from the SMIT menu by calling `smit dfsnfs`.

Put the /etc/rc.dfsnfs script into the /etc/inittab file to support automatic start-ups. It should be added after the rc.dce and rc.nfs lines in the inittab file because it needs them to be active.

7.3.4.2 Exporting the DFS Filespace to NFS

The DFS filespace should be made available to NFS clients by NFS-exporting the DFS filespace. This means that any portion of the DFS file tree can be exported just like any regular AIX directory.

The top of the DCE tree is /..., global root. Below that is the cell name and the junction point into DFS (/.../cellname/fs). Any portion of this part may be exported. By exporting /..., NFS clients will have access to other cells for which intercell registration is set up. However, in most cases, access to the foreign cells will be unauthenticated access. By exporting /.../<cellname>/fs, NFS client accesses will be limited to a particular cell. Administrators should consider these options when deciding what part of the DFS filespace is to be exported.

However, when you export the DFS filespace to NFS, you can expose it to a decreased level of protection due to the less secure nature of the NFS/RPC compared to the DCE/RPC. Administrators should take precautions against forged NFS requests, replays, and IP address spooling. They should be careful by exporting only to a specific group of machines and not to everybody.

After considering these alternatives, the DFS filespace can be exported to NFS clients.

7.3.4.3 Removing Expired Authentication Mappings

Credentials can expire. See also 7.3.5.6, “Managing Expired Authentication Mappings” on page 264, for explanations how to renew them. Expired mappings are removed by the translator.

In addition, a local system administrator can explicitly clean all expired mappings by using the following `dfsiauth` command:

```
#dfsiauth -flush
```

7.3.4.4 NFS Anonymous Mappings

An NFS server, by default, maps root (UID=0) requests to the nobody (uid=-2) remote requests. NFS administrators can choose to map remote root to local root. With the NFS/DFS Translator, you can also do the same thing by modifying the /etc/exports file. For example, if you have a DCE principal root with UID=0 and you need access to root’s DFS files from an NFS client, then:

1. The NFS *anon* mapping must be set to UID 0.
2. An authentication mapping for UID=0 must be set for the DCE principal root.

7.3.4.5 Local UID Difference

The NFS/DFS Translator adds the authentication information to NFS requests before they are passed to DFS. This authentication mechanism allows the NFS request to become associated with some DCE principal. The DCE value for the UID associated with the principal may be different from the name and UID of the NFS client. In fact, any NFS UID can be mapped to any DCE known principal identifier. This difference can lead to unexpected behavior. For this reason, it is highly recommended to synchronize your client’s local or NIS-maintained /etc/passwd file with the DCE registry or vice versa.

See 6.6, “Integrating an NFS/NIS Environment” on page 212, which demonstrates ways of making UIDs unique.

7.3.5 Administration Tasks for the DFS User

As explained in the previous section, administration and configuration of the NFS/DFS Translator involves steps to be performed by the AIX system administrator and by the DCE user who is willing to make his DCE authentication available to the NFS users.

This section covers all the steps that these DCE users have to execute, which is basically the dfsiauth command. However, our recommendation is to create one or more special users operated by the system administrator because we suppose that you would not want to leave these tasks up to regular users.

Eventually to provide file access to the end users, a system administrator on the NFS client machine must mount the exported DFS directory.

Here are the tasks for a DFS user to perform on the NFS/DFS Translator site:

- Registering (add) authenticating mappings
- Deleting authentication mappings
- List existing authentication mappings
- Setting @sys and @hosts values within a mapping
- Managing expired authentication mappings

7.3.5.1 The dfsiauth Command

The dfsiauth command can be used by either the AIX super user or by a normal AIX user if they have a DCE account. This command is only available on the system where the NFS/DFS Translator is running. To use it, a user must log in into that system, either locally or remotely. The DCE account that is used for a particular mapping must already exist in the DCE security registry.

```
$dfsiauth -?
dfsiauth -add [-overwrite] | -delete -r remote_host -i numeric_uid
[-u principal] [-p password] [-s sysname] [-h host]
Usage: dfsiauth -list [-u principal] [-p password] | -flush
```

The meaning of the options is the following:

-r remote_host	Hostname of the machine requesting authenticated access
-i remote_uid	UID of the user requesting authenticated access
-u principal	DCE principal to authenticate as
-p password	Password of the DCE principal
-s sysname	Associate parameter sysname with the @sys property for the input host/UID pair
-h hostname	Associate parameter hostname with the @host property for the input host/UID pair.
-add	Add the specified mapping information
-delete	Delete the specified mapping information
-overwrite	Change information about an existing mapping. Option only valid with the -add option
-list	List the registered authentication mappings
-flush	Remove expired authentication mappings

The password does not have to be entered in the dfsiauth command in clear text. If it is omitted, the user is prompted for it. The dfsiauth -list command displays the currently active mappings.

For more information, we suggest reviewing the *DCE for AIX NFS/DFS Authenticating Gateway Guide and Reference*.

7.3.5.2 Registering Authentication Mappings

This task is to register your authentication mapping. In order for an NFS client user to have authenticated access to DFS, an authentication mapping must exist that maps the NFS client machine's IP address and remote UID to a DCE principal that has the proper access rights to DFS.

If the authentication mapping does not exist, DFS will determine the NFS client's request for data is unauthenticated. This is equivalent to a DFS user not having previously performed a `dce_login`.

The following command adds an authentication mapping for user ID 107 of NFS client system *ev5*, who will be authenticated as DCE principal *brice*:

```
$dfsiauth -add -r ev5 -i 107 -u brice
```

Note that this command is normally executed by the user brice, but the system administrator can also use this command on behalf of brice if user brice lets this administrator know his (brice's) DCE password.

7.3.5.3 Display Authentication Mappings

You can display the translation mappings by using the following command:

```
$dfsiauth -list -u brice

Host   Uid     Principal       @sys    @host     Expiration
----  ------  ----------      -----    -----     ----------
ev6     107      brice                          5/27/94  00:30
```

7.3.5.4 Unregistering Authentication Mappings

When you no longer need your authentication mapping, you can remove it by using the following command:

```
$dfsiauth -delete -r ev5 -i 107 -u brice
```

7.3.5.5 Setting @sys and @host Variables

DFS uses the @sys and @host variables to access operating system-specific and host-specific files and directories if the administrator has set them up. The DFS client expands @sys and @host names that it encounters to a defined system name or hostname.

The NFS/DFS Translator also has the capability to make these path name substitutions if the @sys and @host values are registered for a host/UID pair. To register a host/UID pair, you use the `dfsiauth` command with values for @sys and @host substitution options.

The following command registers the user with UID 107 from remote host *ev3* as authenticated DCE principal brice. The sysname rs_aix32 is also associated with this mapping.

```
$dfsiauth -add -r ev3 -i 107 -u brice -p my_passwd -s rs_aix32

dfsiauth:<ev3, 107> mapping added
DCE principal:brice
System Type (@sys)rs_aix32
```

For more information, we suggest reviewing the *DCE for AIX NFS/DFS Authenticating Gateway Guide and Reference*.

7.3.5.6 Managing Expired Authentication Mappings

As we previously said, credentials can expire. When DCE credential tickets expire, the NFS client user is not notified. On the NFS client machine, it will appear to the user that authenticated access has been suddenly lost. They start to experience *access denied* errors. If this happens, you (who are a known DCE principal user) should renew the ticket by loging in to the NFS/DFS Translator site. There you must reregister your authenticated mappings by using this command:

```
$dfsiauth -add -r ev2 -i 107 -u brice -p my_passwd -s rs_aix32
```

You will notice that it is exactly the same command as was entered to first set up the mapping.

In addition, the NFS/DFS translator will collect the expired authentication mappings and remove them from the authentication mapping table to avoid overrunning the translator with expired mappings.

In addition to that, a local system administrator can explicitly clean all expired mappings as outlined under the system administrator's tasks.

7.3.6 Making DFS Access Available on the NFS Clients

As outlined above, you need to set up a mapping on the translator machine before you can get authenticated access from any remote NFS client machine. The following three options achieve this before mounting an exported directory to a local one:

- Make a remote login (`telnet`, `rlogin`) to the server machine, and issue the `dfsiauth` command to register your authentication mapping.
- Log in locally on the server machine and do the same task.
- Let a system administrator on the translator machine know your DCE password to do this for you.

As you know, when you perform `telnet` or `rlogin`, you expose your password across the network. So we recommend using this method only in a LAN environment, if at all.

For all systems, check that TCP/IP and NFS are running correctly before trying to mount a directory.

7.3.6.1 Using the Translator from a UNIX NFS Client Machine

If you have already registered your authentication mappings on the NFS/DFS translator, you issue the `mount` command:

```
#mount -v nfs -n ev5 /:/remote_dir /your_local_dir
```

7.3.6.2 Using the Translator from an OS/2 NFS client

There are two ways two mount the DFS filespace from an OS/2 machine:

1. Mount by using the login ID and password.
2. Mount by not providing the login ID and password.

   In this case, you must set some parameters on the OS/2 machine. For example, if you have login ID 107 on the server machine and you belong to a group ID number 100 in the same system, you must do this on the OS/2 system before mounting:

   ```
   >set UNIX.UID=107
   >set UNIX.GID=100
   ```

Then issue the command:

```
c>mount E: nfs_server_machine:/:/remote_dir
```

7.3.6.3 Using the Translator from a PC-NFS Client

If the translation mapping is already done, issue this following command:

```
c>mount E: nfs_server_machine:/:/remote_dir
```

7.4 Integrated Login AIX and DCE

In DCE for AIX Version 2.1, the AIX base operation security services have been integrated with the DCE Security Services. In previous versions of DCE, the user had to log in to AIX first and then to DCE. This required the maintenance of two user IDs and passwords. Furthermore, the DCE login credentials (represented by the environment variable KRB5CCNAME) could only be passed to child windows but not to parent windows or other descendents of the parent window. Depending on your environment, you might have had to log in to DCE several times.

This release of DCE permits the user to see a single-system image rather than separate images of AIX and DCE. Most users will be able to acquire DCE credentials through AIX commands, such as `login` and `su`, to change their password through the AIX `passwd` command and to get information from the standard C library (libc.a) routines, such as `getpw*()` and `getgr*()`. Note, however, that `getpwent()` and `getgrent()` are *not* DCE-integrated. Remote telnet or ftp users will also authenticate with the DCE registry when they access an account set up for DCE authentication.

7.4.1 AIX 4.1+ Authentication Parameters

To support DCE integration with the AIX login, two new user attributes (*SYSTEM* and *registry*) have been defined in the /etc/security/user file. This file defines such things as authentication methods, password policies, or the umask for the user accounts of the local system. It defines default values and allows you to create a stanza for each individual user that may override some or all of the default values. The file also contains lots of explanations.

It is important to note that the user attributes are applied on a domain-relative name basis. That means that a wandering DCE user who logs onto a system as brice, /.:/brice or /.../this_cell/brice, is affected by user attributes that can be present in a stanza for a local user brice in the /etc/security/user, the

/etc/security/limits, and the /etc/security/audit/config files. However, the wandering user is not affected by the local password attributes.

7.4.1.1 The SYSTEM Attribute

The *auth1 (or auth2*) attribute in /etc/security/user specifies a primary (or a secondary) authentication method to access the AIX system. The value SYSTEM means that a system-provided authentication method is to be used in contrast to a user-provided authentication method, which is also possible.

The *SYSTEM attribute* is used to select one or more of the system-provided methods which authenticate a user to the local machine. The valid values for this attribute are Boolean expression strings made up of the following tokens. Their meanings are:

`files`	Use local authentication with the /etc/security/passwd file
`compat`	Use local files and/or NIS
`DCE`	Authenticate through DCE

The value for the SYSTEM attribute can be a complex expression formed by concatenating the methods (tokens) above with the *AND* and *OR* Boolean operators. If the method succeeds, its value is TRUE. To test for other conditions, you can also specify an expected result for the method, and if the method ends with this given result, it evaluates as TRUE. The syntax is `method[result]`. Valid results are:

`UNAVAIL`	Authentication service was unavailable
`NOTFOUND`	User was not found in the database
`FAILURE`	Authentication failed for a different, unspecified reason
`SUCCESS`	Authentication succeeded (default, when no result is specified)

In the following example, the primary method is to verify (authenticate) the user with the DCE registry. If this fails because DCE is not available, authentication can be performed either with the local passwd file or via NIS:

```
auth1 = SYSTEM
SYSTEM = "DCE OR (DCE[UNAVAIL] AND compat)"
```

Explained in more detail, this entry means: The first method to be used is DCE. If it fails, another combination of methods can be tried (*OR*). The second construct will succeed if the DCE method failed because DCE was not available (and not for any other reason) *and* if a compatible method succeeds. In other words, if login DCE fails because the user does not exist or the password is not correctly entered, login fails without trying the compatible methods.

The default stanza in /etc/security/user contains `SYSTEM=compat`. It is globally valid for all users unless it is overwritten in the stanza of a particular user. The local administrator can change this stanza in order to create another default, or he/she can define a SYSTEM attribute in the stanza for users who require another authentication scheme. The local user root should always have `SYSTEM=compat` (as well as `registry=files`) to make sure that the superuser is not dependent on non-local authentication mechanisms.

7.4.1.2 The registry Attribute

The *registry attribute* defines the database where a user's password is administered. This attribute determines where password queries and changes take place. The valid values for this attribute are:

`files`	Use the local /etc/security/passwd file
`NIS`	Use NIS
`DCE`	Use the DCE registry

There is no default value for registry and, as explained later, the administrator can choose to leave the default undefined. Notice that the local user root should always be defined as `registry=files` (as well as `SYSTEM=compat`). In that way, superuser password operations are not dependent on non-local mechanisms.

7.4.1.3 Access Method Identification

After authentication, an *AUTHSTATE variable* is set in the user's environment. This variable is set to the authentication mechanism with which the user was successfully authenticated. It can have a value of *compat*, *NIS*, or *DCE*.

The AUTHSTATE variable determines which database will be accessed for a user's subsequent password operations. However, if the user's *registry attribute* is defined, password operations are directed toward the database defined by registry attribute, regardless of the user's AUTHSTATE.

The AUTHSTATE variable also determines whether the `getpw*()` and `getgr*()` calls look *first* for their information in the local AIX files or in DCE. If the information is not found in the first choice, the others will be searched.

The local administrator should make sure that a user's registry attribute does not conflict with the user's AUTHSTATE, which is determined indirectly by the SYSTEM attribute. One method is to leave the registry attribute undefined for all but local-only users. Local-only users (such as root) should always be defined as `registry=files`.

7.4.2 User Synchronization Between AIX 4.1+ and DCE

In an integrated environment between AIX and DCE, we have to solve two major issues. One is the definition of authentication methods in the /etc/security/user file as described in 7.4.1, "AIX 4.1+ Authentication Parameters" on page 265, and the other one is the mapping of user/group IDs defined on the AIX machines and in the DCE registry.

It is strongly recommended to maintain the DCE registry and local files synchronized as closely as possible. This means that for a particular user or group, the user ID and group ID should be the same on all systems of the cell and in the registry. If this is not the case, a user authenticating with DCE might assume the user ID of another user defined locally on the machine. There are ways to map different IDs, but this can become very complex and confusing.

In this section, we discuss:

- How user IDs and group IDs can be synchronized between AIX and DCE
- How user IDs and group IDs can be mapped between AIX and DCE
- What a synchronized user is and how they are defined
- What wandering users are
- How to protect AIX systems from wandering users

7.4.2.1 Synchronizing User/Group IDs

The /opt/dcelocal/bin/passwd_import command is a mechanism for creating registry database entries from local password and group file entries. If there are duplicate entries, passwd_import follows your directions on how to handle them.

The /opt/dcelocal/bin/passwd_export command creates local password and group files from registry data. Use passwd_export to keep these local files consistent with the registry database. When passwd_export runs, it makes backup copies of the current password and group files, if they exist. The files are named passwd.bak and group.bak, respectively. The passwd_export is commonly run through an entry in root's crontab file. However, with the possibility to define *wandering* users, it might, in many cases, become unnecessary to synchronize local password files with the DCE registry.

7.4.2.2 Mapping DCE User/Groups to AIX Users/Groups

The local administrator can redefine user information for DCE users accessing the local AIX machine by putting the appropriate entries in the /opt/dcelocal/etc/passwd_override or /opt/dcelocal/etc/group_override file. The override information that you enter is in effect only for the local machine, which is the machine on which the passwd_override file is stored. When a user logs into a machine with an override file, any information for the user's account in the override file replaces the pertinent information obtained from the registry.

Entries in the passwd_override file have a format similar to those in a regular passwd file as found on every UNIX system:

```
principal:passwd:UNIX_id:grp_id:GECOS:home_dir:shell
```

The principal, UNIX_id and grp_id fields are key fields. Empty fields are not overwritten; they are accepted by AIX as submitted by the DCE registry. For instance, the following entry would assign the Korn shell to all DCE users coming in with a DCE group ID of 22:

```
:::22:::/bin/ksh
```

If a principal is specified, then the UNIX_id and grp_id fields are used to reassign these values for the local AIX system. An incoming DCE principal that matches a principal field specified in the file gets the values of that entry assigned for the session on the local AIX machine. For example, if DCE user *brice* logs in to an AIX machine with a passwd_override file and the following entry in that file:

```
brice::0:0:Brice:/home/root/:/bin/ksh
```

he becomes root user on this machine. See also 7.4.2.5, "Denying DCE Users/Groups Access to AIX Machines" on page 270, for more information on the usage of the override files.

7.4.2.3 Synchronized Users

In a well-integrated cell, most users are defined and administered in the DCE registry. If some users are also defined locally on any machine, then their local definition should be synchronized with a DCE user. Such a user is referred to as a *synchronized user*.

The passwd_export utility can be used to synchronize all local user information with the DCE registry except for passwords. The AIX passwd user command or the pwdadm administrative command should then be used to synchronize passwords. When a synchronized user uses regular AIX Base Operating System

(BOS) commands that require or maintain the password, commands are automatically directed toward the registry defined by the *registry* or the *AUTHSTATE* attributes. See 7.4.4, “Managing Passwords” on page 271, for more information on synchronizing passwords.

To enable a user to use the DCE registry, the user’s stanza (whether a specific stanza or the default) in /etc/security/user should have a SYSTEM attribute that defines DCE as the first authentication method to try, as shown in the following examples:

```
SYSTEM = "DCE OR (DCE[UNAVAIL] AND compat)"
SYSTEM = "DCE OR (DCE[FAILURE] AND compat)"
SYSTEM = "DCE AND compat"
```

In the above examples, DCE is the first authentication method tried. If authentication passes, the user is granted access; all UNIX-type information (UID, GID, home directory, login shell) as well as DCE credentials are obtained through DCE. In the first example, if DCE fails because of unavailability, local authentication is attempted. The second example attempts local authentication if DCE authentication fails for any reason. The third example requires both DCE and local authentication to succeed before the user is allowed access to the system.

Note: For synchronized users, do not set the registry attribute.

A user’s AUTHSTATE environment variable is set to the first authentication method that succeeds, and subsequent password operations are directed toward the registry defined by AUTHSTATE (see access method identification in this section). If the registry attribute is explicitly set to a registry that conflicts with the AUTHSTATE of the user, password operations can fail.

7.4.2.4 Wandering DCE Users

Wandering DCE users are users defined in the DCE registry but not defined on the local machine. You can enable wandering users to log onto DCE from any machine by setting the default SYSTEM attribute as follows:

```
default:
        SYSTEM = "DCE OR compat"
```

Any DCE user can log on to any machine configured in that way by supplying his or her DCE name and password. Of course, if local users with colliding names or IDs exist and have been protected either by their own stanzas (in the /etc/security/users file) or by entries in the passwd_override or group_override files, a wandering user is denied system access.

If you generally allow access to wandering DCE users, you should set an individual entry for the local root user as shown below:

```
root:
        SYSTEM = compat
        registry = files
```

In this way, all authentication and password operations are directed toward local files.

7.4.2.5 Denying DCE Users/Groups Access to AIX Machines

The local administrator can deny DCE authenticated access to the local machine for a specific principal by putting the appropriate entry in the /opt/dcelocal/etc/passwd_override or /opt/dcelocal/etc/group_override file. For example, to deny local system access to DCE user joe, we can add the following line to the passwd_overrride file:

```
joe:OMIT:::::
```

Similar entries, keyed by user ID or group ID, can prevent DCE authentication to the local system of users having the specified ID.

Another way to prevent wandering DCE users from gaining access to the local system is to exclude DCE from the SYSTEM attribute in /etc/security/user:

```
default:
        SYSTEM = compat
        registry = files
```

Local users who are synchronized with the DCE registry can still acquire DCE credentials at login by having DCE specified as an authentication method in their individual stanzas. For example, we will have this entry in the /etc/security/user for user brice.

```
brice:
        SYSTEM = "DCE OR DCE[UNAVAIL] AND compat"
```

Note: For performance reasons, it is preferable to protect local resources with the /etc/security/user file (if possible), rather than with the DCE override file.

7.4.3 Configuring a System for Integrated Security

To activate the security integration, you must do the following:

1. Ensure that the /usr/lib/security/DCE module is installed on the system.
2. Edit the /etc/security/login.cfg file, and add the following lines:

   ```
   DCE:
        program = /usr/lib/security/DCE
   ```

   This defines the DCE authentication method to the system.
3. Ensure that the dceunixd daemon is running. If not, start it:

   ```
   # dceunixd
   ```

   This daemon communicates to the DCE servers secd and dced on behalf of the AIX commands. Be sure to start the DCE services before dceunixd. You can add this daemon to the /etc/inittab file, but then the command **must** be entered as `dceunixd -d 1`.
4. Edit the /etc/security/user stanza file to define the authentication method(s) for the users. This means setting the *auth1* and *SYSTEM* attributes as explained above.
5. To explicitly prevent certain DCE users from any access to the local system, you should modify the /opt/dcelocal/etc/passwd_override and /opt/dcelocal/etc/group_override files.

Set passwd_override ERA

For users that authenticate via DCE, you should set the passwd_override extended registry attribute (ERA) to 1. This allows them to log in with an expired password. If the passwd_override ERA is 0 or undefined, users are locked out of DCE when the password expires. The DCE administrator would have to set a new password in this case and revalidate the account.

If the ERA is set and the user's password is expired, DCE grants the login and notifies the calling process of the fact that the password is expired. In the case of the integrated login, the AIX login process that performs the authentication with DCE notices that and enforces a change of the password.

7.4.4 Managing Passwords

Password operations are directed toward the registry defined by the registry user attribute or, in the absence of a registry attribute definition, to the registry defined by the AUTHSTATE environment variable. Password operations are not directed to both local and DCE registries in one shot.

Changing passwords for a DCE-only users can be done through dcecp. Changing passwords (both DCE and local) for a synchronized user can be done with the AIX passwd command in a two-step procedure:

```
$ AUTHSTATE=DCE passwd
$ AUTHSTATE=compat passwd
```

The user should echo the AUTHSTATE value before changing it and should set it back to the original value after changing is complete.

Note: Passwords must be kept synchronized for synchronized users. If they are not, either DCE or the local authentication fails. Also, if a user exists locally on more than one machine, the local password must be synchronized on all machines.

7.5 Mass User/Group (and ACL) Management

User management encompasses tasks such as adding, modifying, and deleting users, accounts, and groups. The user namespace or the policy according to which user names and user identification numbers (UIDs) are assigned has to be carefully planned. It might be necessary to keep the names and UIDs unique over multiple cells belonging to the same company. Please refer to 2.4, "Planning the User Namespace" on page 25, for planning information.

Managing single users is well supported within SMIT. Easy-to-use menus allow one to add, modify, and delete users, accounts, groups, and organizations. However, if a security registry has to be populated with dozens or hundreds of users at the same time, using SMIT menus is no longer practical. The following situations may require tools that are able to handle a lot of users:

- First DCE installation
- Migration from Open Network Computing (ONC) or NFS/NIS environments
- Global changes within a cell
- Splitting and joining cells

The purpose of this section is to show how to manage multiple users at the same time. We created sample tools to add, modify, and delete a lot of users. Information about users is managed from a so-called central repository. This central repository is a directory and consists of a file for each user, the user definition file (UDF). This file can be generated with default values or from existing sources, such as an /etc/passwd file, NIS, or a DCE cell. The administration tools then access these files and maintain a state for each user. Values can be modified when the user is in the SUSPENDED state. The resulting changes will be applied upon reenabling the user.

The way the user information is provided can easily be changed, and these scripts can be adjusted to perform additional tasks.

As a matter of fact, our tools go far beyond the scope of just adding, modifying, and deleting users. By providing and even managing ACL information for each user, they actually build a complementary ACL management environment. DCE provided ACL managers can list all users and groups with their access rights to a specific object, but there is no way to answer the question: to which objects does a specific user have any rights?

The following sections describe the idea and tools for mass user management.

Please Note

Our tools, as well as the underlying architecture, should be seen as a working framework for DCE administrators who want to manage large DCE environments in a more effective way. Their usage can be very simple when you rely on the defaults you can set, or it can be used as a sophisticated user and ACL management tool. See 6.4, “Administering Users and Groups” on page 186, for tips on how to use the tools in specific tasks.

They probably need more work or adjustments before they can be applied in specific customer scenarios.

7.5.1 User Identifications, Groups, and Access Rights

Each cell in DCE has a *cell administrator* which has comprehensive privileges in DCE, just like *root* in UNIX. The cell administrator is responsible for user/group management as well as for all other security-related tasks, such as password composition policy, expiration policy, ticket lifetime, and so on. Each cell has a security database, called the registry database. It is located under the directory /opt/dcelocal/var/security/rgy_data.

Users are accessing objects. In order to do so, they need access rights because objects are protected and carry Access Control Lists. ACLs specify which user has which kind of rights. In order to facilitate administration of access rights, users with equal access rights or job profiles can be lumped into groups. Groups, as well as the information on which users belong to which groups, are stored and managed in the registry database as well.

Each object in DCE is represented through the universally unique identifier (UUID), which is a 128-bit string. In DCE, users are solely identified by their UUID. Access rights and ownership to objects are expressed by UUIDs. DCE commands internally use UUIDs, but for display purposes, they look the name up in the registry database and display the user’s *name*. If a user is deleted from

the registry, DCE objects show an orphaned UUID string instead of the name if that user still has any rights on these objects.

DCE also stores with each user and group a user identifier (UID) and a group identifier (GID), similar to UNIX. These IDs, however, need not be the same as in UNIX. Thus, for example, user heinz can have a UID of 123 in UNIX and a UID of 456 in DCE. DCE also encodes the UID/GID into the UUID of the user/group when it creates the principal/group. This makes it impossible to change the IDs after the creation of a user or group.

UNIX commands do not know how UIDs/GIDs are related to principal/group names in DCE. And DCE commands do not care about local UNIX UID and GID definitions. Which UID/GID is considered depends on the context in which a command is executed. Suppose somebody logged in as root (UID 0) in UNIX and cell_admin (UID 100) in DCE. If that person creates a file in the DFS file space, UID 100 becomes the owner of the file, and if the file is in the local file tree, UID 0 becomes the owner. The `ls -l` command in UNIX takes the UID and looks the name up in the /etc/passwd file. The file created in DFS then seems to belong to user guest that has a UNIX UID of 100.

A similar confusion arises when heinz logs in using the integrated login. His DCE UID of 456 will be assigned to him for his AIX session unless an override is defined. When he creates a file, 456 will be the owning UID. Depending on who runs the `ls -l` command now, the *owner name* may appear to be different. Since heinz's login is integrated, he will see *heinz* as the owner because, for him, the UID is resolved with the DCE registry. A local user, for instance root, will see the name of the user defined with that UID in /etc/passwd.

So, if UUID and UID do not match, DFS file ownership may appear differently when the file is looked at with UNIX or DFS commands.

Please Note

The absolutely most important prerequisite in user/group management is that for each user and group, the UID/GID in the registry must match the UID/GID defined for them on any possible UNIX system in the cell.

Synchronizing the users can be achieved with the integrated login in AIX 4.1+ (see 7.4, “Integrated Login AIX and DCE” on page 265).

OS/2 and Windows workstations do not suffer from the same problem because they are not multi-user systems and hence do not have UIDs.

7.5.2 Management-Tool Structure and Overview

AIX DCE 2.1 provides a DCE shell called `dcecp`. It is Tcl (Tool Command Language)-based and covers almost all functions supported by the existing DCE commands, such as `rgy_edit`, `acl_edit`, `cdscp`, and so on. The user-management tools we describe in this chapter were created using the existing DCE commands. Originally, they were developed on AIX DCE V1.3. We tested them on AIX DCE V2.1, and they worked accurately without any change. So, the only changes we made to the tools were to fix some minor bugs.

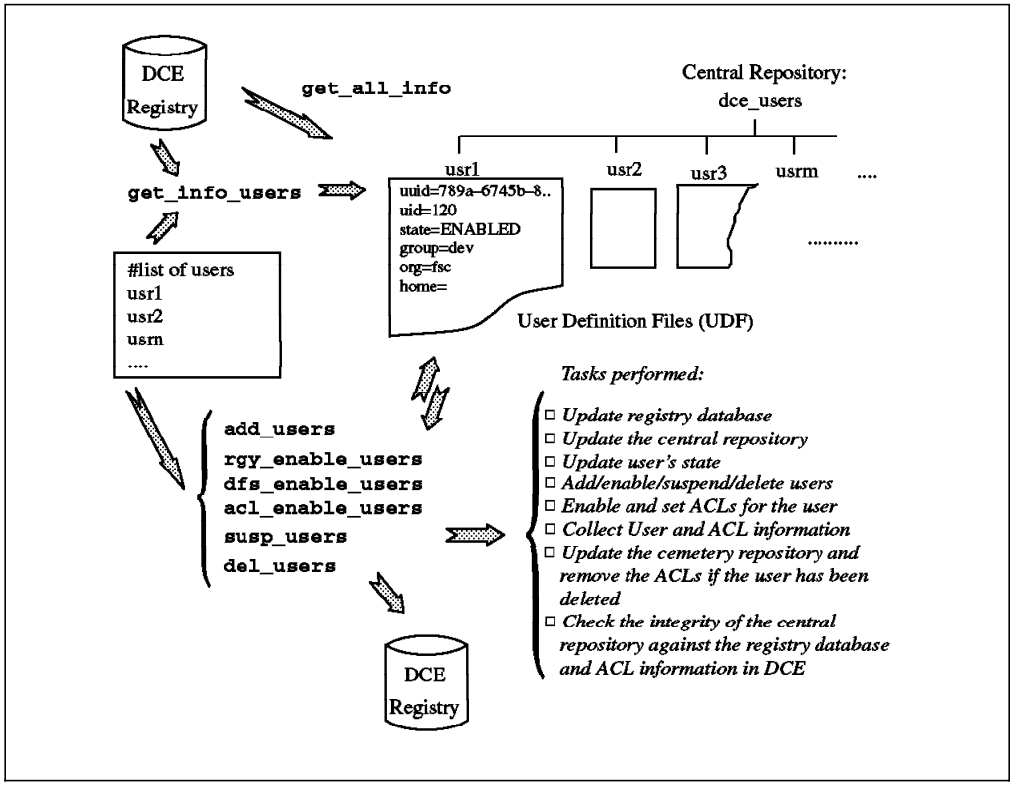


Figure 64. User-Management Workflow

As you can see from Figure 64, the commands are:

`get_all_info` Extracts principal, account, and ACL information for *all* users defined in the DCE registry. Creates a UDF for each user in a (possibly separate) repository directory. This is useful also for integrity checks with the central repository. If the central repository is specified, all existing UDFs are updated, and new ones are created for users who did not have one.

Group definition files (GDFs) are created in the same run.

`get_info_users` Gets information on certain users only and updates the central repository with current information as defined in the registry and in the ACLs of all objects.

Existing DONE records, which reflect recent changes to ACL definitions, are deleted because the currently valid ACLs are extracted from DCE.

`add_users` Adds accounts to the registry database and to the central repository.

This will create principals and basic accounts that are not yet enabled for login. All account attributes are set to their default value, except for the UIDs, which can be predefined in a file in the central repository.

If not already there, a file is created for each user in the central repository.

`rgy_enable_users` Enables accounts for DCE, applying all DCE registry information found in the user definition files, such as the home directories, group memberships, and so on. The users' state is set to RGY_ENABLED.

`dfs_enable_users` This step basically sets all the ACLs for the users' initial containers, their home-directories. It also sets the initial container creation and initial object creation ACLs.

If nothing is specified in a file when this command is used, no ACLs are set. However, the accessibility of the user's home directory is checked and then the state is updated to DFS_ENABLED, assuming the ACLs had already been set upon the creation of the directory.

`acl_enable_users` This step basically sets all the ACLs in CDS and DFS objects for which the users have a user type ACL entry. The state is updated to FULL_ENABLED.

`susp_users` Suspends the accounts for the users and updates their state information. The users can now be moved to another cell, or they can be deleted or re-enabled.

`del_users` Removes the users from DCE. Entries in the DCE registry database and ACLs are removed. The initial containers are not touched; they should be removed with regular AIX or DFS commands.

All of the above commands accept single user names, a file with a list of users, or read user information files from the central repository that contain a certain state. Users can have names such as: usr2, rolf, hans, and so on. But they can also take on the form paris/brice or munich/sal. These types of names will be stored in the central repository with file names like paris%brice and munich%sal.

The approach we used is simple and is not intended to solve all your user-management problems, but it can represent a good foundation for further development.

7.5.2.1 Central Repository

The reason for writing new tools was the lack of tools to add and modify many users at once. This might become necessary for migration processes such as:

- Migrating from AIX to DCE
- Migrating from NIS to DCE
- Splitting a DCE cell
- Joining DCE cells

Another requirement was to manage, at the same time, not only the registry information but also state, ACL, and file system information for users. So the central repository looks like:

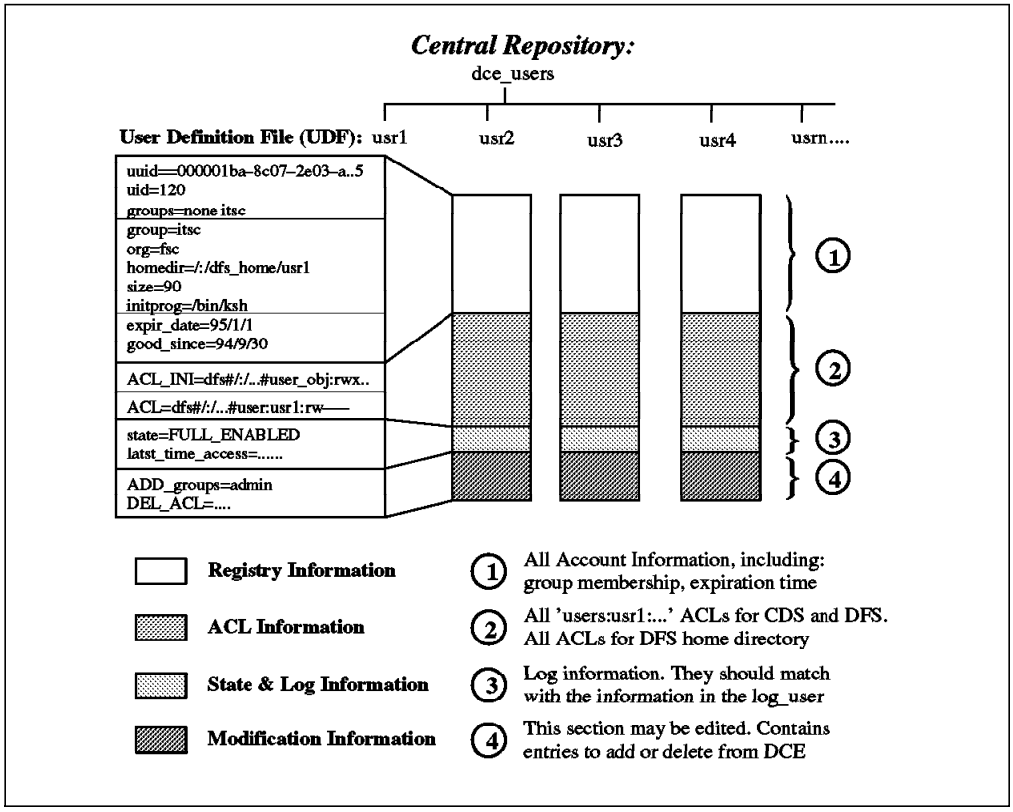


Figure 65. The Central Repository

Organized in this way, the central repository (dce_users directory) contains ACL information related to a specific account or user. This is information that cannot easily be accessed otherwise. With the standard `acl_edit` command, you can only access ACLs per object. However, not all ACL information is covered within the central repository.

One of the main objectives of the ACL entries in the repository was to enable an administrator to clean up all the ACL entries when a user is deleted and to support recreation of the same information in another (or the same) cell.

Which objects are relevant for that purpose?

The answer is: ACLs for objects owned by that user and ACLs for other objects in which that user has a specific entry in the form user:usr1:rw----. To find the latter type of entries, all objects have to be searched. Objects owned by that user are too numerous to capture all ACL information. It could be done, though, but in order to limit these type of object entries in the user definition file, we decided to just manage the DFS home directories.

Our goal was to maximize the coverage of standard situations with a reasonable effort. This led us to the following assumptions:

1. Users do not own CDS objects, but they can have ACL entries in CDS objects in the form user:sal:rw----. This means no ACL_INI entries supported for CDS objects in the UDFs.
2. Users only own DFS objects (files and directories) underneath one single directory (usually their home directory). This means we do not look for other objects owned by the users; we just look up their home directories.

3. Files and directories underneath a home directory all belong to the same user and they do not change any ACLs themselves. This means we can rely on the initial creation ACLs for all new files and directories underneath the home directory.
4. A user can have specific entries in other DFS objects in the form user:sal:r-x---.

In a well-organized production environment, you are very likely to find these conditions, and our tools might possibly be used as they are. Exceptions from these assumptions require special treatment. Depending on how close you are to the ideal environment, you will have to:

- Treat other objects that do not fit into the above-mentioned scheme manually. For instance, run a `find` command to find all files belonging to a certain user, and do whatever you need to do with these files (delete, backup, and so on)
- Edit our user definition files manually or with stream editing commands (`sed`)
- Extend our scripts to do more sophisticated things

ACL management on a per-user basis is actually a big weakness in today's DCE. Maintenance of a global ACL repository should be built into the `acl_edit` command so that integrity is always guaranteed. In order to extract ACL information from the CDS and DFS objects into our central repository, we have to scan through all objects. This will most likely only be done before major reconfigurations or deletion of users. However, modifications to ACL entries for users could always be made in our repository and applied with `acl_enable_users` instead of using `acl_edit`. In this way the central repository would always be up-to-date in this regard without the need to extract the information over and over again.

The central repository can be located anywhere in the file system, for instance on the security server or any other central system. It can even be located in DFS. It is a directory named dce_users and contains a file for each user. Each file represents a user profile. The next section provides details about the attributes in that file.

7.5.2.2 User Definition File (UDF)

The UDF of usr2 could look like this:

```
# -- !!!!!!!!!!!!!!  Do NOT change manually the first part  !!!!!!
# --- Principal info:
uuid=000001b5-76ec-2e02-ad00-10005a4f4165
uid=437
groups=fsc, staff, security
# --- Account info:
group=itso
org=ibm
homedir=/:/dfs_home/usr2
size=487408
initprog=/bin/ksh
expir_date=1995/06/17
good_since=1994/06/17
# --- ACL_INI info:
ACL_INI=dfs#/:/dfs_home/usr2#mask_obj:r-x---
ACL_INI=dfs#/:/dfs_home/usr2#user_obj:rwxcid
ACL_INI=dfs#/:/dfs_home/usr2#group_obj:rwx---
ACL_INI=dfs#/:/dfs_home/usr2#other_obj:r-x---
```

```
ACL_INI_OC=dfs#/:/dfs_home/usr2#mask_obj:r-x---
ACL_INI_OC=dfs#/:/dfs_home/usr2#user_obj:rwxcid
ACL_INI_OC=dfs#/:/dfs_home/usr2#group_obj:rwx---
ACL_INI_OC=dfs#/:/dfs_home/usr2#other_obj:r-x---
ACL_INI_CC=dfs#/:/dfs_home/usr2#mask_obj:r-x---
ACL_INI_CC=dfs#/:/dfs_home/usr2#user_obj:rwxcid
ACL_INI_CC=dfs#/:/dfs_home/usr2#group_obj:rwx---
ACL_INI_CC=dfs#/:/dfs_home/usr2#other_obj:r-x---
# --- ACL info:
ACL=cds#/.:/sec#user:usr2:r----
ACL=cds#/.:/subsys/dce/dfs#user:usr2:r----
ACL=cds#/.:/sec#user:usr2:r----
ACL=dfs#/:/dev/dce#user:usr2:rw----
ACL=dfs#/:/dev/aix#user:usr2:r-----
# --- ACL history (will be consolidated by next "get_info_users")
DONE=ADD_ACL#cds#/.:/hosts#user:usr2:rw
# --- State and last access:
state=FULL_ENABLED
last_time_access=Mon Jun 20 10:55:03 CDT 1994 op=acl_enable_users
#!!
#!!!!!!!!!!!!!!! Do NOT change manually above this line !!!!!!!!!!!!!
#!! Edit below (values that could not be applied):
ADD_groups=g8
#!! Edit below (values to be applied next time):
DEL_groups=fsc
DEL_ACL=cds#/.:/sec#user:usr2
```

The following parameters are filled in and updated only by the user-management procedures and should *never* be edited manually; otherwise you are most likely to introduce inconsistencies. For the administrator, these values are for information only. You can run the powerful UNIX commands, such as **sed**, **grep**, **awk**, **cut**, and so on, against these files and attributes to get useful information about your cell. The user-management tools need this information to determine the current state and definitions for a user to be able to correctly apply changes.

uuid	Universal unique identifier
uid	UNIX user identifier
groups	All groups of which the user is member
group	Primary group
org	Organization
valid	Indicates whether the user may log in or not; should correspond to the state
gecos	UNIX GECOS field for user description
homedir	Home directory of the user
size	Size of the user's home directory
initprog	User's favorite shell
expir_date	Account expiration date
good_since	Account good since date
ACL_INI	ACLs for the initial container, normally /:/dfs_home/user
ACL_INI_OC	Initial Object Creation ACLs for the initial container
ACL_INI_CC	Initial Container Creation ACLs for the initial container

ACL=cds#	ACLs for the user on a CDS directory or object
ACL=dfs#	ACLs for the user on a DFS directory or file
DONE=ACL#dfs#	Reflect changes to ACLs as executed by ADD_ACL or DEL_ACL instructions; they are consolidated and deleted with the next get_info_users command
state	Account state information
last_time_access	Logs time and operation of the last access to this user definition file

The next section of the file contains values that should have been applied in the last enabling process and failed for some reason.

ADD_groups=g8	The user should have been added as a member to group g8, but the group did not exist. The entry is left there for a later update or for manual deletion.

The last section of the file is where an administrator is allowed to specify modifications. He can add or delete values. In order to do so, he should suspend the users. During the next re-enabling steps, all the values defined here are applied. When this is successful, the value is deleted from the modification section, and the according values in the top section are updated to reflect the correct situation. If one value fails to be applied, it is moved to the section values that could not be applied. The following are valid modification values:

ADD_uid	This is to predefine the UID; this is only applied with the add_users command
ADD_groups	A list of groups to which the user is to be added as a member
DEL_groups	A list of groups from which the user is to be removed as a member
ADD_newgrp	(Re)define the primary group
ADD_neworg	(Re)define the organization
ADD_gecos	Add or change the GECOS field
ADD_homedir	(Re)define the home directory
ADD_initporg	(Re)define the shell or initial program
ADD_ACL_INI	Define a new ACL entry for the home directory (or for which the user is the owner)
ADD_ACL_INI_CC	Define a new initial container creation ACL entry for the home directory (or for which the user is the owner)
ADD_ACL_INI_OC	Define a new initial object creation ACL entry for the home directory (or for which the user is the owner)
ADD_ACL	Define a new CDS or DFS ACL entry for the user
DEL_ACL*	All of the ADD_ACL-entries have a counterpart that allows removal of an ACL entry from an object

The expiration date and the good-since date can be defined as part of the environment and will be applied in the same way for all users. See 7.5.2.4,

"General Structure and Customization" on page 281, for global configuration options.

7.5.2.3 Account State Information

The state in the central repository can provide cell administrators with information such as:

- The number of enabled users
- The number of suspended users
- The users that have been deleted

The state information is used by the different scripts and commands during the decision process about which files need to be touched. To answer the above questions, an administrator can run UNIX commands such as `grep`, `sed`, `awk`, and so on against the UDFs.

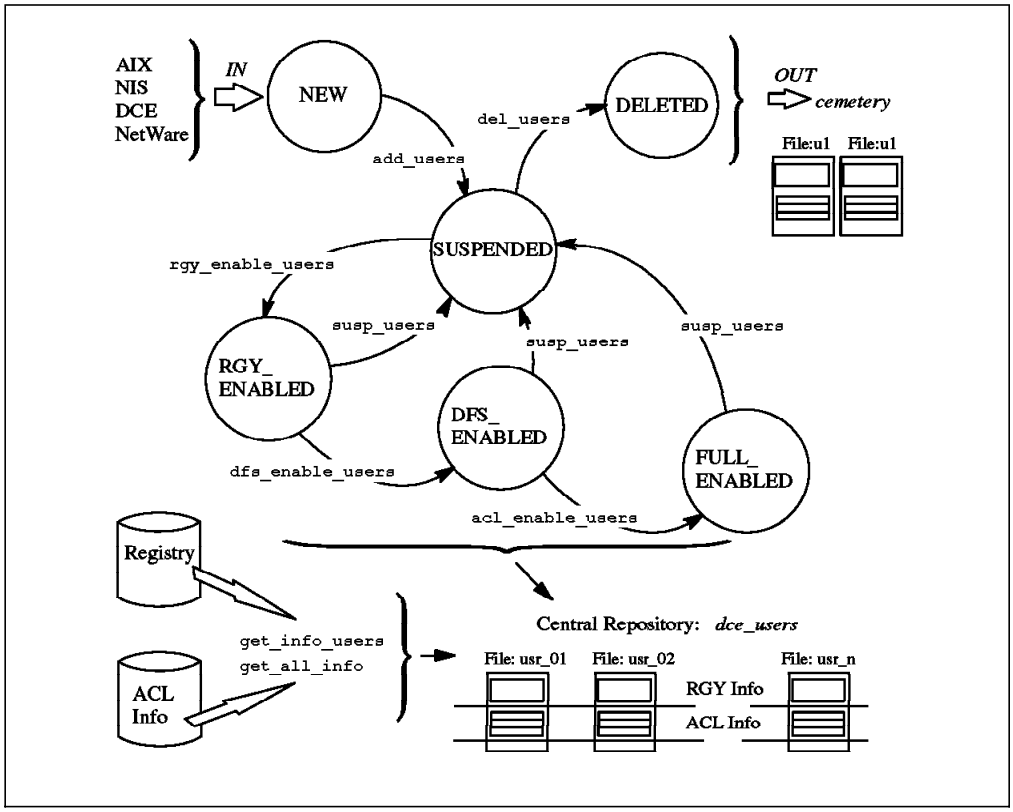


Figure 66. DCE User-State Diagram

As shown in Figure 66, each user defined with these set of tools is always in a certain state. The defined commands are used for state transition. The following states are used:

NEW
: Users can be added without a UDF. Then they get default values for their UID. If a UID needs to be predefined, a UDF has to be created, and the state has to be set to NEW. The command `CR_EMPTY_UDF` creates an empty UDF. UDFs can also be created for instance from an /etc/passwd file.

SUSPENDED
: The account for the user is defined but not enabled for login. Files in this state can be edited. Changes will be applied by a subsequent `rgy_enable_users` command.

RGY_ENABLED The account for the user is enabled in the DCE registry and is working in DCE. No ACL information is applied yet.

DFS_ENABLED The account for the user is enabled in the DCE registry, and the user's home directory is in DFS and has the correct ACLs set.

FULL_ENABLED The account for the user is enabled in the DCE registry, and all defined ACLs are generated.

DELETED The user has been removed from DCE and the central repository and moved into the cemetery repository.

An account cannot be deleted from DCE if it is not in a SUSPENDED state. So, before deleting an account, the cell administrator needs to suspend it. The suspension state disables the user and prevents the user from logging into DCE, but it keeps all information about group membership and ACLs. However, it does not force a user out of DCE. As long as their ticket from a previous login is valid, they can work. The suspension state is a transition state. Together with NEW, it is the only state in which the UDFs should be edited.

If users need to be modified, they have to be put into the SUSPENDED state first. This is the way it is implemented now to keep everything in an order. This means, for instance, to change an ACL requires the sequence susp_users, rgy_enable_users, dfs_enable_users, and finally acl_enable_users. This is not a big penalty because none of the steps does unnecessary processing; they just change the state.

7.5.2.4 General Structure and Customization

Our user-management tools are contained in the tar file ugmgt.tar. They expand in the current directory, no matter what directory you choose.

1. Primary commands as described in this publication:

```
./umgt
./get_all_info
./get_info_users -> umgt
./add_users -> umgt
./rgy_enable_users -> umgt
./dfs_enable_users -> umgt
./acl_enable_users -> umgt
./susp_users -> umgt
./del_users  -> umgt
```

2. Internal commands used by the primary commands:

```
./ADD_USER
./DEL_USER
./SUSP_USER
./RGY_ENABLE_USER
./DFS_ENABLE_USER
./ACL_ENABLE_USER
./GET_PRINCIPAL
./GET_ACCOUNT
./GET_ACL
./GET_ACL_INI
```

3. Internal commands that are partly configurable:

```
./ENVIRONMENT
./READ_UDF
./WRITE_UDF
./LOG
```

4. Directories (repositories):

```
./cemetary_users
./dce_users
./dce_groups
./tmp
```

In order to run the commands, you must be in this directory. All commands, including the internal commands, can be called with a -h flag, which displays the purpose of each command. All scripts are extensively commented.

The primary commands, such as add_users, rgy_enable_users, and so on, are all linked to the same script umgt because they basically all perform the same task. They prepare a candidate list of users with the correct state and call the right internal command.

The commands in capital letters are those which are called internally by the primary commands. They all accept an unlimited number of user names as arguments and could just as well be called manually. However, if you call them manually, there is no state checking performed, and the user definition is applied in the registry as they appear in the file.

The ENVIRONMENT File: All of the scripts read their environment variables from the script ENVIRONMENT. This allows you to configure certain things at one place for all commands:

USER_PWD	Initial password supplied for each new account.
CENTRAL_REPOS	The default directory name for the repository is dce_users in the directory where all commands are located. This can be changed to any other directory and path name.
DFS_STARTDIR	In well-structured file systems, the files to which users have specific ACL entries are probably concentrated into only a subtree of the entire file system. By setting this variable, an administrator can limit the scanning process for ACL entries to a subtree.
DEF_EXPIR_DATE	Specifies an expiration date for login accounts. It is currently calculated to be one year from the current date. This will be applied to every account, whenever rgy_enable is called.
DEF_GOOD_SINCE	Is set to the current date.
LOGFILE	Name of the log file.

The READ_UDF File: This is the place to define all default values for new accounts. This script reads the UDF and is called from all the internal commands. The first section of this file can be edited:

```
# ------------------------------------------------------------------
# Assign the default values ($1 is basically the user name):
# ------------------------------------------------------------------
# !!!!!! DO NOT use a ':' in GECOS. Otherwise parsing will be corrupted
gecos="Account for $1"
homedir=/:/dfs_home/$1
initprog=/bin/ksh
expir_date=$DEF_EXPIR_DATE
good_since=$DEF_GOOD_SINCE
```

The rest of the script first resets all values, then reads the values from the UDF, and finally assign the values to variables used in the other scripts. If new parameters will be introduced, an entry has to be made in the big case statement. Otherwise, the parameter will be overlooked.

The WRITE_UDF File: This script is called from all other routines to write the UDF. By changing this script, the outlook of the UDF can be changed.

7.5.2.5 Migration from Other Environments

To create a tool which translates specific user-definition information, such as an /etc/passwd file into UDF format, you can use the READ_UDF and WRITE_UDF scripts.

In READ_UDF you will see what variables can be set. A conversion tool has to perform the following steps for each user:

1. It needs to execute the `. READ_UDF` command, which assigns the default values to the new user file.
2. Then it must call `. WRITE_UDF` to write the upper part of the file with the default values.
3. Finally, it needs to interpret the values in the old environment and to create and append ADD instructions to the UDF.

The `pwd2dce` procedure is provided on the diskette that comes with this publication. It is essentially the same procedure as performed by `nis2dce_users`, which is listed in 6.6.2.3, “The nis2dce_users Procedure” on page 217.

When UDFs are created from an old environment, they should be carefully inspected before the users are added to DCE.

7.5.2.6 Integrity of the UDF

There are two ways of maintaining integrity in the user definition files:

1. If you do not care whether there are inconsistencies between the UDFs and what is defined out there in the DCE registry and/or in the CDS or DFS ACLs, you can just run `get_info_users`. This command gathers all information for the specified users from DCE and overwrites the UDFs. So, this basically just refreshes the UDFs.
2. If you want to know what might be inconsistent, you run `get_all_info` and extract the information for all users into a separate directory. You can then compare the UDFs in the central repository with the files in the new directory with regular UNIX commands such as `diff`.

7.5.3 Group Management

It was very easy to adapt the user-management concepts to group management. We decided anyone who intends to use the user-management tools will also want to treat groups in the same way. The concept for the group management is basically the same as for users, and the commands work in the same way.

Note: The members of groups are managed through UDFs rather than through GDFs. When a group is created, it has no members. Users are then added to or deleted from groups with ADD_groups or DEL_groups entries. This is a more natural way for defining group memberships. When you add a new user, you want to specify a list of groups he/she is supposed to belong to, as opposed to

updating every group with a new member. GDFs show their members for information only. To make this happen, you must run the get_info_groups or the get_all_info commands. Groups cannot be deleted as long as they contain members.

7.5.3.1 Group State Overview

Please refer to 7.5.2.3, “Account State Information” on page 280, for the general idea of the states and the commands that are possible in each state.

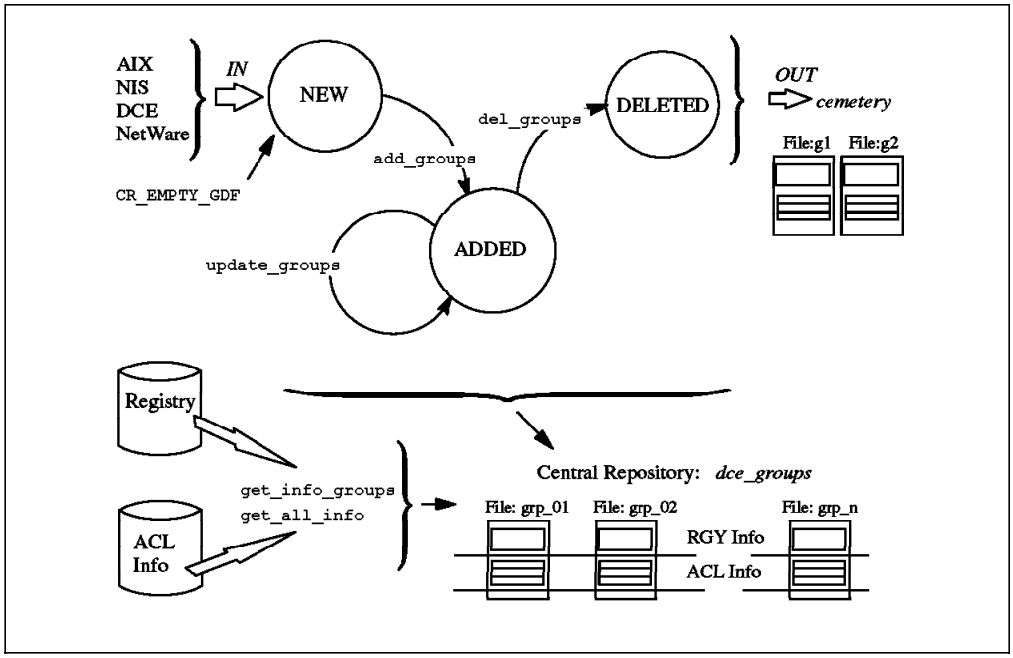


Figure 67. DCE Group-State Diagram

The following states are used:

NEW — Groups can be added without a GDF. They receive default values for their GID. If a GID needs to be predefined, a GDF has to be created, and the state has to be set to NEW. The command CR_EMPTY_GDF creates an empty UDF. GDFs can also be created, for instance, from an /etc/group file.

ADDED — The group is defined in the registry. Files in this state can be edited. Changes will be applied by a subsequent update_groups command.

DELETED — The group has been removed from DCE and the central repository and moved into the cemetery repository.

A group cannot be deleted from DCE if it still has members or if previous ADD_ACL or DEL_ACL entries have been executed, but the GDF is not consolidated yet, which means it still contains DONE entries.

7.5.3.2 Group Commands Overview

Please refer to 7.5.2, “Management-Tool Structure and Overview” on page 273, for an overview on how the user management tool is structured.

The commands needed for group management are:

get_all_info — Is the same as described for user management. It extracts all user and group related information from the DCE registry and writes the UDFs and GDFs.

get_info_groups — Gets information on certain groups only and updates the central repository with current information as defined in the registry and in the ACLs of all objects.

add_groups — Adds groups to the registry database and to the central repository. If the GDF already exists, the group is created with the specified GID.

If it is not there yet, a file is created for each group in the central repository, and the GID is automatically assigned.

update_groups — This step basically sets all the ACLs in CDS and DFS objects for which the groups have a group type ACL entry. The state remains ADDED.

del_groups — Removes the groups from DCE. Entries in the DCE registry database and ACLs are removed. If the group still has members, the group is not deleted. You must first delete that particular group membership from each user. This prevents any accounts from being deleted together with the group, if this was their primary group. If ACLs cannot be removed or if there are still any DONE entries in the file, the group is not deleted either. This ensures that no orphaned UUIDs are left in object ACLs.

All of the above commands work with the same logic as the ones for user management. The group repository is dce_groups.

7.5.3.3 Central Group Repository

The central repository for groups is the directory dce_groups. It works the same way as the user repository, which is described in 7.5.2.1, “Central Repository” on page 275.

7.5.3.4 Group Definition File

The GDF of g7 could look like this:

```
# -- !!!!!!!!!!!!!!  Do NOT change manually the first part  !!!!!!
# --- Group info:
uuid=00000070-77c4-2e88-8801-02608c2fff91
gid=112
# --- Memberships (information only; managed via principals):
users=cell_admin
# --- ACL info:
ACL=cds#/.:/hosts#group:g7:-w-t-
# --- State and last access:
state=ADDED
last_time_access=Wed Sep 28 11:44:02 CDT 1994  op=GET_ACL
#!!
#!!!!!!!!!!!!!!! Do NOT change manually above this line !!!!!!!!!!!!!
#!! Edit below (values that could not be applied):
DEL_ACL=cds#/.:/hosts#group:g7
#!! Edit below (values to be applied next time):
```

The following parameters are filled in and updated only by the user-management procedures and should *never* be edited manually; otherwise you are most likely to introduce inconsistencies:

`uuid`	Universal unique identifier for the group
`gid`	UNIX group identifier
`users`	List of group members (for information only)
`ACL=cds#`	ACL entry for the group in a CDS directory or object
`ACL=dfs#`	ACLs for the user on a DFS directory or file
`state`	Group state information
`last_time_access`	Logs time and operation of the last access to this group definition file

The next section of the file contains values that should have been applied in the last process, which failed for some reason. Or they did not have to be executed, because a `get_info_groups` command was run.

`DEL_ACL=g8`	An ACL entry for a CDS object should have been updated or created. The entry is left there for a later update or for manual deletion.

The last section of the file is where an administrator is allowed to specify modifications. They can add or delete values. If one value fails to be applied, it is moved to the section `values that could not be applied`. The following are valid modification values:

`ADD_gid`	Predefines the GID; this is useful with the `add_groups` command only
`ADD_ACL`	Defines a new CDS or DFS ACL entry for the group
`DEL_ACL`	Deletes a CDS or DFS ACL entry for the group

7.5.3.5 General Structure and Customization

Our group-management tools are contained in the tar file ugmgt.tar. Together with the user-management commands, they expand in the current directory, no matter what directory you choose.

1. Primary commands as described in this publication:

```
./gmgt
./get_all_info
./get_info_groups -> gmgt
./add_groups -> gmgt
./update_groups -> gmgt
./del_groups  -> gmgt
```

2. Internal commands used by the primary commands:

```
./ADD_GROUP
./DEL_GROUP
./ACL_ENABLE_GROUP
./GET_GROUP
./GET_ACL
```

3. Internal commands that are partly configurable:

```
./ENVIRONMENT
./READ_GDF
./WRITE_GDF
./LOG
```

4. Directories (repositories):

```
./cemetary_groups
./dce_groups
./tmp
```

In order to run the commands you must be in this directory. All commands including the internal commands can be called with a -h flag, which displays what the purpose of each command is. All scripts are extensively commented.

7.5.3.6 Migration from Other Environments

To create a tool that translates specific group definition information, such as an /etc/group file into GDF format, you can use the READ_GDF and WRITE_GDF scripts.

In READ_GDF you will see what variables can be set. A conversion tool has to perform the following steps for each group:

1. It needs to execute the . READ_GDF command, which assigns the default values to the new group file.
2. Then it must call . WRITE_GDF to write the upper part of the file with the default values.
3. Finally, it needs to interpret the values in the old environment and to create and append ADD instructions to the GDF.

The grp2dce procedure is provided on the diskette that comes with this publication.

When GDFs are created from an old environment, they should be carefully inspected before the groups are added to DCE.

7.5.4 Adding Users: add_users

The add_users procedure is a simple shell script that adds principals and accounts to DCE. After execution of this step, the accounts are not yet enabled for login; their state is SUSPENDED. If there is no user definition file for a user in the central repository, this step creates one.

There are special scripts to create user definition files from the following user definition environments:

- Network Information System
- AIX
- DCE

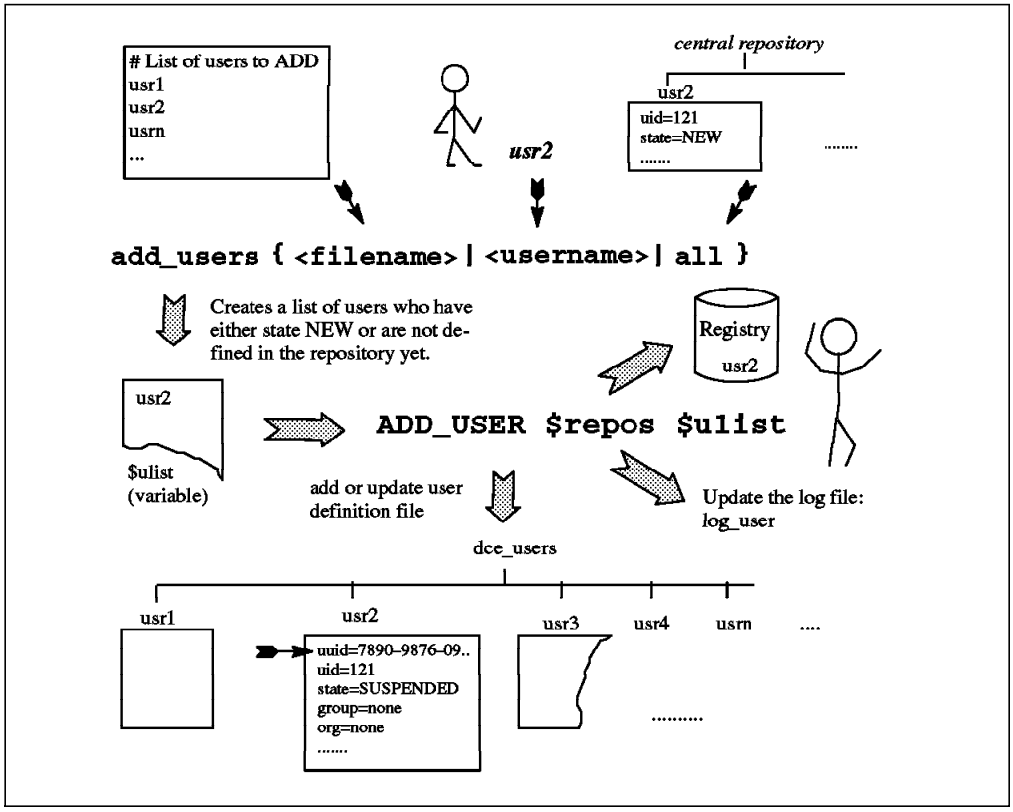


Figure 68. The add_users Procedure

7.5.4.1 Syntax

The command takes one of the following forms:

```
add_users <username>
add_users <filename>
add_users all
add_users -h
```

7.5.4.2 Arguments

`username`	Single user name to be added, such as joe or austin%joe
`filename`	File containing only user names
`all`	Keyword that indicates that the central repository should be searched for all users in the state NEW
`-h`	Displays more information on the purpose of the program

7.5.4.3 Description

The command requires exactly one argument. If it is not `all`, it checks whether it is a file name. If not, it assumes it is a user name. If the argument is a file, that file may only contain user names or comment lines that begin with #.

The first step is to evaluate a candidate list of potential users to be added. New users may either be added without being defined in the central repository at all or from user definition files which need to have their state set to NEW. If the argument is a file or a single user, these prerequisites are checked before a user is added to the candidate list. If the argument was `all`, then all users in the central repository with state NEW yield the candidate list. The candidate list is

simply an environment variable, $ulist, that contains all the approved user names.

The user definition files can be prepared by running CR_EMPTY_UDF new_file repos and filling in the correct values with an editor, or they can be generated from other sources, such as an /etc/passwd file. Consult 7.5.2.1, “Central Repository” on page 275, for information about which parameters may be set.

Internally, the add_users procedure then calls ADD_USER $repos $user_list. The administrator is told how many users are going to be added and is prompted for the password:

```
You are going to add ″nn″ users
Starting to work with rgy_edit ...

Please provide your password:
```

If user definition files are used and UIDs are specified, these UIDs are checked to see if they are already in use. If this happens or if the user is already defined, an error message is displayed and entered into the log file.

The account is then created with all values defined in the UDF but not enabled for login in DCE. In order to enable the user for login and to apply the rest of the attributes, you must run the rgy_enable_users procedure. A file for each new user is then created in the central repository, or it is updated if it had been previously defined.

At the end of the of the adding operation, an additional message might be given to the administrator telling him which users have failed to be added and indicating the reason for the failure:

```
The following users/accounts were not good for ″add″,
please check again:

paris/brice excluded from list. Reason: state not NEW
ADD_USER daemon failed. Reason: Principal already exists
ADD_USER guest failed. Reason: UID already exists

you can find them in the file: .../tmp/no_good_users
```

The result of the operation is logged in the file log_user. For each user, there will be an entry telling whether he was successfully added or not. For all unsuccessful entries, the reason for the failure is indicated:

```
account:a2 date:Tue Jun 21 11:27:31 CDT 1994 op=add_users from_host:ev4
aix_user:root result=SUCCESS
account:adm date:Tue Jun 21 11:27:32 CDT 1994 op=add_users from_host:ev4 \
aix_use r:root result=FAILURE why=uid already exist
account:usr2 date:Wed Jun 22 18:19:02 CDT 1994 op=enable_users from_host:ev4 \
from_user:root op_result=FAILURE why=user not in NEW state
```

After successful creation of a user, his new state and the attribute last_time_access is entered to that user’s file in the central repository:

```
state=SUSPENDED
last_time_access=Tue Jun 21 11:28:44 CDT 1994 op=add_users
```

The file is now ready for the rgy_enable_users command. If other than the default values need to be specified, they should now be filled in the UDF.

7.5.4.4 Implementation Specifics

The add_users script is a link to the umgt script, because the checks to be performed are the same for user management scripts.

The kernel of the ADD_USER command, which is internally called, are the following few rgy_edit commands:

```
rgy_edit << EOF >/dev/null 2>&1
domain principal
add $princ_name $uid
quit
EOF

rgy_edit << EOF >/dev/null 2>&1
domain account
add $princ_name -g none -o none -anv -pw "$USER_PWD" -mp "$PASSWD" -pnv
quit
EOF
```

The option account not valid (-anv) is provided; so the user is added as account but cannot log in to DCE because he/she is in the SUSPENDED state.

7.5.4.5 How to Specify UIDs

The only way to predefine specific UIDs for the add_user function is to predefine a user definition file in the central repository and set its state to NEW. You can predefine any of the supported attributes in the user definition files (see also 7.5.2.1, “Central Repository” on page 275):

- ADD_uid=120
- ADD_newgrp=dev
- ADD_homedir=/:/dfs_home/usr2
- ADD_initprog=/bin/csh
- state=NEW
- .....etc.

Only the add_users command can apply the UID because this value cannot be changed later on. The rest of the attributes can be defined at any time. This might be important when you move users from NFS or AIX into DCE and want to keep their UID assignment.

Another way to control UID assignment to a certain extent is to set the lowest UID for principal creation registry property to a value of your choice, for example to 2400:

```
# dcecp
dcecp> registry show
.....
.....
{minuid 100}
.....
.....
dcecp> registry modify -minuid 2400
dcecp> registry show
.....
.....
{minuid 2400}
.....
.....
```

By running this procedure each time before executing the add_users command, you can at least control the range of each portion of users you want to add. This lowest UID can be set anytime and does affect already existing accounts.

7.5.4.6 Error Checks and Messages

The add_users command does the following checks:

- If no argument is given, the add_users command displays the following message:

```
Usage:  add_users <username>
        add_users <filename>
        add_users all
        add_users -h

        <filename>      = File with a list of user names
        <username>      = String like joe or austin%joe
        all             = Extracts info for all users in the according state
        -h              = Display more information
```

- If the current user is not cell administrator, the following message is displayed:

```
Checking to be sure you are cell_admin
 You must login as cell_admin first ... sorry
```

- If a list of users is given as a file, it first checks if the file is not an empty list. If it is, the following error message is displayed:

```
A file must contain something !!!
```

- If the single user or some of the specified users of <filename> already have a user definition state, but it is not NEW, the following message appears:

```
Creating a candidate list of users to add:
  Checking user97 ...not ok   (state not NEW)
```

- If users or UIDs of the candidate list already exist in the DCE registry, the following message appears on the screen:

```
  Checking user95 ...failed   (Principal already exists)
  Checking user98 ...failed   (UID already exists)
```

- If the candidate list is empty because the specified users have a user definition file, but none of them is in the state NEW, the following message is displayed:

```
All users have files in the central repository,
but their state is not NEW.
```

7.5.4.7 Initial Password

The initial password for each user will be set with an environment variable that can be set by editing the file ENVIRONMENT. All users will get the same initial password.

The uglier the password, the more likely a user will actually change it upon first login. Unfortunately, even if the option *password not valid* is given in the add command line (-pnv flag), DCE won't force the users to change the password the first time they log in. DCE will simply display the message:

```
Password must be changed!
```

It is a good practice for the cell administrator to check with a simple script if the user has changed the password or keeps the initial one. The password composition and management in DCE is minimal and does not comply with the Green Book from the Department of Defense (DoD) nor with the Minimum Security Requirements for Multi-User Operating Systems from the National Institute of Standard and Technology (NIST). As a matter of fact, DCE warns you if you enter a wrong user or a wrong password:

```
# dce_login user_does_not_exist
Sorry.
User Identification Failure. - Registry object not found (dce / sec)
# dce_login cell_admin
Enter Password:
Sorry.
Password Validation Failure. - Invalid password (dce / sec)
```

This provides hackers with valuable information for guessing users and passwords in DCE. To solve this problem, AIX DCE 2.1 provides Extended Registry Attributes (ERA). By using ERA, administrators can set a limit on the number of invalid login attempts before the account is disabled. The *Understanding OSF DCE 1.1 for AIX and OS/2* redbook explains these ERAs in detail.

7.5.5 Enabling Users for DCE Login: rgy_enable_users

The rgy_enable_users procedure enables users to log in to DCE. It applies all the attributes defined for each user in their user definition file and enables the account. The procedure basically works in the same way as add_users shown in Figure 68 on page 288. The difference is that the state must be SUSPENDED and the RGY_ENABLE_USER $repos $ulist is called.

7.5.5.1 Syntax

The command takes one of the following forms:

```
rgy_enable_users <username>
rgy_enable_users <filename>
rgy_enable_users all
rgy_enable_users -h
```

7.5.5.2 Arguments

username	Single user name to be enabled, such as joe or austin%joe
filename	File containing only user names
all	Keyword that indicates that the central repository should be searched for all users in state SUSPENDED
-h	Displays more information on the purpose of the program

7.5.5.3 Description

The command requires exactly one argument. If it is not all, it checks whether it is a file name. If not, it assumes it is a user name. If the argument is a file, that file may only contain user names or comment lines that begin with #.

The first step is to evaluate a candidate list of potential users to be enabled. Each user name derived from the input arguments is checked to see if it has a user definition file and whether its state is SUSPENDED. If these conditions are not met, an error message is created.

This command is also used for updates for already-enabled users. The users who need to be updated have to be suspended. Then the user definition files may be edited.

The next step is to call RGY_ENABLE_USER $repos $ulist, where $ulist is the list of candidates to be enabled. This procedure gets all registry-relevant attributes from the user definition file or assigns a default value if a certain attribute is not defined. The default values can be specified in the ENVIRONMENT and the READ_UDF procedures as described in 7.5.2.4, “General Structure and Customization” on page 281. Then the DCE rgy_edit command is called to update the account in the registry.

That command might fail for some or all of the users. If this happens, an error message is displayed. Reasons can be:

- User does not exist
- Primary group does not exist
- Primary organization does not exist

Other minor errors may be discovered, for instance, if one of the other groups the user is supposed to be a member of does not exist. In such cases a warning is issued, but the account is enabled anyway. At last, the user definition file is updated, if necessary. In this case the entry of the UDF is left in the file so that it can be applied later on or deleted by the administrator.

7.5.5.4 Implementation Specifics

The rgy_enable_users script is a link to the umgt script, because the checks to be performed are the same for all user management scripts.

The kernel of the RGY_ENABLE_USER command is the following rgy_edit command, shown for a specific user *usr2*:

```
rgy_edit << EOF >/dev/null 2>&1
domain account
change usr2 -ng itso -no ibm -h /:/dfs_home/usr2 -d /bin/ksh \
-x ″one year from the current date″ -gsd ″current date″ -av
quit
EOF
```

7.5.5.5 Example

Let us assume we want to make the following changes for usr2:

- Change the primary group to security
- Add usr2 to group g8
- Delete usr2 from group fsc
- Delete an ACL entry that usr2 has on CDS object /.:/sec

The following entries have to be made at the end of the user definition file:

```
# -- !!!!!!!!!!!!!!  Do NOT change manually the first part  !!!!!!
# --- Principal info:
uuid=000001b5-76ec-2e02-ad00-10005a4f4165
uid=437
groups=fsc, staff, security
# --- Account info:
group=itso
   .......
   .......
state=SUSPENDED
```

```
last_time_access=Mon Jun 20 10:55:03 CDT 1994 op=acl_enable_users
#!!
#!!!!!!!!!!!!!!! Do NOT change manually above this line !!!!!!!!!!!!!
#!! Edit below (values that could not be applied):
#!! Edit below (values to be applied next time):
ADD_newgrp=security
ADD_groups=g8
DEL_groups=fsc
DEL_ACL=cds#/.:/sec#user:usr2
```

Let us further assume that group g8 does not exist. The directive to add usr2 to group g8 is left in the file, whereas the other entries which were successfully applied are removed or embedded into the information in the upper part of the file. After running the rgy_enable_users, the UDF will look as follows:

```
# -- !!!!!!!!!!!!!!  Do NOT change manually the first part  !!!!!!
# --- Principal info:
uuid=000001b5-76ec-2e02-ad00-10005a4f4165
uid=437
groups=staff, security
# --- Account info:
group=security
   .......
   .......
state=RGY_ENABLED
last_time_access=Mon Jun 20 10:55:03 CDT 1994 op=rgy_enable_users
#!!
#!!!!!!!!!!!!!!! Do NOT change manually above this line !!!!!!!!!!!!!
#!! Edit below (values that could not be applied):
ADD_groups=g8
DEL_ACL=cds#/.:/sec#user:usr2
#!! Edit below (values to be applied next time):
```

The ACL entry was not deleted because adding or deleting ACL entries are only done when acl_enable_users is executed.

7.5.6 Enabling the Users Home Directory: dfs_enable_users

The dfs_enable_users procedure basically applies ACL definitions on the users' DFS home directories, including the initial container creation and initial object creation ACLs. It represents a way to administer individual ACL definitions for each user's home directory.

The home directories must exist prior to running this command. They must be created separately. Also, basic file access permissions and ownership must be set on these home directories. An example for these actions is given in 6.6.4.2, "Creating and Mounting the Filesets for Home Directories" on page 223. The tools could easily be extended to perform the creation of home directories and to set the ownership and initial permissions.

If an administrator decides to first define all ACLs manually, no ACL_INI definitions in the UDFs are needed. The ACL_INI definitions can also be generated later from what is actually defined on the DFS directories by running get_info_users. If the administrator wants to set user ACLs later on with acl_enable_users, acl_enable_users must be run to set the state to DFS_ENABLED, even if no ACL_INIs are defined.

7.5.6.1 Syntax

The command takes one of the following forms:

```
dfs_enable_users <username>
dfs_enable_users <filename>
dfs_enable_users all
dfs_enable_users -h
```

7.5.6.2 Arguments

username	Single user name to be processed, such as joe or austin%joe
filename	File containing only user names
all	Keyword that indicates that the central repository should be searched for all users in state RGY_ENABLED
-h	Displays more information on the purpose of the program

7.5.6.3 Description

The command requires exactly one argument. It follows the same preparation steps as rgy_enable_users except for the state, which has to be RGY_ENABLED.

The next step is to call DFS_ENABLE_USER $repos $ulist, where $ulist is the list of candidates to be enabled. This procedure gets all ADD_ACL_INI and DEL_ACL_INI attributes from the user definition file and tries to apply the values to the specified DFS directories. The ones that cannot be applied remain in the UDF. If they can be applied, a DONE_INI entry is created in the UDF for each successfully added or deleted ACL entry. The existing ACL_INI entries are not consolidated to reflect the new ACL entries. To achieve that, a get_info_users or get_all_info command has to be executed, which deletes the DONE_INI entries and reflects the currently valid ACL_INIs.

If no ACL_INI attributes are defined, it is assumed that the cell uses only default values. The command then only checks whether the home directory is in DFS and whether it is accessible. If both are true, the state is set to DFS_ENABLED; otherwise the state remains unchanged.

If acl_edit fails for any reason, a message is issued, and the state is not changed.

Note: This is primarily meant for the users' home directories, but actually, any DFS object (file or directory) for which the user is owner could be specified. However, the GET_ACL_INI procedure only collects information from the home directory. So, ACL_INIs specified for other DFS objects for the dfs_enable_users command would never be updated or consolidated. If you wanted to extend this concept to other DFS objects, you would need another UDF attribute such as *dfs_objects* and extend the GET_ACL_INI procedure.

7.5.6.4 Implementation Specifics

The dfs_enable_users script is a link to the umgt script, because the checks to be performed are the same for all user management scripts.

This step is separated from rgy_enable_users to allow for creation of *all* DCE users before setting any ACLs. If you assign each user his own fileset, you must create these filesets before you run this command. After this command, you restore all files so that the initial ACLs take effect when the files are restored.

This procedure was created with tasks like migration from an NFS environment or splitting a cell in mind.

The kernel of the DFS_ENABLE_USER command is the following acl_edit command:

```
for acl_entry in $ADD_ACLs $DEL_ACLs
do
   object=`echo $acl_entry | cut -f2 -d= | cut -f2 -d#`
   perm=`echo $acl_entry | cut -f3 -d#`
   acltype=`echo $acl_entry | cut -f1 -d#`

   case $acltype in
           ADD_ACL_INI=dfs)
                parms=" $object -m $perm"
                ;;
           ADD_ACL_INI_OC=dfs)
                parms=" $object -io -m $perm"
                ;;
           ADD_ACL_INI_CC=dfs)
                parms=" $object -ic -m $perm"
                ;;
           DEL_ACL_INI=dfs)
                parms=" $object -d $perm"
                ;;
           DEL_ACL_INI_OC=dfs)
                parms=" $object -io -d $perm"
                ;;
           DEL_ACL_INI_CC=dfs)
                parms=" $object -ic -d $perm"
                ;;
           *)
                parms=" -wrong"
                BAD_ACLs=$BAD_ACLs"$acl_entry "
                continue
   esac

   acl_edit $parms
done
```

7.5.7 Enabling the ACLs in CDS and DFS: acl_enable_users

The acl_enable_users procedure applies user ACL entries of the type user:sal::rwx--- to DFS or CDS objects that do not belong to that user.

This procedure can also be used to manage these types of ACL entries. To extract all such ACLs defined for a certain user, call get_info_users. Then suspend the user, and add instructions to either remove (DEL_ACL) or add new ACLs (ADD_ACL) to the user's UDF. After that, you must call rgy_enable_users, dfs_enable_users and acl_enable_users.

7.5.7.1 Syntax

The command takes one of the following forms:

```
acl_enable_users <username>
acl_enable_users <filename>
acl_enable_users all
acl_enable_users -h
```

7.5.7.2 Arguments

username	Single user name to be processed, such as joe or austin%joe
filename	File containing only user names
all	Keyword that indicates that the central repository should be searched for all users in state DFS_ENABLED
-h	Displays more information on the purpose of the program

7.5.7.3 Description

The command requires exactly one argument. It follows the same preparation steps as rgy_enable_users except for the state, which has to be DFS_ENABLED.

Once a list of users to operate on is created, ACL_ENABLE_USER $repos $ulist is called, where $ulist is the list of candidates to be enabled. This procedure gets all ADD_ACL and DEL_ACL attributes from the user definition file and tries to apply the values to the specified DFS or CDS objects.

The ones that cannot be applied remain in the UDF for the next trial or for manual deletion by the administrator. If they can be applied, a DONE entry is created in the UDF for each successfully added or deleted ACL entry. The existing ACL entries are not consolidated to reflect the new ACL entries. To achieve that, a get_info_users or get_all_info command has to be executed, which deletes the DONE entries and reflects the currently valid ACLs.

The state is set to FULL_ENABLED if the command is successful.

If acl_edit fails for any reason, a message is issued, and the state is not changed.

7.5.7.4 Implementation Specifics

The acl_enable_users script is a link to the umgt script, because the checks to be performed are the same for all user management scripts.

This step is separated from dfs_enable_users to allow for creation or restoration of all DFS directories and files between setting the initial ACLs and applying the user type ACLs. Before you can apply the user type ACLs, all files or objects must be there, but you probably want to restore the files after you have set the initial ACLs so they take effect upon restoration.

This procedure was created with tasks like migration from an NFS environment or splitting a cell in mind.

The kernel of the ACL_ENABLE_USER command is the following acl_edit command:

```
for acl_entry in $ADD_ACLs $DEL_ACLs
do
   object=`echo $acl_entry | cut -f2 -d= | cut -f2 -d#`
   perm=`echo $acl_entry | cut -f3 -d#`
   acltype=`echo $acl_entry | cut -f1 -d#`

   case $acltype in
           ADD_ACL=dfs)
                parms=" $object -m $perm"
                ;;
           ADD_ACL=cds)
                parms=" -e $object -m $perm"
```

```
                ;;
        DEL_ACL=dfs)
                parms=" $object -d $perm"
                ;;
        DEL_ACL=cds)
                parms=" -e $object -d $perm"
                ;;
        *)
                parms=" -wrong"
                BAD_ACLs=$BAD_ACLs"$acl_entry "
                continue
   esac

   acl_edit $parms
done
```

7.5.8 Suspending Users: susp_users

The susp_users procedure invalidates the DCE account in the DCE registry (NOT valid) so the users can no longer log in. Then it sets the state of the UDF to SUSPENDED. This is a safe state for either deleting the user or changing some attributes and re-enabling them again.

7.5.8.1 Syntax

The command takes one of the following forms:

```
susp_users <username>
susp_users <filename>
susp_users all
susp_users -h
```

7.5.8.2 Arguments

username	Single user name to be suspended, such as joe or austin%joe
filename	File containing only user names
all	Keyword that indicates that the central repository should be searched for all users in state *_ENABLED
-h	Displays more information on the purpose of the program

7.5.8.3 Description

The whole description is in the introductory remarks.

7.5.9 Deleting Users: del_users

The del_users procedure is used to delete a user from the DCE registry. All user type ACLs of the form user:sal::rwx--- are removed from the objects. The UDF is moved to a cemetery directory from which it can be used to define the user with the same characteristics in another cell.

7.5.9.1 Syntax

The command takes one of the following forms:

```
del_users <username>
del_users <filename>
del_users all
del_users -h
```

7.5.9.2 Arguments

`username` Single user name to be deleted, such as joe or austin%joe

`filename` File containing only user names

`all` Keyword that indicates that the central repository should be searched for all users in state SUSPENDED

`-h` Displays more information on the purpose of the program

7.5.9.3 Description

Before the user can be deleted, all user type ACLs for that user should be removed; otherwise the ACL entry will be orphaned. This means the username that was defined for this ACL entry is replaced by its UUID, which is somewhat ugly. First, it is difficult to figure out who the user was, and second, the entry cannot be deleted before a new user adopts the UUID in the registry.

To remove all ACLs, run `get_info_users` to find all these entries first. This procedure then operates on all user-type ACLs defined in the UDF of the form:

```
ACL=cds#/.:/sec#user:usr2:r----
ACL=dfs#/:/dev/dce#user:usr2:rw----
```

and removes the according entry from the object ACL.

If ACL entries have been created or deleted recently, the UDF contains DONE attributes that reflect the update history for ACL entries managed via this UDF. This also means that the ACL attributes in the UDF do not reflect the current state as defined in DCE. To achieve a consolidation, you must either run `get_info_users` or `get_all_info`.

As long as DONE attributes are in the file or if deletion of an ACL entry fails, the user is not deleted, and an error message is displayed.

7.5.9.4 Implementation Specifics

The ACL removal part is implemented in the same way as in the `acl_enable_users` command.

7.5.10 Getting Information for Users from DCE: get_info_users

The `get_info_users` procedure collects information from the DCE registry, CDS, and DFS to update or create all attributes of the users' UDFs. After execution of this command, the upper part of the UDFs reflect exactly what is defined in the cell.

Use this command before you delete a user so that all ACLs defined for that user are found and can be deleted. See also 7.5.9, "Deleting Users: del_users" on page 298, for a description of this issue.

7.5.10.1 Syntax

The command takes one of the following forms:

```
get_info_users <username>
get_info_users <filename>
get_info_users all
get_info_users -h
```

7.5.10.2 Arguments

`username` Single user name to be processed, such as joe or austin%joe

`filename` File containing only user names

`all` Keyword that indicates that the central repository should be searched for all users defined in the repository. You can also use `get_all_info` instead, which searches for all users and groups as defined in the DCE registry and creates new UDFs or GDFs as necessary.

`-h` Displays more information on the purpose of the program

7.5.10.3 Description

The whole functional description is in the introductory remarks.

If a user name specified as <username> or contained in the file <filename> does not have a UDF, one will be created.

You can limit the search for ACLs to a specific subtree of DFS by specifying the environment variable in the file ENVIRONMENT:

```
DFS_STARTDIR=/:/dfshome
```

The default is /:/*. It scans through the ACLs of *all* DFS files and leaves out explicit read/write mount points like /:/.rw or /:/.usrbin, for instance.

7.5.10.4 Implementation Specifics

This routine is able to collect all ACLs for groups, too. You can define an environment variable with certain group names for which ACLs need to be extracted:

```
export GROUPFILES="none staff group2"
```

The GDFs of these groups are then updated or created in the same run.

7.5.11 Getting Information for All Users from DCE: get_all_info

The `get_all_info` procedure does basically the same thing `get_info_users all` does.

The only difference is: It will query all user names and group names from the DCE registry and collect information for all of them.

7.5.11.1 Syntax

The command takes one of the following forms:

```
get_all_info <userdir> <groupdir>
get_all_info -h
```

7.5.11.2 Arguments

`userdir` Repository (directory) for user files

`groupdir` Repository (directory) for group files

`-h` Displays more information on the purpose of the program

7.5.11.3 Description

You may specify new directories to create new UDFs and GDFs. If you specify the central repository name for users and groups, then existing UDFs and GDFs will be updated. For users and groups that do not have a file, one is created.

This routine extracts all information from the registry to update or create user and group definition files (UDFs and GDFs). It then gathers ACL information for each user's home directory. Finally, it scans all CDS and DFS objects to get their user and group type ACL entries and adds them to each user's or group's definition file.

This procedure can be used to perform an integrity check. Compare the UDFs/GDFs generated in new directories, which reflect the state as actually defined in DCE, with the UDFs/GDFs of the central repository, which might have been corrupted through inadequate editing. However, integrity is automatically achieved also by overwriting the existing files. The only difference is that you do not know afterwards what was inconsistent.

You can limit the search for ACLs to a specific subtree of DFS by specifying the environment variable in the file ENVIRONMENT:

```
DFS_STARTDIR=/:/dfshome
```

The default is /:/*. It scans through the ACLs of *all* DFS files and leaves out explicit read/write mount points like /:/.rw or /:/.usrbin, for instance.

Appendix A. Installing the Tools

Along with this book, we deliver a diskette with several tools we developed during this project. Install these tools in any separate directory, for instance in /dce_tools:

```
# cd dce_tools
# tar -xvf/dev/fd0
x cleanif, 9746 bytes, 20 media blocks.
x cleanup_cache, 1313 bytes, 3 media blocks.
x cleanup_cds_cache, 1317 bytes, 3 media blocks.
x cleanup_ip, 2448 bytes, 5 media blocks.
x create_cds_entry, 662 bytes, 2 media blocks.
x copy_ch, 8519 bytes, 17 media blocks.
x list_ch, 2420 bytes, 5 media blocks.
x new_cell_adm, 1507 bytes, 3 media blocks.
x show_cds, 6179 bytes, 13 media blocks.
x show_hosts, 6181 bytes, 13 media blocks.
x batch
x batch/Makefile, 165 bytes, 1 media blocks.
x batch/README, 3583 bytes, 7 media blocks.
x batch/start_batch.c, 5042 bytes, 10 media blocks.
x start_batch, 9901 bytes, 20 media blocks.
x ugmgt.tar, 163840 bytes, 320 media blocks.
```

250KB of free space is needed in /dce_tools. You may want to copy the shell scripts into /usr/local/bin or in any other directory available in your PATH environment variable.

The ugmgt.tar file contains tools and configuration files needed for user management. The concepts and structure of these tools as well as the details about the commands are described in 7.5, “Mass User/Group (and ACL) Management” on page 271.

We suggest creating a directory /ugmgt on a separate file system, which is to hold the user information database. The size of the user and group definition files (UDF/GDF) is about 1KB per user or group. Since 4KB disk space is allocated for even a one-byte file, we have to reserve 4KB per user and per group plus 200KB to hold the shell scripts.

Install the tools as follows:

```
# cd /umgt
# tar -xvf/dce_tools/umgt.tar
x ACL_ENABLE_GROUP, 4605 bytes, 9 media blocks.
x ACL_ENABLE_USER, 5095 bytes, 10 media blocks.
x ADD_GROUP, 4510 bytes, 9 media blocks.
x ADD_USER, 5400 bytes, 11 media blocks.
x CR_EMPTY_GDF, 163 bytes, 1 media blocks.
x CR_EMPTY_UDF, 158 bytes, 1 media blocks.
x DEL_GROUP, 7125 bytes, 14 media blocks.
x DEL_USER, 6245 bytes, 13 media blocks.
x DFS_ENABLE_USER, 6374 bytes, 13 media blocks.
x ENVIRONMENT, 3148 bytes, 7 media blocks.
x GET_ACCOUNT, 5843 bytes, 12 media blocks.
x GET_ACL, 10390 bytes, 21 media blocks.
x GET_ACL_INI, 6056 bytes, 12 media blocks.
```

```
x GET_GROUP, 5372 bytes, 11 media blocks.
x GET_PRINCIPAL, 5597 bytes, 11 media blocks.
x LOG, 113 bytes, 1 media blocks.
x READ_GDF, 3772 bytes, 8 media blocks.
x READ_UDF, 5801 bytes, 12 media blocks.
x RGY_ENABLE_USER, 7999 bytes, 16 media blocks.
x SUSP_USER, 3522 bytes, 7 media blocks.
x WRITE_GDF, 3009 bytes, 6 media blocks.
x WRITE_UDF, 3651 bytes, 8 media blocks.
x acl_enable_users is a symbolic link to umgt.
x add_groups is a symbolic link to gmgt.
x add_users is a symbolic link to umgt.
x del_groups is a symbolic link to gmgt.
x del_users is a symbolic link to umgt.
x dfs_enable_users is a symbolic link to umgt.
x get_all_info, 2596 bytes, 6 media blocks.
x get_info_groups is a symbolic link to gmgt.
x get_info_users is a symbolic link to umgt.
x umgt, 9775 bytes, 20 media blocks.
x gmgt, 8681 bytes, 17 media blocks.
x rgy_enable_users is a symbolic link to umgt.
x susp_users is a symbolic link to umgt.
x update_groups is a symbolic link to gmgt.
x nis2dce_groups, 742 bytes, 2 media blocks.
x nis2dce_users, 1337 bytes, 3 media blocks.
x grp2dce, 754 bytes, 2 media blocks.
x pwd2dce, 1346 bytes, 3 media blocks.
```

The user management tools have to be executed in the directory, into which they are restored. Make sure the PATH environment variable contains the current directory. Otherwise add the following command into your /etc/environment file:

```
PATH=$PATH::
```

Appendix B. Special Notices

This publication is intended to help customers, system engineers, and, to a certain extent, marketing representatives understand and find solutions for planning, configuration, and administration issues in a DCE and DFS environment. It is mainly focused on AIX DCE and DFS Release 2.1, as well as DCE 2.1 for OS/2 Warp in a DCE/DFS client role. It also shows how DOS Windows workstations or platforms without DCE are integrated. The information in this publication is not intended as the specification of any programming interfaces that are provided by AIX 4.1, OS/2 Warp, DOS Windows, IBM's DCE Version 2.1 product family for AIX Version 4.1, or the IBM DCE 2.1 for OS/2 Warp Beta program. See the PUBLICATIONS section of the IBM Programming Announcement for AIX 4.1, OS/2 Warp, DOS Windows, IBM's DCE Version 2.1 product family for AIX Version 4.1, or the IBM DCE 2.1 for OS/2 Warp for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

ADSTAR	Advanced Function Presentation
AFP	AIX
AIX/6000	CICS
HACMP/6000	IBM
InfoExplorer	LoadLeveler
MQSeries	NetView
OS/2	OS/400
Print Services Facility	PS/2
PSF	PSF/6000
RISC System/6000	RS/6000
Trouble Ticket	

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.
PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.
UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.
Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

DG/UX	Data General Corporation
DEC, VMS, DIGITAL	Digital Equipment Corporation
Encina, Transarc	Transarc Corporation
HP, HP/UX	Hewlett-Packard Company
NetWare, Novell	Novell, Inc.
Network File System, NFS, NIS, ONC, Solaris, Sun	Sun Microsystems, Inc.
OEC	Open Environment Corporation
OSF, Motif, Open Software Foundation, OSF/1	The Open Software Foundation
POSIX	Institute of Electrical and Electronic Engineers
SCO	The Santa Cruz Operation, Inc.
Siemens, Siemens-Nixdorf, Sinix	Siemens Aktiengesellschaft

Other trademarks are trademarks of their respective companies.

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

C.1 International Technical Support Organization Publications

- *IBM DCE Cross-Platform Guide*, GG24-2543
- *Understanding OSF DCE 1.1 for AIX and OS/2*, SG24-4616
- *The Distributed File System (DFS) for AIX/6000*, GG24-4255
- *Using and Administering AIX DCE 1.3*, GG24-4348
- *Developing DCE Applications for AIX, OS/2 and Windows*, GG24-4090

A complete list of International Technical Support Organization publications, known as redbooks, with a brief description of each, may be found in:

International Technical Support Organization Bibliography of Redbooks, GG24-3070.

C.2 Other Publications

These publications are also relevant as further information sources:

General DCE Books

- *Networking Blueprint, Executive Overview*, GC31-7057
- *Understanding DCE Concepts*, GC09-1478
- *OSF DCE User's Guide and Reference (Prentice Hall)*, SR28-4992
- *OSF DCE Administration Reference (Prentice Hall)*, SR28-4993
- *OSF DCE Application Development Reference (Prentice Hall)*, SR28-4995
- *OSF DCE Application Development Reference (Prentice Hall)*, SR28-4995
- *Understanding DCE (O'Reilly & Associates)*, SR28-4855

DCE Version 1.3 for AIX

- *DCE V1.3 for AIX Release Notes*, GC23-2434
- *DCE V1.3 for AIX User's Guide and Reference*, SC23-2729
- *DCE V1.3 for AIX Administration Guide -- Core Services*, SC23-2730
- *DCE V1.3 for AIX Administration Guide -- Extended Services*, SC23-2731
- *DCE V1.3 for AIX Administration Reference*, SC23-2732
- *DCE V1.3 for AIX Application Development Guide*, SC23-2733
- *DCE V1.3 for AIX Application Development Reference*, SC23-2734
- *DCE NFS to DFS Authenticating Gateway V1.3 for AIX*, SC23-2735
- *NetView for DCE and Encina Manager Guide V1.3*, SC23-2736
- *AIX HACMP for DCE and Encina Guide V1.3*, SC23-2737

- *AIX DCE Getting Started V1.3*, SC23-2477
- *AIX DCE and OS/2 DCE Message Reference*, SC23-2583

DCE Version 2.1 for AIX

- *Introduction to DCE V2.1 for AIX*, SC23-2796
- *DCE V2.1 for AIX: Getting Started*, SC23-2797

These are the only printed manuals. The documentation basically comes with the program components in softcopy form only. These manuals can be printed from within the ASCII viewer in ASCII format or from within the IPF/X graphical softcopy browser in PostScript format. The following softcopy books are available:

- *Introduction to DCE*
- *DCE for AIX Getting Started*
- *DCE for AIX Administration Guide*
- *DCE for AIX Administration Command Reference*
- *DCE for AIX Application Development Guide - Introduction*
- *DCE for AIX Application Development Guide - Core Services*
- *DCE for AIX Application Development Guide - Directory Services*
- *DCE for AIX Application Development Reference*
- *DCE for AIX DFS Administration Guide and Reference*
- *DCE for AIX NFS/DFS Authenticating Gateway Guide and Reference*

C.3 DCE Information on the World Wide Web (WWW)

A lot of information on DCE is available via the Web. To access it, start with one of the following URLs:

```
http://www.raleigh.ibm.com/dce/dcehome.html
http://www.osf.org/dce/index.html
```

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com/redbooks.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  http://w3.itso.ibm.com/redbooks/redbooks.html

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **ITSO4USA category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	bookshop at dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29554	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada (toll free)	1-800-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 415 855 43 29 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com/redbooks
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

- **Please put me on the mailing list for updated versions of the IBM Redbook Catalog.**

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number _____
- Credit card number _____

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

List of Abbreviations

ACL Access Control List

ADSM ADSTAR Distributed Storage Manager

AFS Andrew File System

AS Authentication Service

ATM Asynchronous Transfer Mode

BDM Binary Distribution Machine

BOS Base Operating System, Basic Overseer (server in DFS)

C/S Client/Server

CDE Common Desktop Environment

CDMF Common Data Masking Facility

CDS Cell Directory Service

CICS Customer Information Control System

CM Cache Manager

CMA Concert Multithread Architecture

CMIP Common Management Interface Protocol

CMVC Configuration Management and Version Control

COSE Common Open Software Environment

CUT or UTC Universal Time Coordinated

DCE Distributed Computing Environment

DAP Directory Access Protocol

DES Data Encryption Standard

DFS Distributed File System

DME Distributed Management Environment

DNS Domain Name Service

DoD Department of Defense

DSOM Distributed System Object Model

DSS Directory and Security Service

EMS Event Management Service

EPAC Extended Privilege Attribute Certificate

EP Endpoint

ERA Extended Registry Attributes

FCS Fibre Channel Standard

FLDB Fileset Location Database

GDA Global Directory Agent

GDF Group Definition File

GDS Global Directory Service

GSS-API Generic Security Services Application Programming Interface

HACMP High Availability Cluster Multi-Processing

HTTP Hypertext Transfer Protocol

IBM International Business Machines Corporation

ICC Initial Container Creation (ACL)

IDL Interface Definition Language

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IHMP IBM NetView Hub Management Program

IOC Initial Object Creation (ACL)

IP Internet Protocol

ISO International Standardization Organization

IT Information Technology

ITSO International Technical Support Organization

JFS Journaled File System

LAN Local Area Network

LFS Local File System

LF Login Facility

LRPC Local RPC

MAN Metropolitan Area Network

MIB Management Information Base

MOCL Managed Object Class Library

MQI Message Queuing Interface

MPTN Multi-Protocol Transport Networking

MPTS Multi-Protocol Transport Service

NCACN Network Computing Architecture Connection Based Protocol

NCADG Network Computing Architecture Datagram Protocol

NFS Network File System

NIS Network Information System

NIST National Institute of Standard and Technology

NSI Name Service Interface

NTP Network Time Protocol

OLTP On-Line Transaction Processing

OMG Object Management Group

ONC Open Network Computing

OSF Open Software Foundation

PAC Privilege Attribute Certificate

PM Presentation Manager

PS	Privilege Service
PTX	Performance Toolbox
RDBMS	Relational Database Management System
RPC	Remote Procedure Call
RS	Registry Service
SCM	System Control Machine
SMIT	System Management Interface Tool
SNA	System Network Architecture
SNMP	Simple Network Management Protocol
SOM	System Object Model
SQL	Structured Query Language
TCP	Transmission Control Protocol
Tcl	Tool Command Language
TDF	Time Differential Factor
TGS	Ticket Granting Service
TGT	Ticket Granting Ticket
TPI	Time Provider Interface
TPS	Transactions Per Second
TSR	Token Status Recovery
TTL	Time to Live
UDF	User Definition File
UDP	User Datagram Protocol
UID	User Identifier
UUID	Universal Unique Identifier
VFS	Virtual File System
WAN	Wide Area Network
XMP	X/Open Management Protocol

Index

D

E

F

G

H

R

S

T

U

W

X

Printed in U.S.A.