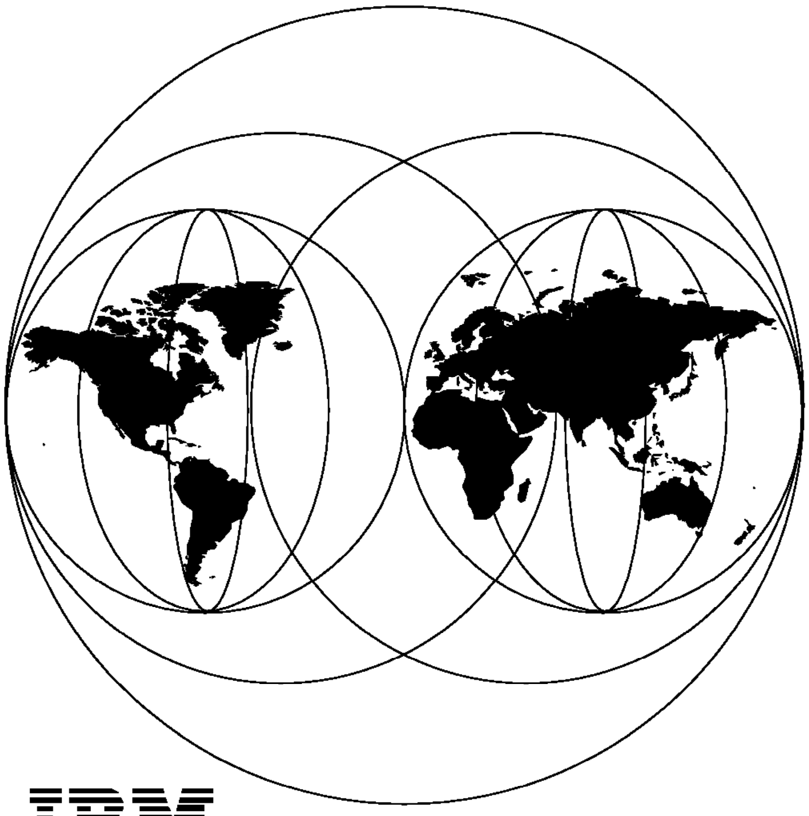
International Technical Support Organization

GG24-4348-00

Using and Administering AIX DCE 1.3

November 1994



IBM

International Technical Support Organization
Austin Center

International Technical Support Organization

GG24-4348-00

Using and Administering AIX DCE 1.3

November 1994

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Special Notices" on page xiii.

First Edition (November 1994)

This edition applies to:

- DCE Base Services for AIX 1.3, (5765-117)
- DCE Security Server for AIX 1.3, (5765-118)
- DCE Cell Directory Server for AIX 1.3, (5765-119)
- DCE Enhanced Distributed File System for AIX 1.3, (5765-121)
- DCE NFS to DFS Authenticating Gateway for AIX 1.3, (5765-457)

for use with the AIX 3.2.5 Operating System.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader′s feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 948S Building 821 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Abstract

This document describes the new features contained in the AIX Distributed Computing Environment (DCE) Version 1.3 and the AIX Distributed File Server (DFS) Version 1.3.

It provides useful instructions and tools for DCE administrators to configure cells for different network topologies, manage a large number of users, perform daily administration tasks, and reconfigure certain aspects of entire cells.

Planners, DCE administrators and system engineers will gain an idea of what components must be used in certain business environments. Starting with a discussion on business requirements, this document provides guidance on DCE component layout in a cell for optimal performance and availability.

(292 pages)

Contents

Figures

Tables

Special Notices

Preface

This document is intended to provide various levels of information on planning, using, and administering a Distributed Computing Environment. It contains a short introductory description of all DCE components and their administration tools. This should enable the reader to understand the DCE cell layout planning issues discussed thereafter. The chapters on planning and implementing DCE cells include integration of OS/2 and DOS Windows workstations. However, since their DCE release is not so advanced yet, we look more at their integration into DCE when they are using the Sun Network File System (NFS). Furthermore, we have documented our experiences in performing several configuration and administration tasks. We have created some useful tools which will facilitate administration and particularly large scale user management.

The first two chapters are intended for anyone who needs to understand the basic DCE components and related planning issues. System administrators and even 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 will give them guidance on how to lay out and implement the DCE components for different network topologies and how to perform many important administration routines.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “Introduction”

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

- Chapter 2, “Planning DCE Cells”

  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”

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

- Chapter 4, “Administering DCE Cells”

  This chapter is organized in a task-oriented format. It describes certain administration routines step-by-step. The tasks range from configuring cells and performing daily administration tasks to reconfiguring certain aspects of entire cells. Some of these tasks are supported by tools that we have developed during our project. This chapter is intended for system administrators.

- Chapter 5, “New Tools and Technologies”

  This chapter describes the new features available with AIX DCE Version 1.3, which corresponds to OSF DCE 1.0.3. It also introduces a set of shell scripts and an integrated login package for AIX and DCE, which facilitate the entire user management of a cell. The section which describes the highlights of AIX DCE 1.3 might be of interest for every reader.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *Understanding DCE Concepts*, GC09-1478
- *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
- *OSF DCE Introduction to DCE (Prentice Hall)*, SR28-4991
- *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
- *Understanding DCE (O'Reilly & Associates)*, SR28-4855
- *HACMP/6000 Licensed Program Specification*, GC23-2594
- *HACMP/6000 System Overview*, SC23-2595
- *HACMP/6000 Planning and Installation Guide*, SC23-2596
- *HACMP/6000 Application Programming Interface Guide*, SC23-2597
- *HACMP/6000 Troubleshooting Guide*, SC23-2598
- *HACMP/6000 Administration Guide*, SC23-2599

International Technical Support Organization Publications

- *OSF DCE for AIX, OS/2 and DOS Windows Overview*, GG24-4144
- *Developing DCE Applications for AIX, OS/2 and Windows*, GG24-4090
- *The Distributed File System (DFS) for AIX/6000*, GG24-4255

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

International Technical Support Organization Bibliography of Redbooks, GG24-3070.

To get listings of ITSO technical bulletins (redbooks) online, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Technical Bulletins (Redbooks)

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their IBM branch office.

You may order individual books, CD-ROM collections, or customized sets, called GBOFs, which relate to specific functions of interest to you.

Acknowledgments

The advisor for this project was:

Rolf Lendenmann
International Technical Support Organization, Austin Center

The authors of this document are:

Rolf Lendenmann
IBM Austin

Brice Muang-Khot
DCE Center of Competence, IBM France

Jacques Dubuquoy
International Education Center, La Hulpe, Belgium

Hanspeter Nagel
IBM Switzerland

Salvatore La Pietra
Security Center of Competence, IBM Germany

This publication is the result of a residency conducted at the International Technical Support Organization, Austin Center.

We would like to offer our thanks to the IBM-Austin DCE Development team for their invaluable advice and assistance in the reviewing of this document.

Chapter 1. Introduction

Distributed client/server (C/S) environments represent a new 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.

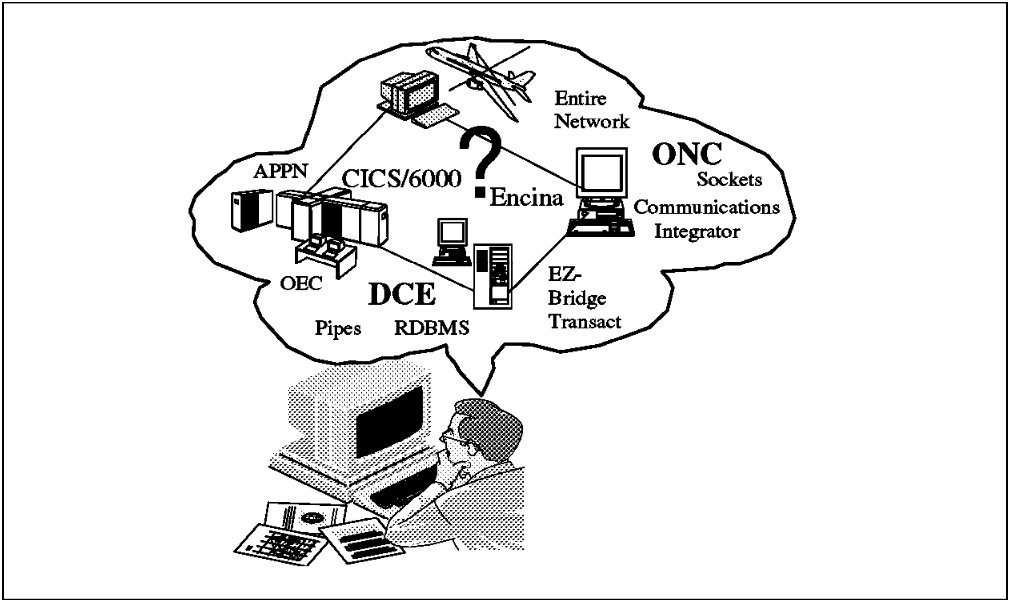


Figure 1. Options in a Distributed Environment. There are so many options to build a solution, the user has a hard time deciding what to choose.

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 related administration issues. We will focus on the major technologies available on the market namely Open Software Foundation** Distributed Computing Environment (OSF** DCE**) and Sun** Network Information System/Network File System** (NIS**/NFS**). We will also discuss IBM* AIX* High Availability Cluster Multi-Processing/6000 (HACMP/6000*).

The reader should be familiar with terms such as protocol and protocol suite, synchronous and asynchronous protocol, TCP/IP, UDP, network interface, connection-oriented and connection-less, Ethernet and Token-Ring, as well as X.25, ISDN, SLIP, LAN, WAN. Terms such as distributed environment, distributed C/S environment, distributed systems, and distributed C/S systems will be used throughout this book without any distinction.

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.

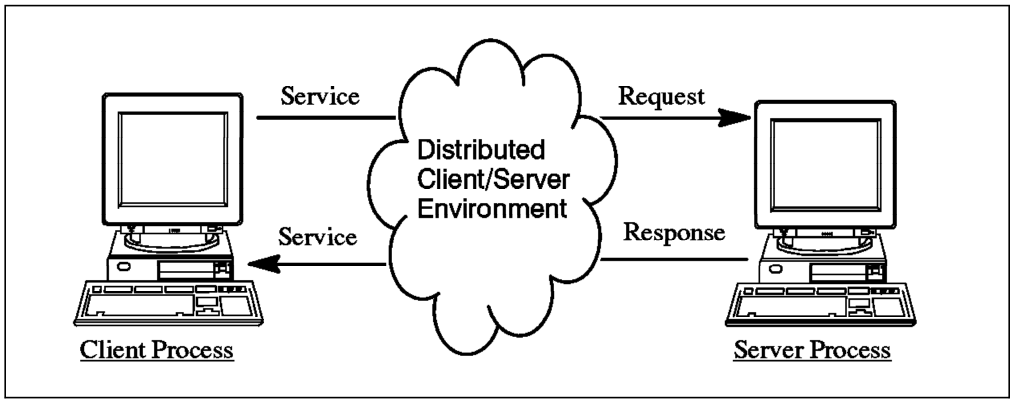


Figure 2. Client/Server Model. The simplest form of C/S computing has only two pieces: a client process and a server process connected via a network.

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 and 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 inter-operate 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, multiple servers can run on the same system and so on.

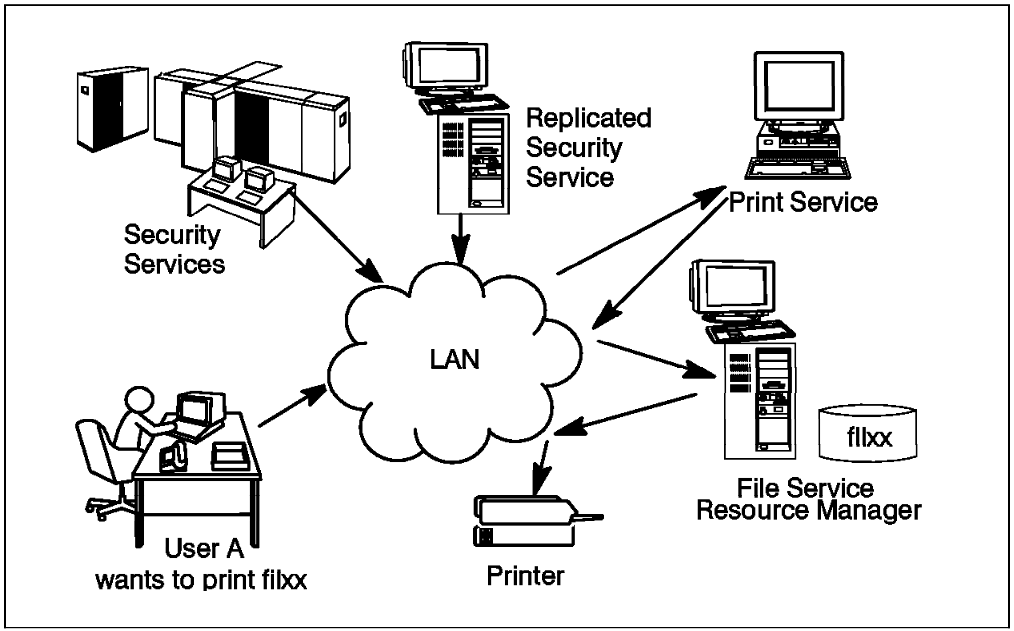

Replicated Security Service
Print Service
Security Services
LAN
filxx
File Service Resource Manager
User A wants to print filxx
Printer

Figure 3. Example of C/S Application. A print request may turn into a client request to the file server to provide a copy of the file to be printed.

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 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.

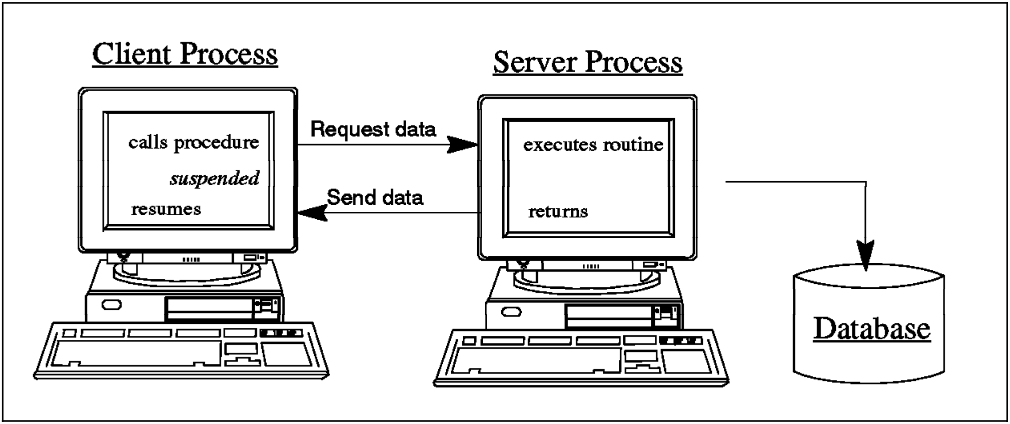


Figure 4. Two-Tier Model. A client machine accesses a server that holds data. The client machine selects, examines and processes the data. The application is on the client.

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

As of today RDBMSs are not really distributed. With trigger or snapshot mechanisms tables can be replicated to other network nodes for load balancing, but this actually only works well for read only (R/O) access. Basically there must always be a master database on a powerful server machine. To offload all other activities from the database machine the applications run on the client systems. They contain all the data processing logic, for instance prepare SQL queries, send them to the database, receive a large amount of data, and process them. This model requires good network connections and powerful client machines.

1.1.2 Three-Tier Client/Server Model

For the three-tier model, also called application-oriented, shown in Figure 5 on page 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* or the Open Environment Corporation Toolkit (OEC** Toolkit) working on top of DCE map to such a model.

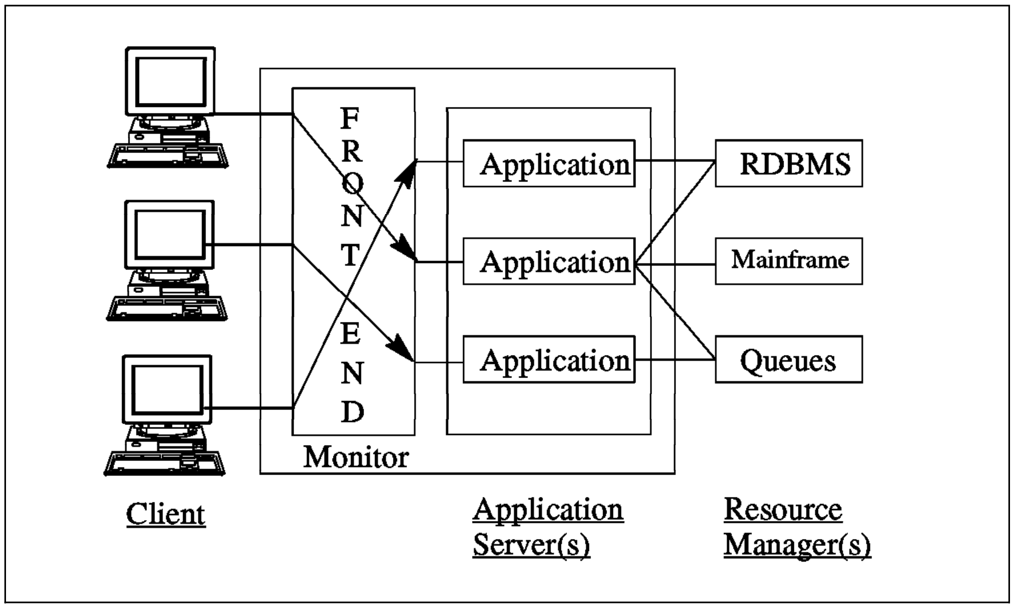


Figure 5. Three-Tier Model. Client requests are issued to a transaction monitor, which schedules the requests for execution by an application server which in turn requests data access from a data server.

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 which 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. In the next chapter we will analyze the system management aspects of C/S environments.

1.2 Administration Issues in Client/Server Environment

The theory and practice of centralized control evolved around a big mainframe and has been driving company policies, user services, hardware configurations, network management, and software development for decades.

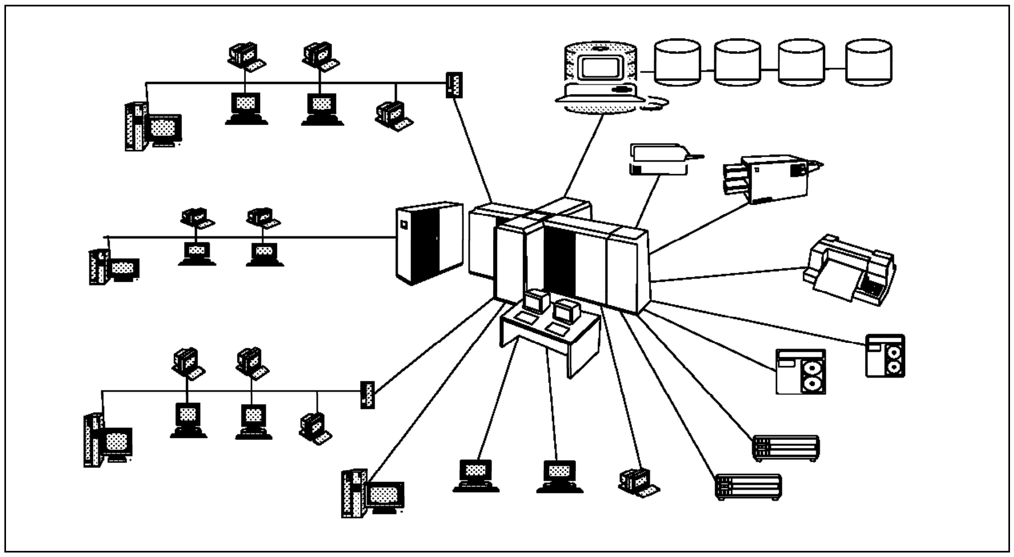


Figure 6. Centralized Administration. Has been the only reliable solution in larger environments for decades.

Usually the following rule is valid for C/S environments and centralized systems: Larger means harder to manage, understand, modify, debug and fix. As requirements of companies, departments, and even offices change, so does the management of such environments. For example R&D departments are driven by release schedules. They are very sensitive to timing and schedules. They keep a very heterogeneous environment with a different set of hardware and various versions of the same software on different platforms. The aspects of security vary depending on the project and company policy. In contrast, the Human Resource or Administration departments are a completely different beast, and so are their system management structures. Computers are viewed as black boxes. Needs for change and innovation are less than in an R&D department. However, the security concerns are much higher.

Administration is a big issue and encompasses many different activities and problem domains. More than in a centralized environment, system management in a distributed environment needs to tie all components and aspects of a distributed system together. This is not an easy task.

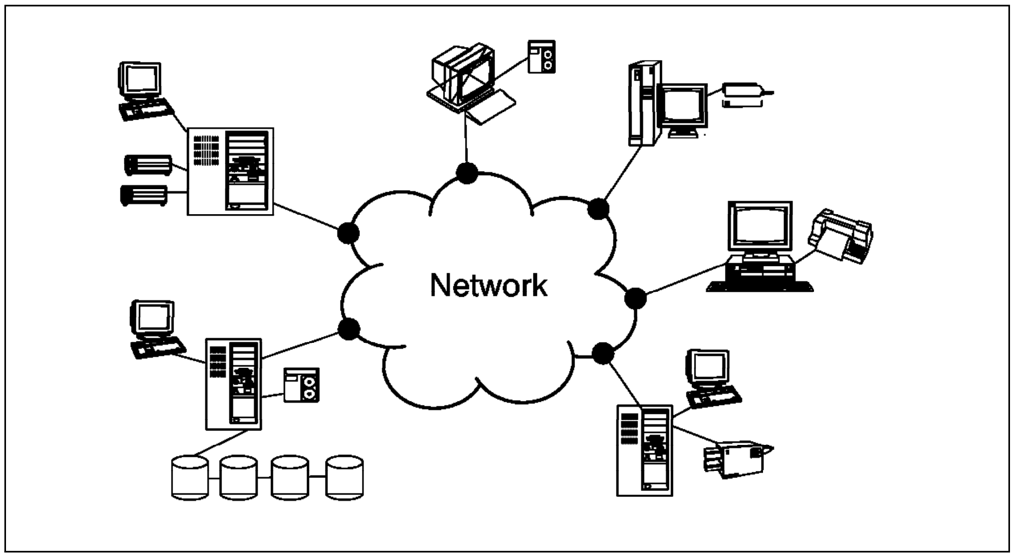


Figure 7. Distributed Administration. More than just printers and terminals are spread over a network.

Some examples of the most common problems include:

- System Management

  Addresses operation of a single computer system such as administration of users/groups, backups, file systems, and so on. In C/S environments this needs to be done for multiple systems, possibly in different locations.

- Network Management

  Addresses services and devices necessary to connect and monitor multiple computer systems. This includes bridges, routers, network interface and Internet addresses.

- Application Management

  Manages the software that is used on a system or network. This includes aspects of software installation, distribution, and configuration.

Many of these problems have been solved with different tools and applications. New technologies are coming out to address certain management aspects, such as job scheduling, tuning and performance monitoring, and error notification, but most of the work still needs to be done.

1.2.1 What is Available for Management

When we start to count the number of packages available on the market that can manage distributed environments, we find bits and pieces but not a unique framework which integrates all the management functions. The holy grail for quite some time was called OSF Distributed Management Environment (DME**). About four years ago OSF announced DME at a member′s meeting in Boston, but soon after the announcement the project had its first problem. OSF then released Distributed Services which includes:

- License Management Service
- Software Distribution Service
- Event Service
- Subsystem Management Service
- Personal Computer Service

DCE technology meets the industry needs, but DME is not on the market. OSF is facing a major reorganization, because former participants of OSF and Unix International (UI) will be brought together in a new organization, which will also consider work done in the Common Open Software Environment (COSE).

Perhaps DME is not yet dead, but by the time DME is available and vendors start to integrate it into their products other solutions may be available. So customers are turning to proprietary software solutions from companies such as IBM, HP**, SUN, DEC**, Computer Associates**, and Tivoli**, a company that has established itself as a distributed system management solutions provider. After their latest announcement of Tivoli Management Environment (TME**) they can manage many PCs in mixed UNIX** and PC environments. TME includes packages for software distribution, backup/restore, scheduling and workload management, which work in a distributed environment. Very popular among mainframe customers but less in the Open Systems arena is Computer Associates** (CA). CA is providing CA-Unicenter** that gives a set of management functions including user management, centralized backup, and security verification. However, this package is still dependent on the central system and not fully distributed.

IBM NetView* for AIX is one of the most popular network management solutions from IBM. Lately it has been adopted by DIGITAL** for their POLYCENTER** solution. IBM NetView for AIX simplifies network management in a multi-vendor transmission control protocol/internet protocol (TCP/IP) network. It provides monitoring and management of TCP/IP devices that include simple network management protocol (SNMP) agents. IBM NetView for AIX provides several application programming interfaces (APIs) to allow for integration of other management tools. These applications can run on the IBM NetView for AIX server and operate without SNMP, or they can extend the function of client systems by providing SNMP subagents. With AIX DCE 1.3 it is possible to monitor DCE services from IBM NetView for AIX in this way. Further examples of applications that can run under IBM NetView for AIX are:

- IBM Trouble Ticket* for AIX - for problem management

  This allows you to easily manage problems from initial discovery to problem closure. Diverse analysis tools and reports help circumvent problems before they occur. It allows easy storage and retrieval of problem management information, which can also be accessed from Microsoft Windows**, SunOS**, and HP/UX** clients.

- IBM Systems Monitor for AIX - for capacity management

  This can be configured to warn the network operator that critical system levels are being reached; action can then be taken before the system goes down. There is even an analysis and automatic command execution function that can be utilized to detect a danger level and execute a command to correct a potential problem before it occurs. It is also available for NCR** UNIX, Sun Solaris**, and HP/UX.

- IBM NetView Hub Management Program for AIX - for network management

  IHMP/6000 facilitates and expands the management of LANs with IBM 8260 Multiprotocol Intelligent Switching Hubs and IBM 8250 Multiprotocol Intelligent Hubs. Graphics and forms provide maximum efficiency for the everyday network control operations.

IBM NetView for AIX also supports the X/Open** management protocol (XMP) API to provide SNMP and common management interface protocol (CMIP) over

TCP/IP. For more information on IBM NetView for AIX please refer to IBM announcement letters.

Other solutions for management of networked systems offered by IBM include the following management packages:

- IBM NetView Distribution Manager for AIX - for software distribution

  NetView DM/6000, a follow-on product of SoftDist/6000, provides services for software and data distribution and change control in a network of workstations from a central RISC System/6000* machine acting as change control server (CC server). Each client workstation in a TCP/IP network must have installed a counterpart acting as change control client (CC client), which can be AIX/6000*, OS/2*, HP/UX or Windows clients.

- IBM Configuration Management Version Control - for change management

  The IBM CMVC products provide application developers with configuration management, change control, and versioning that is integrated with design and defect tracking for heterogeneous environments. CMVC servers are available for AIX, SunOS, Solaris, and HP/UX. These CMVC servers supports six different CMVC clients on AIX/6000, OS/2, SunOS, Solaris, HP/UX and DOS/Windows.

- IBM Performance Toolbox (PTX) for AIX - for performance management

  PTX consists of several performance programs packaged together. It provides a toolbox framework for performance management in a network environment. By gathering information from an SNMP network manager it can provide a fine granularity real-time view into individual network nodes and processes. The X- and Motif**-based applications provide real-time color graphic performance monitors for local and remote systems, performance analysis tools and performance tuning controls. PTX for AIX also includes the PAIDE.

- IBM Performance Aide (PAIDE) for AIX - for performance management

  This is the client function for PTX. It acts as an SNMP subagent on all client nodes, which can be AIX, SunOS, or HP/UX. It performs local data filtering and alert processing.

- ADSTAR* Data Storage Management (ADSM) - for data management

  ADSM is a client/server storage management product that provides administrator controlled, highly-automated, centrally scheduled, network-based back-up and archive functions for workstations and LAN file servers. It backs up data from clients running on OS/2, NetWare**, Windows, DOS, Macintosh**, and UNIX platforms to a backup-server running on AIX/6000, OS/2, VM or MVS.

- UniTree** for AIX/6000 - for data management

  This vendor product is a continuous, non-intrusive, multi-level, transparent file and data storage management product. It migrates infrequently used files from expensive disk storage to lower-cost storage while maintaining frequently used files online and ready for use.

- Legato NetWorker** for RISC System/6000 - for data management

  This vendor product, is a powerful, easy-to-use product designed to backup data across an entire network of computer systems. It also automates and simplifies file recovery, so users never waste time recreating valuable work.

- LoadLeveler* for AIX - for workload management

  It provides a facility for building, submitting, and processing jobs quickly and efficiently in a dynamic environment. LoadLeveler can manage serial batch, parallel batch, and interactive sessions seamlessly across all participating nodes. It is also available for HP/UX, SunOS, Solaris, and IRIX**.

- Network License System** (NetLS**) - for resource management

  NetLS is an enabling software package architected specifically for computing environments and allows users to dynamically allocate software resources over a network. It offers benefits to both software developers, by ensuring their software products are used under properly licensed conditions, and to software users, by ensuring they get maximum utilization of their software assets.

- IBM Distributed SMIT for AIX (DSMIT) - for configuration management

  Provides the functionality of the standard AIX System Management Interface Tool (SMIT) in a homogeneous or heterogeneous distributed systems environment. DSMIT allows flexibility in managing clusters of resources, performing most tasks concurrently or sequentially. It is also available for SunOS and HP/UX.

- IBM Print Services Facility* for AIX (PSF/6000*) - for print management

  PSF* delivers IBM Advanced Function Presentation* (AFP*) capabilities to the RISC System/6000 platform. It provides printing solutions for stand-alone environments, local area network (LAN) environments, distributed print environments (via TCP/IP) and Network File System (NFS) protocols. It also provides printer sharing between LAN systems and host systems.

Some of these solutions are distributed, some are not. You can refer to product related documentation to read more about a specific product.

1.3 Distributed Technology

In the following sections we will describe the technologies which are the subject of our administration experience, namely:

- Open Software Foundation Distributed Computing Environment (OSF DCE)
- Sun Network Information System/Network File System (NIS/NFS)
- IBM High Availability Cluster Multi Processing/6000 (HACMP/6000)

1.3.1 DCE Overview

DCE is establishing a de-facto standard in the client/server arena with thousands of site installations and hundreds already in production. DCE is supported or at least announced on major 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, DEC** VMS**, Siemens-Nixdorf** BS/2000, HP MPE/ix, and on all the UNIX flavors including: AIX, OSF/1**, Solaris, HP/UX, DG/UX**, Sinix**, SCO**. All this has evolved in less than two years from the first release of OSF DCE 1.0. Not even Sun NIS/NFS can claim such broad support in such a short time. After the announcement of the merge between UNIX International and OSF members, it has become clear that DCE is going to be adopted in the NIS/NFS community, too. In this documentation we want to help SEs, customers, and marketing representatives to:

- Plan for DCE

- Understand what costs are involved for a particular DCE configuration
- Understand what configuration can map best 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 will try to answer all these questions, and give a good and fair DCE perspective.

1.3.2 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 inter-operate with other environments.

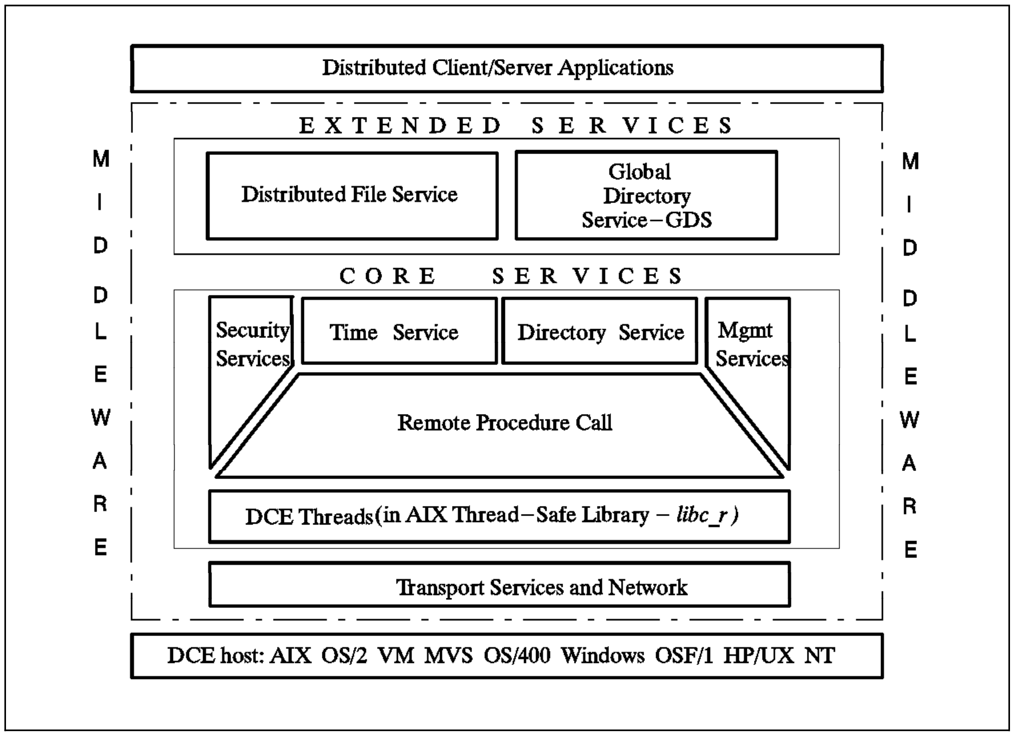

Distributed Client/Server Applications
EXTENDED SERVICES
Distributed File Service
Global Directory Service−GDS
CORE SERVICES
Security Services
Time Service
Directory Service
Mgmt Services
Remote Procedure Call
DCE Threads(in AIX Thread−Safe Library − libc_r)
Transport Services and Network
DCE host: AIX OS/2 VM MVS OS/400 Windows OSF/1 HP/UX NT
MIDDLEWARE
MIDDLEWARE

Figure 8. DCE as Middleware. The operating system is completely hidden by a set of core and extended services offered by DCE.

If we consider DCE as *Middleware* (as most of the 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 see only 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. So the architecture viewing DCE as middleware is explained in the following sections.

1.3.3 OSF DCE Components and Services

The following sections provide a short description of the DCE components and technology. At the end of each section you will be referred to other documentation that contains a full description. In 1.3.3.12, "DCE Packaging and Cost" on page 21 you will find product names and numbers in AIX, OS/2 and DOS/Windows that contain the respective component.

1.3.3.1 DCE Threads

Threads support the creation, management, and synchronization of multiple concurrent execution paths within a single process. The threads API is based on POSIX** 1003.4 Draft 4. This component can map its calls directly to operating system threads, if available. AIX Version 3 does not support threads in its kernel. AIX Version 4.1 has kernel threads, which will be used by DCE, once DCE is available on AIX 4.1.

The DCE core services and all depending applications use threads. This all happens under the covers. Customer applications may or may not use threads for their own purpose. However, application developers must know they *are* using threads anyway through the DCE RPC runtime services.

Since the present threads implementation is running in user mode rather than in kernel mode and the kernel does not know threads, 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. To avoid blocking the process with a thread, they should either use only calls of thread safe libraries, use asynchronous I/O calls, or write their applications in a way that one server is only talking to one client at a time and vice versa.

For a complete overview on the threads facility refer to Chapter 3 in the ITSO Austin publication *OSF DCE for AIX, OS/2 and DOS Windows Overview* and DCE related documentation listed in "Related Publications" on page xvi.

1.3.3.2 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 to build 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
- Runtime service implements the network protocol and communication between the client and server applications

Using 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 refer to Chapter 4 in the ITSO Austin publication *OSF DCE for AIX, OS/2 and DOS Windows Overview* and DCE related documentation listed in "Related Publications" on page xvi.

1.3.3.3 DCE Security Service

Distributed computing encourages a free flow of data between nodes, expanding the capabilities of interconnectivity and interoperability. Security breaches might come from any component of the distributed system. Security is one reason why customers are interested in DCE.

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

For such reasons security is a critical component in a distributed computing environment. A big concern is authentication of clients and servers. DCE solves this using Kerberos. Some customers ask why they should pay thousands of dollars for DCE when they can get Kerberos free from MIT. Not only is DCE an authentication framework like Kerberos but also a complete security framework, with an architecture that enforces a discretionary security policy through the use of Access Control Lists (ACLs).

Introducing Kerberos in your organization and creating Kerberos protected programs such as file transfer (ftp) or remote virtual terminal access (telnet) will not secure the company′s information assets. To protect company′s information assets a company must have a security policy in place, a security architecture that enforces the company′s security policy and finally a technology that implements such an architecture through the use of different security services and mechanisms. Where technology cannot implement such mechanisms, a set of rules and procedures must be in place to specify what needs to be done.

DCE embeds authentication and privacy into the RPC communication facility providing a powerful security framework to developers of DCE applications.

If you were using only Kerberos without DCE, you would have to take care of the transport layer yourself, using sockets, streams or whatever, and decide, if you want authentication to happen just once, or for each message. Also you would have to decide, whether you need privacy for your data, and if so use the session key to encrypt data. You would accomplish this using data encryption standard (DES) and would have to deal with all the related export licence issues outside the US.

In DCE your application needs one call on the server side and one call on the client side to establish the level of authentication and privacy your application needs. For data encryption you can either use DES or the new Common Data Masking Facility (CDMF) which uses a 40-bit encryption key in contrast to the full 52-bit DES key. CDMF is allowed for export and still provides strong confidentiality.

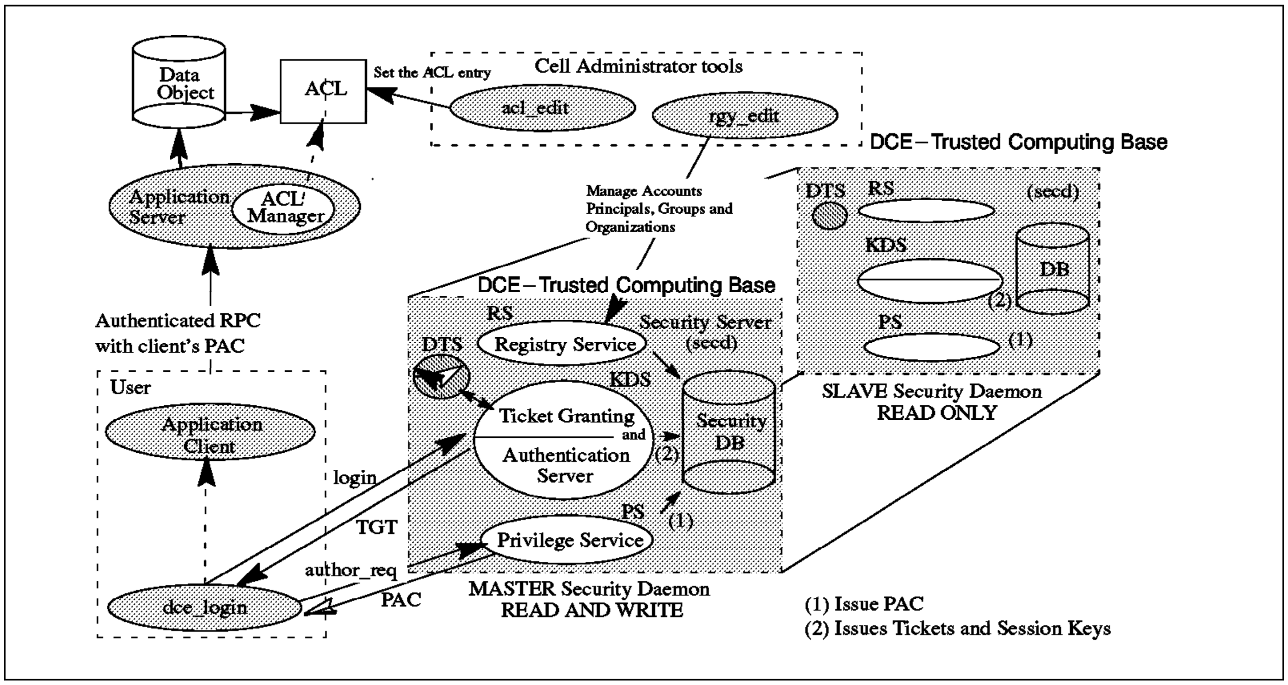

Data Object
ACL
Set the ACL entry
Cell Administrator tools
acl_edit
rgy_edit
DCE−Trusted Computing Base
Application Server
ACL Manager
Manage Accounts Principals, Groups and Organizations
DTS
RS
(secd)
KDS
DB
(2)
PS
(1)
DCE−Trusted Computing Base
RS
Registry Service
Security Server (secd)
Authenticated RPC with client's PAC
DTS
KDS
User
Application Client
Ticket Granting and Authentication Server
Security DB
(2)
SLAVE Security Daemon READ ONLY
login
TGT
PS
(1)
Privilege Service
author_req
PAC
dce_login
MASTER Security Daemon READ AND WRITE
(1) Issue PAC
(2) Issues Tickets and Session Keys

Figure 9. DCE Security Architecture

DCE Security Service consists of:

- DCE Authentication Service: also called Key Distribution Center or Ticket Granting Server provides clients with tickets and session keys
- Privilege Service: provides Privilege Attribute Certificates (PACs) used by ACL Managers to determine the access permissions to services and/or data
- DCE User Registry: manages accounts, principals, groups, organizations, aliases, services, policies and properties with the help of the `rgy_edit` administrative command
- DCE Login Facility: initial authentication is accomplished with the `dce_login` command
- Access Control List (ACL): goes with each object and protects it by listing authorized principals and groups together with their access rights; ACLs are set using `acl_edit` and checked by service ACL managers on access requests of clients
- Privilege Attribute Certificates (PACs): certifies which groups a principal is a member of, so that ACL Managers can check the permissions principals might get from group entries in ACLs
- Authenticated RPC: provides different levels of authentication for API calls, which range from none to a level where for each message exchange between client and server an authentication is required
- Secure RPC: provides different levels of integrity and privacy for API calls, which range from none to a level where each message between client and server is encrypted using the session key and DES or CDMF
- Security Replication: for high availability the security master copy is replicated and propagated to the security slave copy; the replicated security slave server can accomplish read-only operations, such as issuing tickets, but all the write-access operations will hang

Note

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

A complete set of DCE Security APIs is offered to write trusted distributed applications. For a complete overview on DCE Security refer to Chapter 5 in the ITSO Austin publication *OSF DCE for AIX, OS/2 and DOS Windows Overview* and DCE related documentation listed in “Related Publications” on page xvi.

1.3.3.4 DCE Directory Service

DCE Naming Service provides a naming model throughout the distributed environment. This model allows users to identify by name resources such as servers, files, disks, or print queues, and gain access to them without needing to know where they are located in a network. Further, users can continue referring to a resource by the same name even when a characteristic of the resource changes, such as its network address.

The global distributed computing environment is composed of administratively independent cells. The name space is hierarchically organized and forms a tree, with containers (directories) and leaf objects. The root directory /... is global and contains all cell names, which build the root directories for each cell. The subtrees underneath each /.../<cellname> directory are under the management domain of their respective cell. Users within a cell can use the short form /.: to address their local root directory. CDS then replaces the name /.: with the appropriate global name /.../<cellname>. The leaf objects can be any resource as mentioned above. The main purpose however, is to store and retrieve binding information for DCE RPC servers, which can be used by DCE clients to find and bind to a server.

The naming system consists of the following components:

- Cell Directory Service (CDS)
- Global Directory Service (GDS) X.500 (separate product)
- Global Directory Agent (GDA)
- Application Programming Interface

The local naming system is provided by the CDS which can be distributed and replicated. It is integrated with a global naming system X.500 or DNS (Domain Name System) via a Global Directory Agent. Communication between cells is done via the Global Directory Agent (GDA) where the global name space is the naming bridge. The global name space here can be X.500 or DNS.

Objects from the global name space are available from within a cell via the Global Directory Agent (GDA) function which translates CDS internal protocol to OSI DAP (Directory Access) protocol. The GDA supports worldwide DCE access via DNS, the TCP/IP Domain Name Service, or CCITT X.500, which is provided by GDS.

The GDA is the CDS gateway to GDS. Both CDS and GDS offer X/Open Directory Service (XDS) API as their programming interface. For a complete overview on the CDS facility refer to Chapter 6 in the ITSO Austin publication *OSF DCE for AIX, OS/2 and DOS Windows Overview* and DCE related documentation listed in “Related Publications” on page xvi.

1.3.3.5 DCE Distributed Time Service

DTS provides precise, fault-tolerant clock synchronization on the computers participating in a Distributed Computing 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, heavily rely 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 which 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. They advertise themselves into the CDS so they can be contacted by other DTS servers or even clerks, if these 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 clock with values obtained from other DTS servers.
- Courier DTS Servers: Usually one per LAN. 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. By doing so they maintain synchronization between multiple LANs.

A complete set of 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 refer to Chapter 7 in the ITSO Austin publication *OSF DCE for AIX, OS/2 and DOS Windows Overview* and DCE related documentation listed in "Related Publications" on page xvi.

1.3.3.6 Distributed File System

DCE 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 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, which conform to POSIX 1003.6. DFS implements a superset of that POSIX ACL Draft.

As shown in Figure 8 on page 11 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. It depends on the DCE Time Service to keep the clocks in synchronization.

The DCE LFS is a log-based file system that is integrated into the kernel. Also it is based on aggregates which 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 online with minimal down time for users of the filesets.

Directories and files can be accessed from users anywhere on the network, using the same name 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 new 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 userID/nodeID, which will be revoked when the ticket expires.

For a general overview on DFS facility refer to Chapter 8 in the ITSO Austin publication *The Distributed File System (DFS) for AIX/6000* and DCE related documentation listed in “Related Publications” on page xvi.

1.3.3.7 Diskless Workstation Support

A diskless system has no local disk or any other storage media. It is connected to other machines through local area network (LAN) connection. It is not very different from any other DCE client machine except for the following requirements imposed by the absence of a disk:

- The boot image is stored on a server and is obtained when the power is turned on.
- The root file system is stored on a server and is accessed remotely through the distributed file system.
- The distributed file system uses a memory-resident cache instead of a disk-resident cache.
- The virtual memory manager (VMM) pages to a virtual disk, which is mapped to a file or a device on a server.

IBM has not shipped the OSF DCE Diskless technology, and OSF has now dropped diskless support, too. So, the *local* file systems of the diskless client have to be accessed in the traditional way, via NFS. Paging can occur to a local disk (dataless client) or via NFS to a server. Once this client has booted properly it can run as a normal DCE and DFS client.

1.3.3.8 PC Integration

IBM offers PC integration into DCE for OS/2 and for Windows. On OS/2 a client package and a server package are available, on Windows only a client package is available. OS/2 will add a DFS client sometime in the future and Gradient will do the same for the Windows part. For a complete overview on PC-Integration refer to the ITSO Austin publication *OSF DCE for AIX, OS/2 and DOS Windows Overview* and DCE related documentation listed in “Related Publications” on page xvi.

1.3.3.9 Mutual Dependencies between DCE Components

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

Table 1. Dependencies between the DCE Components

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

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

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

1.3.3.10 DCE Management Services

Several administration tools are provided to manage DCE. The following are provided by OSF and 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
passwd_export	Creates /etc/passwd type file out of registry entries

- 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

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 new high level commands 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
rmdfsjlfs	Removes a JFS file system from the DFS file space
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.

1.3.3.11 Organization of a Distributed Computing Environment

In DCE a cell represents the smallest unit of resource 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
- At least three Distributed Time Servers (optional, but recommended)

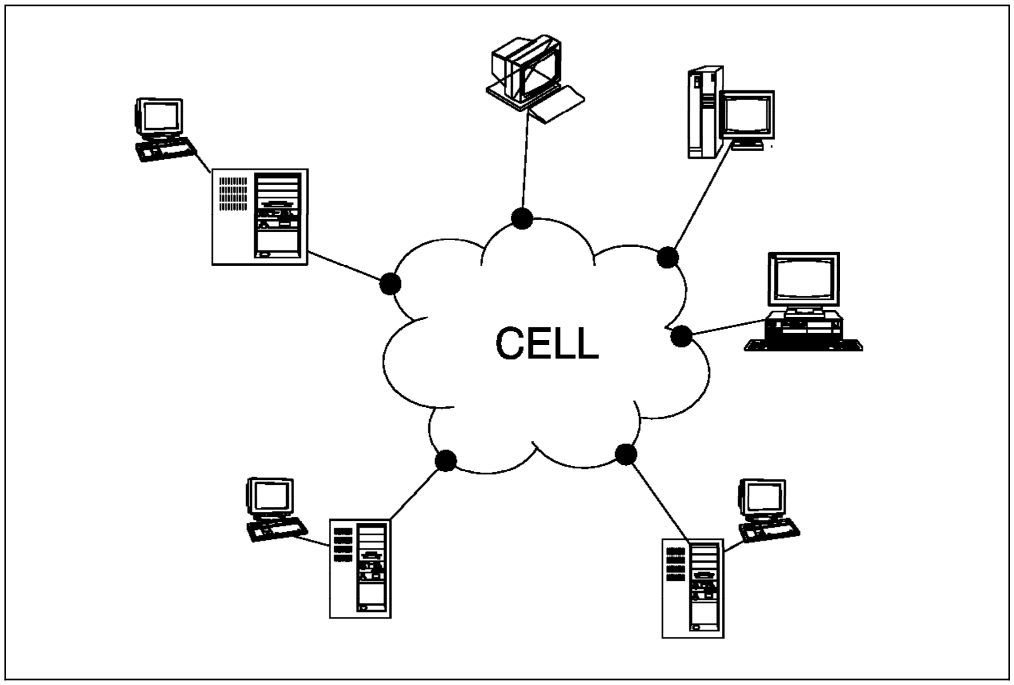


Figure 10. DCE Cell

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 10.

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

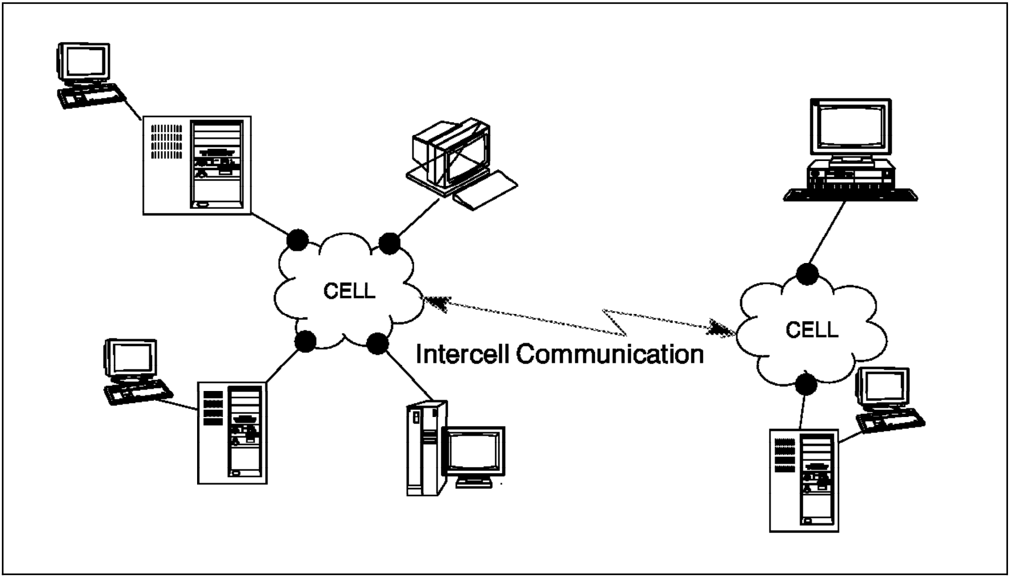


Figure 11. DCE Multi Cell Environment

Inter-cell 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 AIX DCE Version 1.3 introduces a fast local transport family, the local UNIX sockets, for cases where clients and servers are on the same machine.

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.

1.3.3.12 DCE Packaging and Cost

The following two tables give an overview on order numbers, current prices and functionality of the different products available for AIX, OS/2 and DOS Windows.

Table 2 (Page 1 of 2). IBM DCE Products for Workstations

Product Number	Description	Price ($)
	AIX	
5765-232	DCE Threads for AIX 1.3 (1)	99
5765-117	DCE Base Services for AIX 1.3 (DCE only)	295
5765-117	DCE Base Services for AIX 1.3 (DCE & Encina)	395
5765-117	DCE Base Services for AIX 1.3 (Encina only)	150
5765-118	DCE Security Server for AIX 1.3	(2) >2,300
5765-119	DCE Cell Directory Server for AIX 1.3	(3) >2,300
5765-121	DCE Enhanced Distributed File System for AIX 1.3	(4) >2,900
5765-120	DCE Global Directory Server for AIX 1.3	(5) >2,000
5765-259	DCE Global Directory Client for AIX 1.3	195
5765-456	DCE Manager for AIX 1.3	3000
5765-457	DCE NFS to DFS Authenticating Gateway for AIX 1.3	(6) >1,700

Table 2 (Page 2 of 2). IBM DCE Products for Workstations

Product Number	Description	Price ($)
	OS/2	
5696-692	IBM DCE Client for OS/2	65
5696-657	IBM DCE for SDK OS/2 and Windows	1,095
	DOS/WINDOWS	
5696-690	IBM DCE Client for Windows	65

(1) The Threads package is included in DCE Base for AIX. This separate offering enables the usage of threads with AIX 3.2.5 in environments without DCE.

(2) The indicated price is valid for processor group D5. The full list of prices for the security server is $2,300 (D5), $3,050 (E5), $5,750 (F5), $8,625 (G5), and $2,670 (P5).

(3) The indicated price is valid for processor group D5. The full list of prices for CDS is $2,300 (D5), $3,050 (E5), $5,750 (F5), $8,625 (G5), and $2,670 (P5).

(4) The indicated price is valid for processor group D5. The full list of prices for DFS is $2,900 (D5), $3,900 (E5), $5,700 (F5), $8,550 (G5), and $2,625 (P5).

(5) The indicated price is valid for processor group D5. The full list of prices for the GDS server is $2,000 (D5), $3,050 (E5), $5,750 (F5), $8,625 (G5), and $2,670 (P5).

(6) The indicated price is valid for processor group D5. The full list of prices for the NFS to DFS gateway is $1,700 (D5), $1,950(E5), $3,650 (F5), $5,475 (G5), and $1,710 (P5).

The DCE implementations for OS/2 and DOS/Windows are based on OSF DCE 1.0.1. They operate on Windows 3.1 and OS/2 2.1.

Table 3 (Page 1 of 2). Functionality of the Different DCE Products

	AIX DCE 1.3							**OS/2 & Windows**		
DCE Package ->	**Threads**	**Base**	**Security**	**CDS**	**EDFS**	**GDS.Server**	**GDS.Client**	**SDK (1)**	**OS2.Client**	**Win.Client**
Threads	√	√						√	√	√
RPC		√						√	√	√
Security Client		√						√	√	√
Security Master			√					√		
Security Replication			√							
CDS Client		√						√	√	√

Table 3 (Page 2 of 2). Functionality of the Different DCE Products

DCE Package ->	**AIX DCE 1.3**							**OS/2 & Windows**		
	Threads	**Base**	**Security**	**CDS**	**EDFS**	**GDS.Server**	**GDS.Client**	**SDK (1)**	**OS2.Client**	**Win.Client**
CDS Server				√				√		
CDS Replication				√				√		
DTS Client		√						√	√	√
DTS Server		√						√		
DFS Client		√								
DFS Server (JFS)		√								
DFS Server (LFS)					√					
DES (US Only)		√						√		
CDMF (exportable)		√								
X.500 Access							√			
X.500 Directory						√				
OSF DCE Level	1.0.3							1.0.1		

(1) The Software Development Kit includes five licenses each of the OS/2 and the DOS Windows DCE clients.

1.3.4 NIS/NFS

Sun Network Information System / Network File System (NIS/NFS) was released in 1985 by Sun Microsystems. NFS was a more successful product than the equivalent Remote File System (RFS) implemented by AT&T** in the same period. Sun published the specifications and made them publicly available. NFS is the most popular solution for sharing file systems since it is available on almost any operating system. Sun Remote Procedure Calls (Sun RPCs) are used to build NFS. RPC is built on top of the External Data Representation (XDR) protocol, which standardizes the various data types used in remote communications. UDP/IP is the transport protocol used in NFS. NFS daemons provide services such as mount and control of remote file systems.

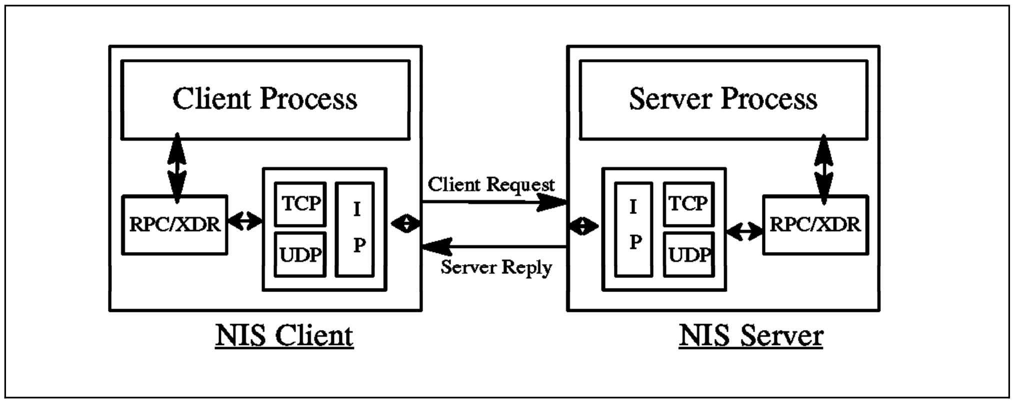


Figure 12. NIS and NFS, a Client/Server Environment

Network Information System (NIS) is also a Sun RPC-based application. This service retrieves information about users, passwords and host names from the NIS database, so called NIS maps. The master NIS database and related daemons can be replicated on other slave servers for high availability.

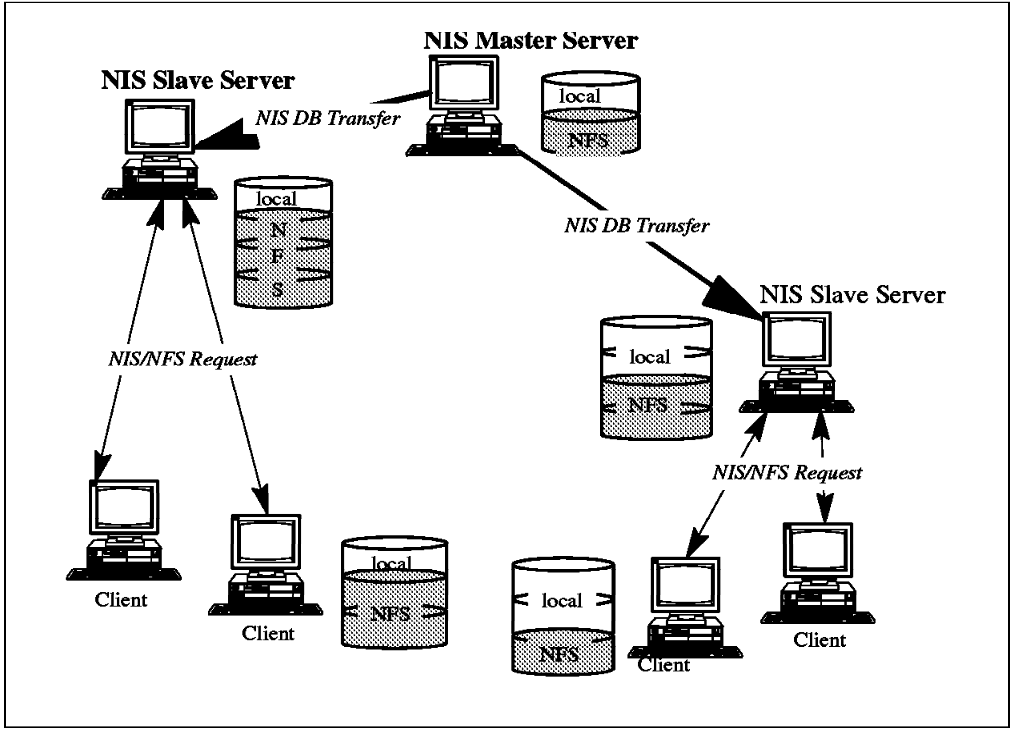


Figure 13. A Replicated NIS/NFS Environment

Although NFS and NIS are usually installed together, each one is independent of the other and each one is configured and administered individually.

As we mentioned earlier, NIS/NFS is very popular but it suffers from severe security exposures. In Figure 14 on page 25 you see one of the most common threats in an NIS/NFS environment. An intruder system can be plugged into the network and masquerade as another identity. It then can intercept data (lack of confidentiality), interrupt the service (lack of availability), and modify the data (lack of integrity).

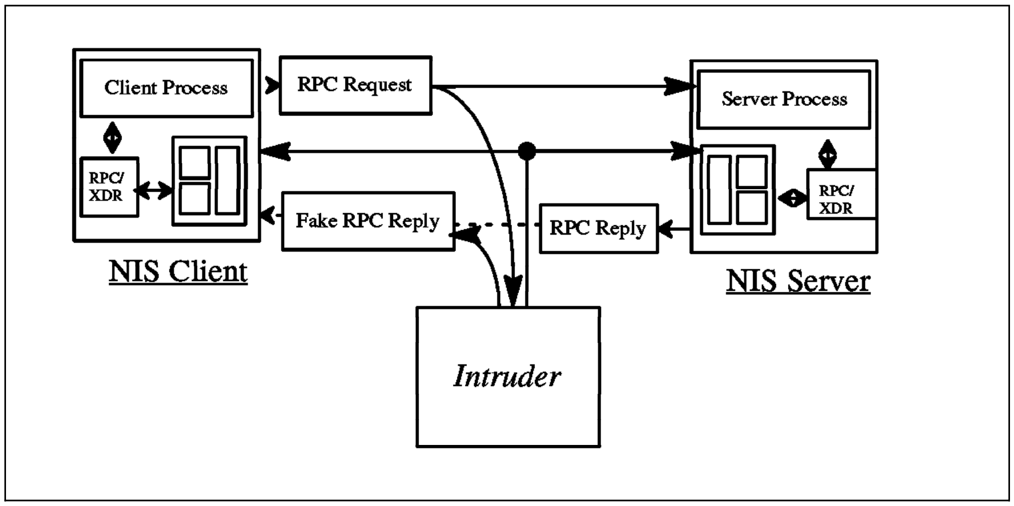


Figure 14. Security Breach in a NIS/NFS Environment

Security improvements are being made, but NFS and NIS were originally designed with one goal in mind: to share file systems and resources on the network. Security was not an issue. A security architecture for NIS/NFS will require a complete redesign of such a distributed environment. This would cause an obvious problem of interoperability with existing NIS/NFS installations.

1.3.5 IBM HACMP/6000

IBM AIX High Availability Cluster Multi-Processing/6000 represents an important feature that combines software and hardware to minimize down time by quickly restoring services when a system, a component, or an application fails. While not instantaneous like fault tolerant or continuous availability systems, restoring services is rapid and usually takes only a couple of minutes.

HACMP/6000 features are needed in business critical applications such as order processing, debit/credit transactions in banking or hotel reservations and others. Particularly in conjunction with RAID disks HACMP/6000 provides a stable environment for RDBMS applications.

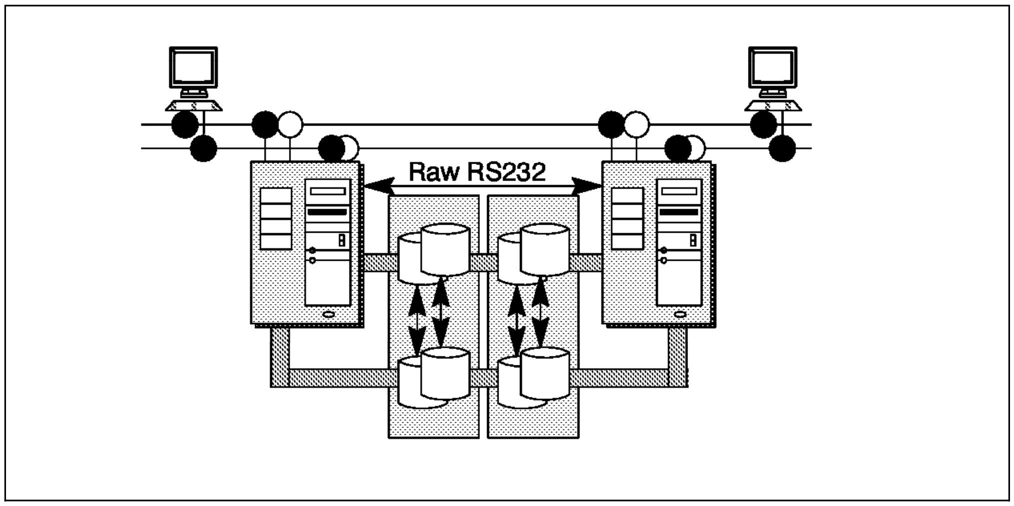


Figure 15. The HACMP/6000 Environment.

HACMP/6000 guarantees no single point of failure and operates in three modes:

- Idle standard known as mode 1

  One machine is sitting idle and watching the other one. If the other one appears the be dead, it takes over the disks and the IP address.

- Mutual operation takeover known as mode 2

  Both machines are doing work. Some of the disks are assigned to one, the rest of the disks to the other machine. Both are watching each other, ready to take over the others resources.

- Concurrent operation takeover known as mode 3

  They actually share the disks and are able to concurrently access them. Special daemons control and serialize competing disk access requests. Programs running in this mode such as ORACLE** Parallel Database Server have to use the special API provided by those daemons.

HACMP/6000 requires extra hardware to operate correctly, such as two TCP/IP link attachments per cluster node, a direct serial connection, and shared (twin-tailed) SCSI or serial disks. It is supporting configurations of AIX DCE and Encina in mode 1 and mode 2 with the restriction that DCE services are running only on one machine.

AIX DCE Servers such as CDS and Security have been tested in environments where the servers can fail over to a second RISC System/6000* in hot-standby, rotating standby, and one-sided takeover configurations. These are all special cases of a mode 1 environment. See 5.6.1, “HACMP/6000 Support for DCE” on page 273 for more explanations on these different mode 1 configurations.

Both security and CDS servers can be replicated within DCE, but if the system containing the master database fails, write access is not possible anymore. Tickets can be issued from a slave security server, thus leaving the cell operational. However, if for any reason write access to the security registry is always required, the master security server must be in a HACMP cluster. CDS is a little bit different. When application servers start, they export their binding information into the CDS namespace. Also the program offering Single Login/6000 introduced in 5.4, “Single Login/6000” on page 235 requires write access to CDS to store user login information. So, running CDS in an HACMP cluster always makes sense, if the directories to which write access is needed are not distributed.

All the write access will continue to work properly guaranteeing the most reliable DCE cell environment. For more details on this topic see 5.6, “DCE on IBM AIX High Availability Cluster Multi-Processing/6000” on page 272.

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

Todays 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 does 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 which can happen.

2. How is your company structured?

   - How large are the branch or regional offices (branches)?
   - 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. It is 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. To make changes is always more expensive than to carefully plan ahead.

4. Does your company have security conventions?

   Most companies have security conventions or even a security department which 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 other network operating systems than DCE, which is likely, you must consider how these different systems can coexist. In certain cases, these systems do not only have to coexist but to 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.

All this means, you must understand the way your whole company works, rather then just having 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

This section describes how the core components work with regards to planning of the cell layout. The aspects to look at are:

- Replication capabilities
- Server selection mechanisms

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, the administrator must define every detail. He must explicitly create replicas, define the replica set with a master, and must 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, which follows the global naming convention, to locate the binding information to 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 of all filesets in DFS, which are DFS file server addresses.

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 the ubik library routines 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.

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

2.2.2 Server Selection Mechanisms

The clients to services that can be replicated as described in the previous section all use a random server selection. 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. The requester can get all these addresses 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 a 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 security calls 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 73 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 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

The selection of the FLDB is random. Nothing can be configured.

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.

2.3 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 3-4K 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 three entries in that directory for the cdsclerk, rpcd and profile. Some quick experiments adding the entries for a thousand client machines show a pretty linear growth of 1.4K per additional machine in disk space requirements. Memory usage increased by about 1.7K 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 contributor 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 which 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. That 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 a piece 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 tracked back, 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 basically define only groups into ACLs of objects and not single users. 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 importance of 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
- 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 inter cell access, having unique UIDs is:

- Useful with DFS so that a company wide unique user name is shown as owner of files when a directory is listed.
- Required at least for global users when Single Login/6000 is used. Global users as defined in this product are those, who are allowed to go to another cell′s machine and login with authentication from their home cell. Their UIDs and names as defined in their home cell are added to the local /etc/passwd file to give them a local identity.

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 which is 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.

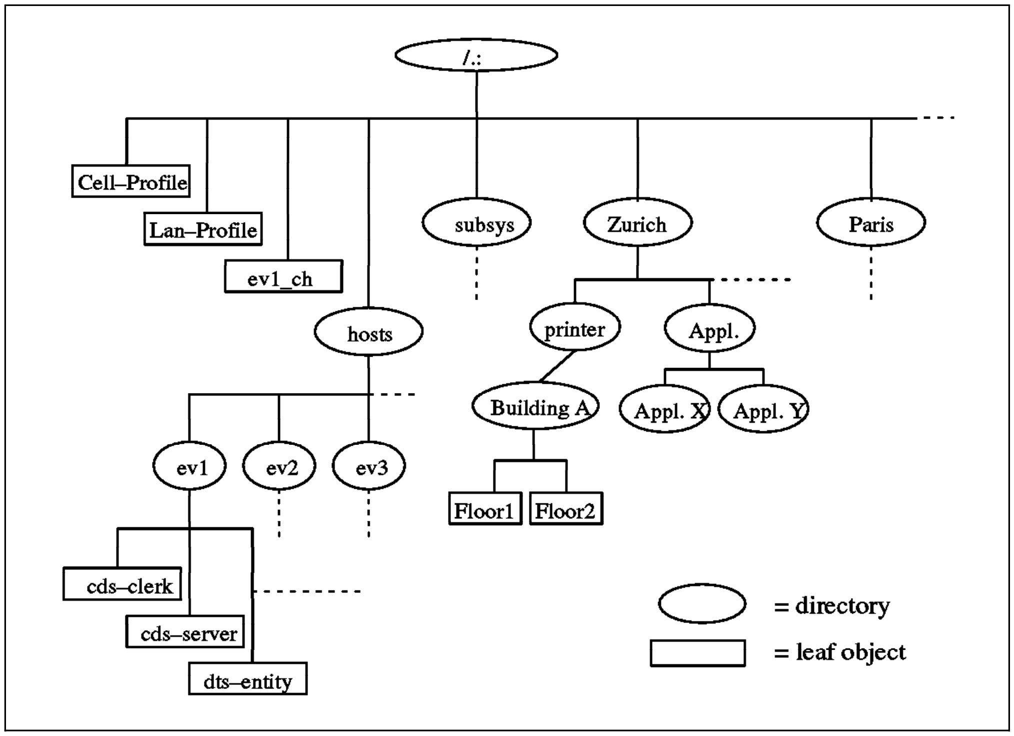


Figure 16. Example of a Cell Namespace

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 which contains all master replicas and another one that contains all read-only replicas. This ensures the availability of the namespace.
2. Have 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 Conclusions and Planning Tips

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

User requirement considerations:

- Business data and service flow
- Performance
- Availability
- Security
- Cost

Technical implementation considerations:

- Server selection mechanisms
- Replication capabilities

Geographical considerations are:

- 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, not to say impossible, task. Like any other decision 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.

2.6.1 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
- Are there remote locations, called branches hereafter
- Type of business
- How large are the branches
- How many and what kind of network services do the branches need
- How is the 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 (example banking). The opposite of that could be a company which is working in different business areas with many independent departments (example university).

2.6.2 Tips for Service Layout and Application Design

What a DCE user perceives is how the application performs and whether it is reliable as far as availability is concerned. This actually depends on many different factors:

- Application architecture and implementation in DCE
- How robust is the application (replication)
- How does the application cope with more users and data
- How does the application use the DCE core services
- How often does an application use the DCE core services
- How fast are the network connections
- How robust is the network
- How robust are the DCE core services (replication, availability)

The term *robust* stands for the ability to deal with error conditions and provide high availability.

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 above listed technical aspects.

2.6.2.1 Application 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.

2.6.2.2 How Robust is 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, ranging 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.

2.6.2.3 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.

2.6.2.4 How Does the Application Use 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 timeouts 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.

2.6.2.5 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 login several times a day and use many different applications, or if a lot of applications are started and stopped again, security and CDS services 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.

2.6.2.6 How Fast are 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.

2.6.2.7 How Robust is 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 multi-protocol 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.

2.6.2.8 How Robust are 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 and ATM or 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 34. 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 locationwise, 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 get performance problems because of this CDS configuration, you can install CDS servers in the remote location that has the problem and install replicas of these location specific directories. Directories which are used by all locations should not be replicated to a remote location, because all locations not having a replica would randomly access also this remote location over the slow link.

If you want to avoid any calls for location specific objects to the central site, move the 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 where *you* want. So you have to work with the pe_site files as described in 2.2.2.1, "Security Service" on page 31.

DFS FLDB:

If the DFS FLDB is replicated, which DFS file server a DFS client accesses is randomly chosen and not configurable. The ubik routines determine among multiple copies of the database which one is the master and there is no possibility to configure which FLDB is going to be contacted by DFS clients.

Therefore replicate it for performance and availability reasons, but only in the main site. Completely 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 locationwise into filesets. Start off with central DFS file servers. Install file servers in the remote locations as soon as there is demand for it. 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 out in the locations. For filesets which 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 which 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).

2.7 Planning Summary

We have discussed planning relevant issues in several sections throughout the first three chapters of this document. The following is a list of the most important references:

- 2.6.1, “One Cell or Multiple Cells?” on page 35
- 2.4, “Planning the User Namespace” on page 33
- 2.5, “Planning the CDS Namespace” on page 34
- 2.6.2, “Tips for Service Layout and Application Design” on page 36
- Chapter 3, “Implementing DCE Cells” on page 43, 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 which 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 multi-protocol 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 which might be temporarily unavailable or 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 2.6.2.8, "How Robust are the DCE Core and DFS Services?" on page 38.

  As a very simplified summary of Chapter 2, "Planning DCE Cells" on page 27 and Chapter 3, "Implementing DCE Cells" on page 43 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 3.1.2, "Scenario 2: Master Servers on One Machine and Replicas on Another" on page 48.

  – 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 3.2.5, "Scenario 6: A Branch Connected with Two Links" on page 81.

  – 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 2.6.2.8, "How Robust are the DCE Core and DFS Services?" on page 38.

- 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 layed 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 avoid unnecessary timeouts by correctly handling interface exports to CDS. See also 2.6.2.4, "How Does the Application Use the DCE Core Services?" on page 37.

- Use HACMP/6000 for highly available write access

  Put the DCE core services, DFS, or any application server on HACMP/6000, when you need to guarantee write access all the time. 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 for example accounts. Tickets are granted without write access.

Chapter 3. Implementing DCE Cells

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 a customer to decide what abstract or technical solution fits his business best when he knows what he 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. NFS/NIS environments
4. High availability (HACMP) scenarios

This chapter covers the first two scenario groups. We look at different network topologies and vary the placement of the different core services. We provide step-by-step implementation instructions for eight 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.

The NFS/NIS environment and HACMP/6000 are actually topics that are equally relevant to all of the above scenarios and can, as well, be treated separately in the task-oriented Chapter 4, “Administering DCE Cells” on page 83 which describes different administrative tasks.

For an overview of our test environment with all IP addresses, host names and machine configurations see Appendix B, “Description of the Systems in our Scenario” on page 279.

3.1 Local (LAN-type) Cells

This section discusses cell topologies which can be considered a local environment without slow communication links. 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 a 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 services on the same server, no replication
- All master service on one server and replicated servers on another
- Master and replica servers on different nodes

We provide all commands to create each specific scenario such that there is a complete guideline which can be followed. For more explanations, sample SMIT screens, and command output see 4.1, “Configuring a Cell” on page 84.

3.1.1 Scenario 1: All Servers on One Machine without Replicas

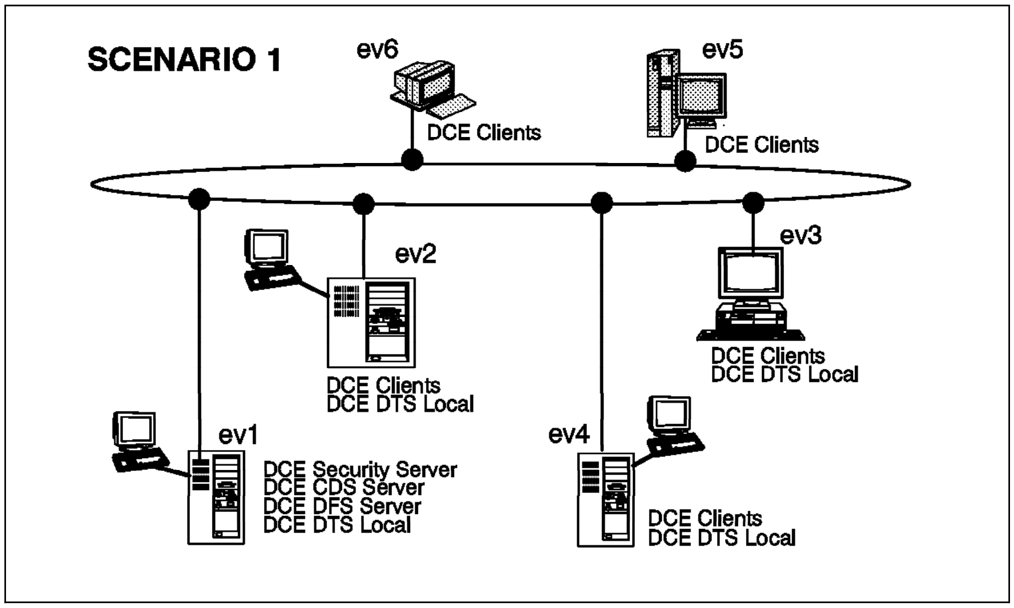


Figure 17. Scenario 1: One Server Machine - No Replicas

3.1.1.1 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 4.1.1, "Preparing for DCE Configuration" on page 84.

3.1.1.2 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
   #rgy_edit -v
   #cdsli -world
   #rpccp show mapping
   #exit
   ```

2. Configure the DFS components:

   a. Configure the System Control machine:

      ```
      #mkdfs dfs_scm
      ```

   b. Configure the DFS Fileset Database machine:

      ```
      #mkdfs dfs_fldb
      ```

c. Configure the DFS File Server Machine: The -e flag loads the DFS kernel extension for now and for subsequent restarts.

```
#mkdfs -e dfs_srv
```

d. Configure the DFS Client machine:

```
#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. Login as cell_admin:

```
#dce_login cell_admin cell_password
```

h. 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.

i. Create another fileset:

- Create a logical volume /dev/usrbin with five blocks of 4MB:

  ```
  #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 with mount point:

  ```
  #mkdfslfs -f usrbin.ft -m /:/usrbin -n usrbin
  ```

- Test that the fileset is correctly exported:

  ```
  #fts lsfldb
  ```

***Configuring machines** ev2, ev3, ev4*

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
   #rgy_edit -v
   #cdsli -world
   #rpccp show mapping
   ```

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. Repeat above steps for ev3 and ev4:

3.1.1.3 Scenario Experiences

If configuring the DFS server is not successful the first time, you must clean all RPC mappings and CDS caches and reboot the machine. Use the script file `cleanup_cache` we provide on the diskette with this document to clean all mappings and caches, if this problems happen to you. See also 4.4.5, "Managing Caches on Client Machines" on page 156.

3.1.1.4 Special Issues

Remember, this is our reference scenario and is not a realistic production environment because of serious availability concerns.

3.1.1.5 Response Times

All DCE/DFS commands we have tested in this scenario (`dce_login`, `rgy_edit`, `cdscp`, `rpccp`, `fts`, and others) took less than five seconds. We use the performance experienced in this scenario as a reference for the scenarios that follow.

3.1.1.6 Performance Discussion

Performance as far as access to CDS and the registry is concerned should not be an issue unless these databases are really big or *ev1* is heavily used for other purposes as well. If you start to get slow responses from CDS or the registry you might consider:

- Moving other services to a different machine.
- Replicating CDS and security servers, which means load balancing and improving availability at the same time.
- Separating CDS and security servers as a last resort, if their databases are really big.

3.1.1.7 Availability Discussion

Availability is not discussed in this scenario. If the one server machine fails, the cell is dead. If such a cell is installed, be sure to back up all DCE databases to be able to recreate the server machine. See 4.4, "Backup/Restore and Other Housekeeping Tasks" on page 142 for a description of backup/restore issues.

3.1.2 Scenario 2: Master Servers on One Machine and Replicas on Another

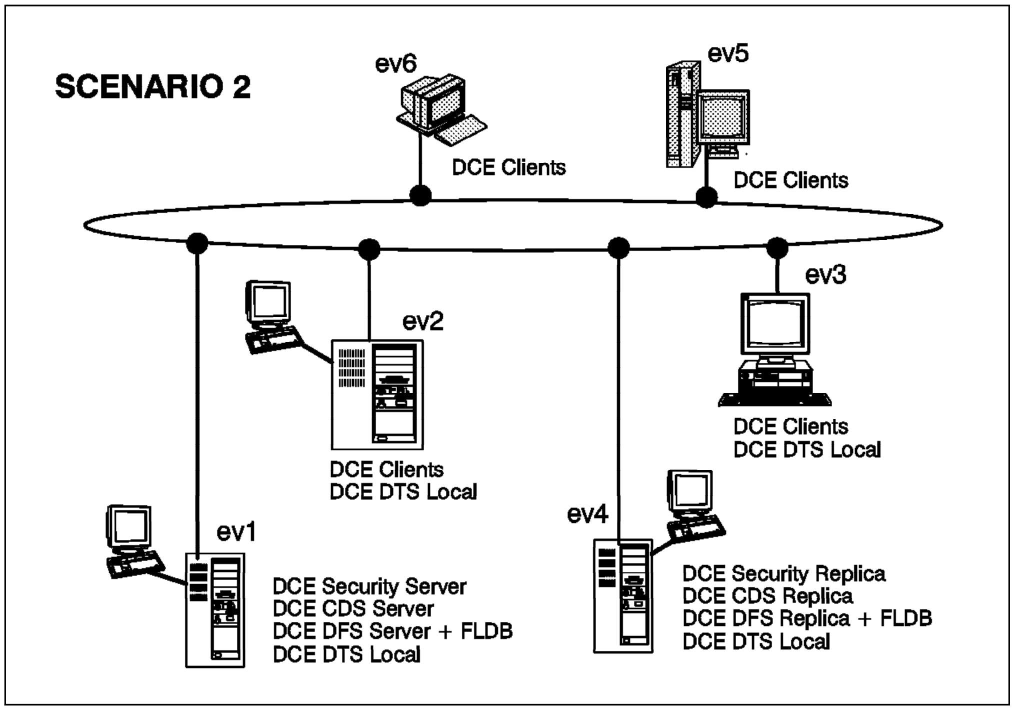


Figure 18. Scenario 2: One Master Server - One Replica Server

3.1.2.1 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 4.1.1, "Preparing for DCE Configuration" on page 84.

3.1.2.2 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
   #rgy_edit -v
   #cdsli -world
   #rpccp show mapping
   #exit
   ```

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. Login as cell_admin:

```
#dce_login cell_admin cell_password
```

e. 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.

f. 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 5.2, "DFS Replication" on page 224 and 4.2.3, "Replicating DFS Server" on page 107.

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
```

g. Create another fileset:

- Create a logical volume /dev/usrbin with five blocks of 4MB:

  ```
  #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
  ```

h. 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
```

i. 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
```

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
#rgy_edit -v
#cdsli -world
#rpccp show mapping
```

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 4.1.6.1, “Replicating a CDS Server” on page 98 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
      ```

      See the remarks about the timing problem in 4.2, “Configuring DFS” on page 101 for reasons why this step is necessary.

   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/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
```

3.1.2.3 Scenario Experiences

Note that being successful configuring a secondary CDS server does not mean that any directories and names are replicated. You have to select which directory to replicate on 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.

3.1.2.4 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.

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 4.2.3, “Replicating DFS Server” on page 107 and 5.2, “DFS Replication” on page 224 for more information on how to replicate filesets.

3.1.2.5 Response Times

All DCE/DFS commands we have tested in this scenario (`dce_login`, `rgy_edit`, `cdscp`, `rpccp` and others) took less than five seconds, when all servers are available. It seems as good as scenario 1.

3.1.2.6 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 is read-mostly to replicated filesets.

Depending on the problems that are 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

3.1.2.7 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, 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.

HACMP should be considered for all cases in which even temporary unavailability is unacceptable.

3.1.3 Scenario 3: Master Servers and Replicas on Different Machines

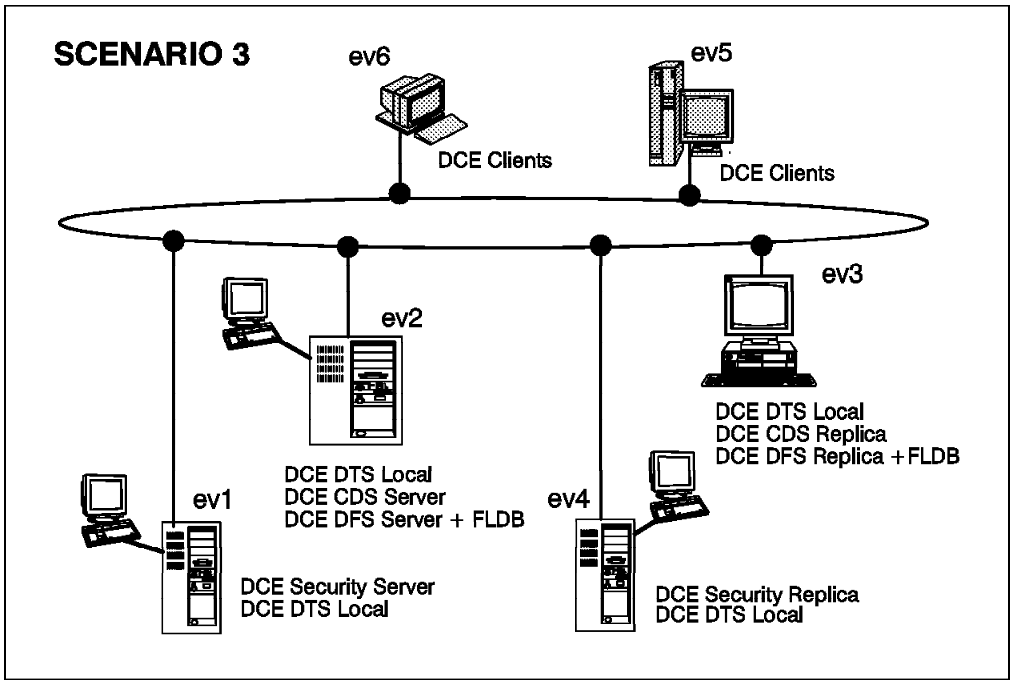


Figure 19. Scenario 3: DCE Servers on Different Machines

3.1.3.1 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 4.1.1, "Preparing for DCE Configuration" on page 84.

3.1.3.2 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
#rgy_edit -v
#cdsli -world
#rpccp show mapping
```

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. Login as cell_admin:

      ```
      #dce_login cell_admin cell_password
      ```

   h. 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.

   i. 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 5.2, “DFS Replication” on page 224 and 4.2.3, “Replicating DFS Server” on page 107.

      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/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
```

j. Create another fileset:

- Create a logical volume /dev/usrbin with five blocks of 4MB:

```
#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
```

k. 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
```

l. 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
```

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 4.1.6.1, “Replicating a CDS Server” on page 98 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
```

3.1.3.3 Scenario Experiences

Same discussion as in scenario 2, see 3.1.2.3, “Scenario Experiences” on page 52

3.1.3.4 Special issues

There are no special issues in this scenario.

3.1.3.5 Performance Discussion

Same discussion as scenario 2, see 3.1.2.6, “Performance Discussion” on page 53

3.1.3.6 Availability Discussion

Same discussion as on scenario 2, see 3.1.2.7, “Availability Discussion” on page 53

3.2 LAN/WAN Cells

This section discusses cell topologies which 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 couple of 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 which does not run any servers or services and a large one which consists of a couple of servers and many client workstations. The following is a list of two site scenarios with different link types, which we look at in this section:

- A small branch connected via X.25 (scenario 4a)
- A small branch connected via SLIP (scenario 4b)
- A large branch connected via X.25 (scenario 5a)
- A large branch connected via SLIP (scenario 5b)
- A branch with redundant communication links (scenario 6)

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, such 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 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 4.1.1.3, "Checking Network Routing" on page 86 for details.

For a summary of our findings and planning hints see Chapter 2, "Planning DCE Cells" on page 27.

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 4.1, "Configuring a Cell" on page 84.

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.

3.2.1 Scenario 4a: A Small Branch Connected via X.25

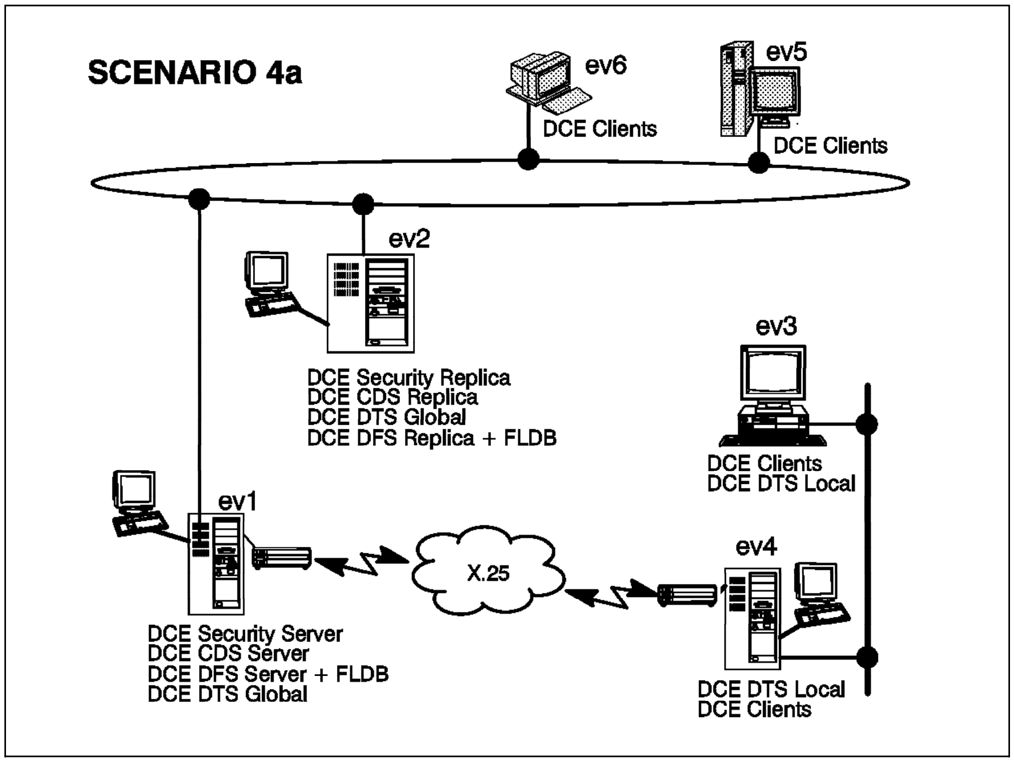


Figure 20. Scenario 4a: A Small Branch Connected via 19,200bps X.25

3.2.1.1 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 4.1.1, “Preparing for DCE Configuration” on page 84. 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   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
xt0   576   <Link>                   3     0        3  0     0
xt0   576   192.1.20    ev1          3     0        3  0     0
```

Check name resolution:

```
#host ev1
ev1 is 9.3.1.68,  Aliases:   ev1tr, ev1x25
#host ev4et
ev4 is 193.1.10.4,  Aliases:   ev4et
#host ev4x25
ev4 is 192.1.20.2,  Aliases:   ev4x25
```

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
```

Exclude the X.25 interface now and forever:

```
export RPC_UNSUPPORTED_NETIFS=xt0
echo "export RPC_UNSUPPORTED_NETIFS=xt0" >> /etc/environment
```

***On machine** ev2:* List the network interfaces:

```
#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       ev2      17654     0    14115  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>               14992     0    14992  0     0
lo0   1536  127         loopback 14992     0    14992  0     0
en0   1492  <Link>               17654     0    14115  0     0
en0   1492  193.1.10    ev3      17654     0    14115  0     0
```

Check name resolution and set the appropriate route:

```
#host ev4
ev4 is 193.1.10.4
#host ev3
ev3 is 193.1.10.3,  Aliases:   ev3et
#host ev1
ev1 is 9.3.1.68
```

```
#route add default ev4 1
```

***On machine** ev4.:* List the network interfaces:

```
#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
en0   1492  <Link>               17654     0    14115  0     0
en0   1492  193.1.10    ev4      17654     0    14115  0     0
xt0   576   <Link>                   3     0        3  0     0
xt0   576   192.1.20    ev4          3     0        3  0     0
```

Check name resolution and set the appropriate route:

```
#host ev1
ev1 is 9.3.1.68,  Aliases:   ev1x25
#host ev1x25
ev1 is 192.1.20.3,  Aliases:   ev1x25
```

```
#route add -net 9.3.1 ev1x25 1
```

3.2.1.2 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
   #rgy_edit -v
   #cdsli -world
   #rpccp show mapping
   #exit
   ```

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. Login as cell_admin:

      ```
      #dce_login cell_admin cell_password
      ```

   e. 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.

   f. 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 5.2, “DFS Replication” on page 224 and 4.2.3, “Replicating DFS Server” on page 107.

      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
```

g. Create another fileset:

- Create a logical volume /dev/usrbin with five blocks of 4MB:

  ```
  #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
  ```

h. 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
```

i. 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
```

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 4.1.6.1, “Replicating a CDS Server” on page 98 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
   ```

   See the remarks about the timing problem in 4.2, “Configuring DFS” on page 101 for reasons why this step is necessary.

   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:

```
#dce_login cell_admin cell_password
#rgy_edit -v
#cdsli -world
#rpccp show mapping
```

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:

3.2.1.3 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.

3.2.1.4 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. We decided that the central site only 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.

3.2.1.5 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 we have all DCE/DFS clients over a WAN, access to DCE always takes more time, but all DCE commands work correctly with an acceptable response time.

3.2.1.6 Performance Discussion

Having replicated servers means load balancing as we discussed in scenario 2 (see 3.1.2.6, “Performance Discussion” on page 53) 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.

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 site with DCE servers a large branch. See in 3.2.3.6, “Performance Discussion” on page 72 (scenario 5a) and 3.2.4.6, “Performance discussion” on page 79 (scenario 5b) for a discussion on server replication over WAN.

3.2.1.7 Availability Discussion

The availability discussion as far as the central site is concerned is the same as in scenario 2 (see 3.1.2.7, “Availability Discussion” on page 53).

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 (see above performance discussion) or redundant layout of the underlying TCP/IP with dynamic routing. See also 3.2.5, “Scenario 6: A Branch Connected with Two Links” on page 81.

3.2.2 Scenario 4b: A Small Branch Connected via SLIP

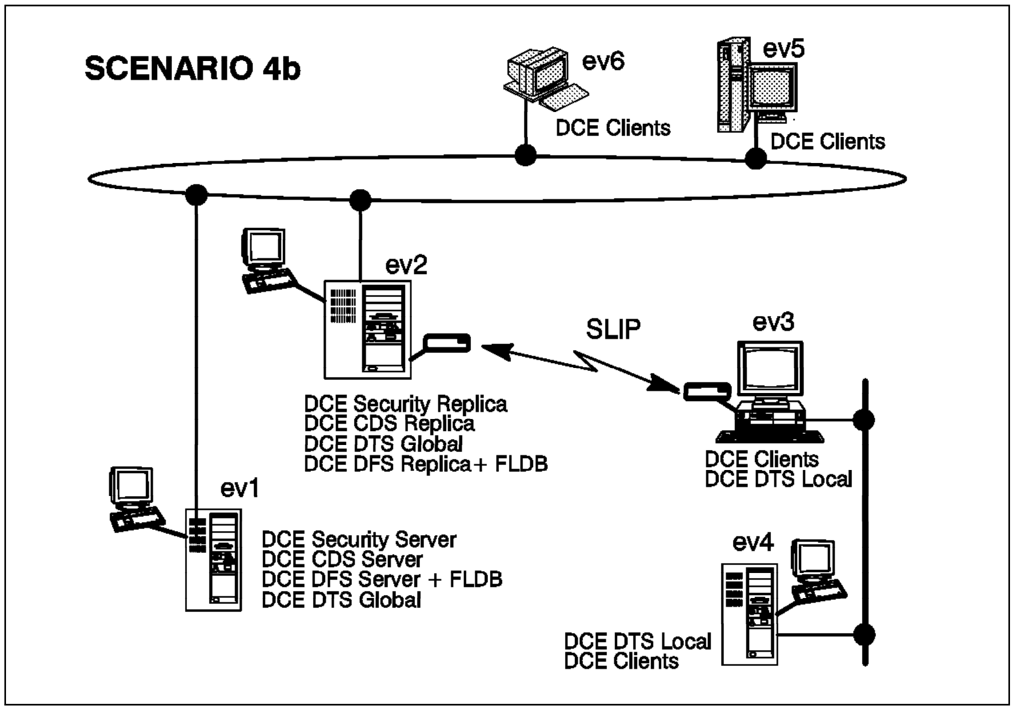


Figure 21. Scenario 4b: A Small Branch Connected via 9600bps SLIP

3.2.2.1 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 - see below
- Checked the network interfaces
- Synchronized the system clocks
- Installed DCE (last of these steps)

For details see 4.1.1, "Preparing for DCE Configuration" on page 84. 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   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
```

Set up the appropriate route to reach the Ethernet network via *ev2*:

```
#route add -net 193.1.10 ev2 1
```

***On machine** ev2:* List the network interfaces:

```
#netstat -i
Name  Mtu   Network     Address  Ipkts    Ierrs Opkts    Oerrs Coll
lo0   1536  <Link>                45826     0    45826     0     0
lo0   1536  127         loopback 45826     0    45826     0     0
tr0   1492  <Link>              260541     0   213737     0     0
tr0   1492  9.3.1       ev2     260541     0   213737     0     0
sl0   1006  <Link>                 2430     0     2360     0     0
sl0   1006  192.1.21    ev2sl      2430     0     2360     0     0
```

Set the appropriate route to the Ethernet network via the SLIP interface of *ev3*:

```
#route add -net 193.1.10 ev3sl 1
```

Exclude the SLIP interface now and forever:

```
export RPC_UNSUPPORTED_NETIFS=sl0
echo "export RPC_UNSUPPORTED_NETIFS=sl0" >> /etc/environment
```

***On machine** ev3:* List the network interfaces:

```
#netstat -i
Name  Mtu   Network     Address  Ipkts    Ierrs Opkts    Oerrs Coll
lo0   1536  <Link>                45826     0    45826     0     0
lo0   1536  127         loopback 45826     0    45826     0     0
en0   1492  <Link>              123451     0   123457     0     0
en0   1492  193.1.10    ev3     123451     0   123457     0     0
sl0   1006  <Link>                    2360     0     2430     0     0
sl0   1006  192.1.21    ev3sl         2360     0     2430     0     0
```

Check name resolution and set the appropriate route:

```
#host ev2sl
ev2sl is 192.1.21.1
#route add default ev2sl 1
```

***On machine** ev4* List the network interfaces:

```
#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
en0   1492  <Link>                  17654     0    14115  0     0
en0   1492  193.1.10    ev4         17654     0    14115  0     0
```

Set the appropriate route:

```
#route add default ev3 1
```

3.2.2.2 DCE Configuration Steps

Follow scenario 4a by exchanging the roles of *ev1* and *ev2*:

***Configuring machine** ev1:* Follow the steps of "Configuring machine " on page 62 for scenario 4a.

***Configuring machine** ev2:* Follow the steps of "Configuring machine " on page 64 for scenario 4a.

***Configuring machines** ev3, ev4:* Follow the steps of "Configuring machines " on page 65 for scenario 4a.

3.2.2.3 Scenario Experiences

When we configure the DCE/DFS Clients machines on the Ethernet side, it takes normally more time than when we configure on the token-ring side where the servers are.

3.2.2.4 Special Issues

The SLIP interface is not recognized by DCE, which means when servers request to export their interfaces to the local RPC map and to CDS, SLIP is simply ignored. Thus, SLIP can be used for DCE, but only on the basis of IP routing. Standalone remote clients connected via SLIP need to define a dummy LAN network on their built-in Ethernet adapter to be able to configure DCE clients.

We excluded the SLIP interface anyway to make sure that the interface is also excluded in case the ignoring of SLIP is just a temporary restriction.

3.2.2.5 Response Times

On the token-ring side, even though we integrate a SLIP line, performance is still as good as in the scenarios 2 and 3. On the Ethernet side, response is not as good as on the token-ring side and not as good as over X.25, but still acceptable.

3.2.2.6 Performance Discussion

Same discussion as in scenario 4a (see 3.2.1.6, “Performance Discussion” on page 66).

3.2.2.7 Availability Discussion

Same discussion as in scenario 4a (see 3.2.1.7, “Availability Discussion” on page 66).

3.2.3 Scenario 5a: A Large Branch Connected via X.25

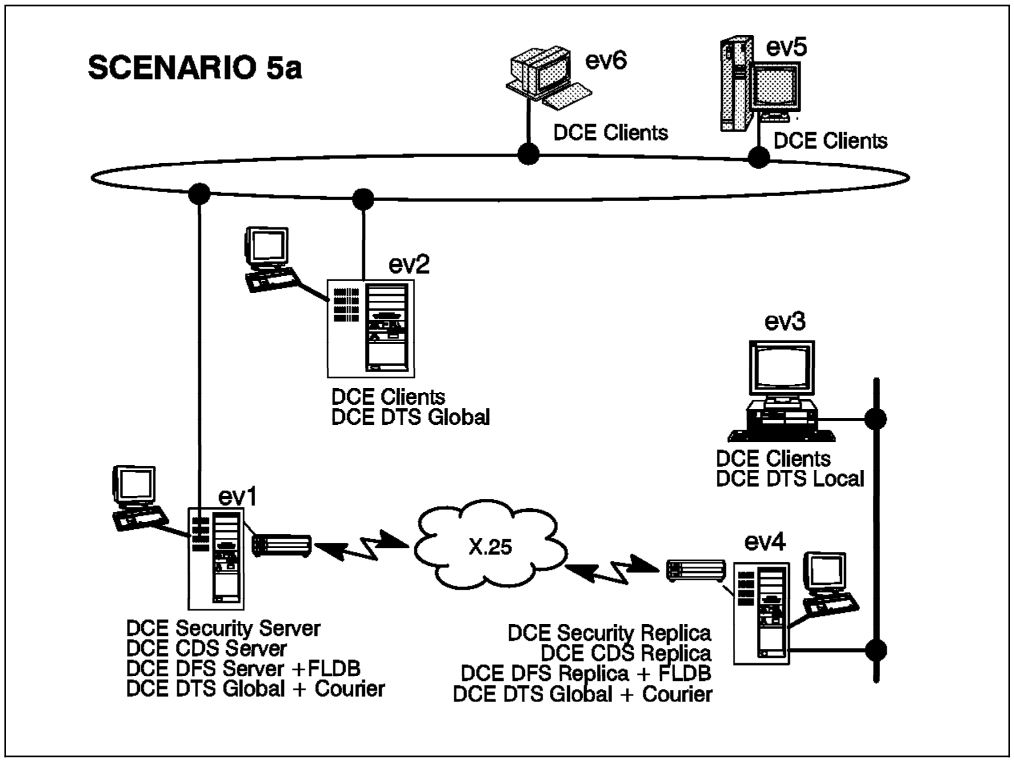


Figure 22. Scenario 5a: A Large Branch Connected via X.25

3.2.3.1 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 4.1.1, "Preparing for DCE Configuration" on page 84.

Be careful with the routing. The routes to be set are the same as in scenario 4a. See 3.2.1.1, "Preparation Steps" on page 60.

Exclude the X.25 interface on *ev1* and *ev4* now and forever:

```
export RPC_UNSUPPORTED_NETIFS=xt0
echo ″export RPC_UNSUPPORTED_NETIFS=xt0″ >> /etc/environment
```

3.2.3.2 Configuration Steps

To configure all machines in this scenario, you can follow exactly the same steps as in scenario 2, as described in 3.1.2.2, "DCE Configuration Steps" on page 48. 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
```

3.2.3.3 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. The link was saturated. With `xmonitor` we observed heavy X.25 traffic even while there was no user activity. 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.

3.2.3.4 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 his 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*.

3.2.3.5 Response Times

There is no difference from scenario 4a (3.2.1.5, “Response Times” on page 66), 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.

3.2.3.6 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 which 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 which 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 basically half of all these calls have to go across a slow WAN link. Is there anything that can be done to override random selection? However, before we implement any of the optimizations outlined below, 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 which 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, overproportional load would be 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 34 for a CDS design discussion.

FLDB and DFS file server:

Access of the FLDB cannot be predetermined. On the DFS client, preferences can be set for the cache manager to access certain 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, 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 as location oriented, so they can be local to the DFS clients as much as possible. If there is a lot of read-only access, replicas should be made for load balancing. If the filesets are defined as 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.

3.2.3.7 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 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 multi-protocol router network or by simply building a backup link which can be activated in case of a failure of the primary link. 3.2.5, “Scenario 6: A Branch Connected with Two Links” on page 81 discusses this topic.

3.2.4 Scenario 5b: A Large Branch Connected via SLIP

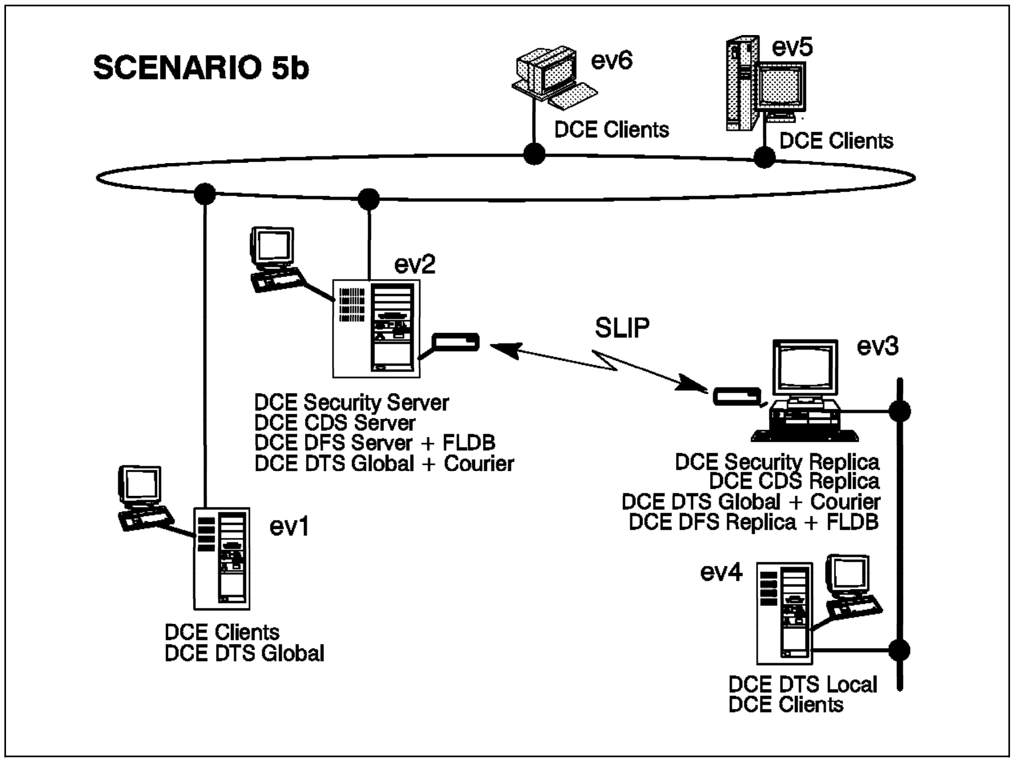


Figure 23. Scenario 5b: A Large Branch Connected via SLIP

3.2.4.1 Preparation Steps

Before you configure any of the DCE machines you should have:

- Created the necessary file systems
- Checked network name resolution
- Checked network routing - see below
- Checked the network interfaces
- Synchronized the system clocks
- Installed DCE (last of these steps)

For details see 4.1.1, “Preparing for DCE Configuration” on page 84.

Be careful with routing. The routes to be set are the same as in the scenario 4b. See 3.2.2.1, “Preparation Steps” on page 68.

Exclude the SLIP interface on *ev2* and *ev3* now and forever:

```
export RPC_UNSUPPORTED_NETIFS=sl0
echo "export RPC_UNSUPPORTED_NETIFS=sl0" >> /etc/environment
```

3.2.4.2 DCE Configuration Steps

Following are all the configuration steps for this scenario.

Configuring machine *ev2*

1. Configure the core components:

   ```
   #mkdce -n itsc.austin.ibm.com -t courier sec_srv cds_srv dts_global
   ```

   Test a few commands to see if DCE is working correctly:

```
#dce_login cell_admin cell_password
#rgy_edit -v
#cdsli -world
#rpccp show mapping
#exit
```

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. Login as cell_admin:

      ```
      #dce_login cell_admin cell_password
      ```

   e. 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.

   f. 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 5.2, "DFS Replication" on page 224 and 4.2.3, "Replicating DFS Server" on page 107.

      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/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
```

g. Create another fileset:

- Create a logical volume /dev/usrbin with five blocks of 4MB:

```
#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
```

h. 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
```

i. 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
```

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 ev2 -c ev2 -t courier sec_cl cds_cl dts_global
   ```

2. Configure the CDS replication server:

   ```
   #mkdce cds_second
   ```

   See 4.1.6.1, "Replicating a CDS Server" on page 98 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
      ```

      See the remarks about the timing problem in 4.2, "Configuring DFS" on page 101 for reasons why this step is necessary.

   b. Configure the DFS client:

      ```
      #mkdfs dfs_cl
      ```

   c. Configure the the Fileset Database (FLDB):

      ```
      #mkdfs -s /.:/hosts/ev2 dfs_fldb
      ```

   d. Configure the DFS File server with the option to load the kernel extension:

      ```
      #mkdfs -s /.:/hosts/ev2 -e dfs_srv
      ```

   e. Configure the DFS replication server machine:

      ```
      #mkdfs -s /.:/hosts/ev2 dfs_repsrv
      ```

   f. Release the forced connection to the master security server:

      ```
      unset BIND_PE_SITE
      ```

   g. Create logical volumes as large as on *ev2*:

      ```
      #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/ev3 -aggr lfsroot
      #fts addsite -fileset usrbin.ft -server /.:/hosts/ev3 -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 machine *ev1*

1. Configure the core components:

   ```
   #mkdce -n itsc.austin.ibm.com -s ev2 sec_cl cds_cl dts_global
   ```

2. Configure the DFS components:

   ```
   #mkdfs dfs_cl
   ```

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
   ```

Configuring machine *ev4*

1. Configure the core components:

   ```
   #mkdce -n itsc.austin.ibm.com -s ev2 -c ev2 sec_cl cds_cl dts_local
   ```

2. Configure the DFS components:

   ```
   #mkdfs dfs_cl
   ```

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
   ```

3.2.4.3 Scenario Experiences

Same discussion as in scenario 5a. see 3.2.3.3, “Scenario Experiences” on page 72. Everything is slower as with X.25.

3.2.4.4 Special Issues

Same discussion as in scenario 5a. see 3.2.3.4, “Special Issues” on page 72.

3.2.4.5 Response Times

Everything works a bit slower than over X.25 (scenario 5a).

3.2.4.6 Performance discussion

See the discussion in scenario 5a, 3.2.3.6, “Performance Discussion” on page 72. Assuming that we have excluded the X.25 interface in scenario 5a, the SLIP works exactly the same as the X.25 environment. In both scenarios we rely on TCP/IP routing. SLIP is simply slower in our setup. This is a matter of modem speed and can be improved with faster modems.

3.2.4.7 Availability Discussion

Same discussion as in scenario 5a, 3.2.3.7, “Availability Discussion” on page 74.

3.2.5 Scenario 6: A Branch Connected with Two Links

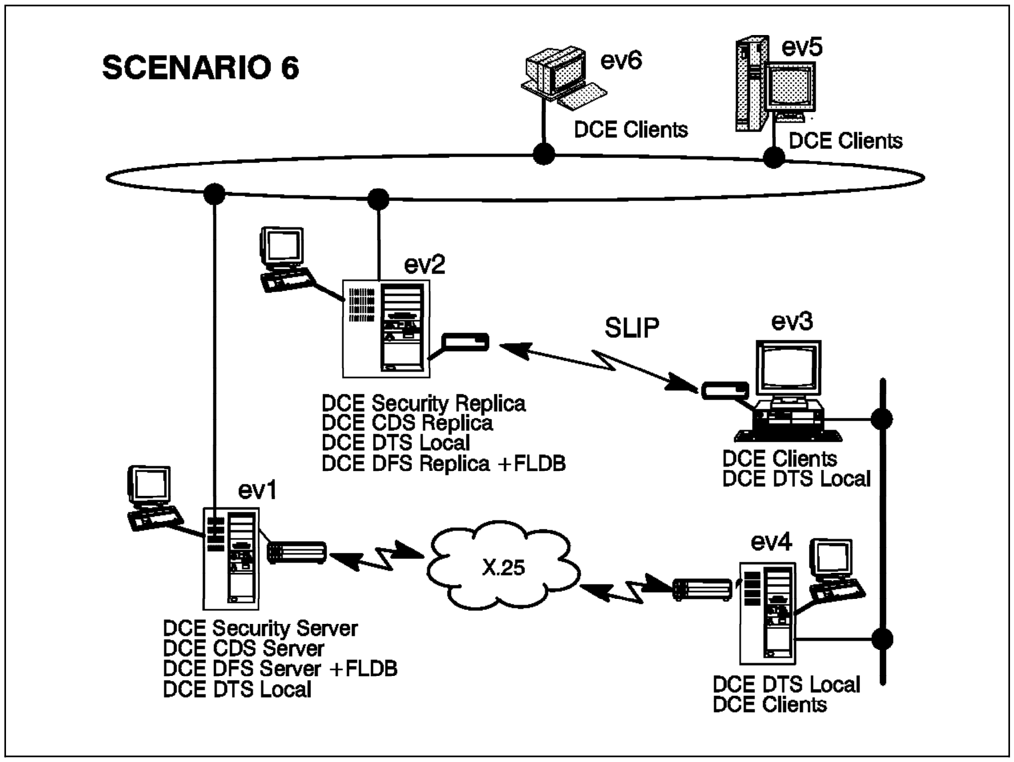


Figure 24. Scenario 6: A Branch Connected with Two Links

Due to 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 "export RPC_UNSUPPORTED_NETIFS=xt0:xt1: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 RPC 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. This is done by the command `cdscp defined cached server` This command is internally executed by `mkdce -c CDS_server_name`. 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 timeouts 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 timeout, before the next handle is tried, which again could be an X.25 binding handle.

The problem of avoiding timeouts 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. Multi-protocol routers are usually able to do this.

The simplest case of redundant network connection to a branch is shown in Figure 24 on page 81. 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 they are really independent from each other to minimize the chance that both links become unavailable at the same time.

3.2.5.1 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 5a, 3.2.3.6, "Performance Discussion" on page 72, 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.

3.2.5.2 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, we can decouple performance and availability issues to a great extent. We can limit our discussion about server replication to performance issues.

Chapter 4. Administering DCE Cells

We identified a list of tasks the administrator 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:

- Configuring a Cell

  This provides step-by-step instructions on how all the scenarios have been configured and guidelines on how to prepare the machines before a DCE installation.

- Configuring DFS

  DFS is well covered in the recently published ITSO publication *The Distributed File System (DFS) for AIX/6000*. This section gives a short step-by-step configuration guideline on how to set up a DFS server with a couple of filesets and a DCE client. It also discusses experiences with fileset replication and considerations about having the users′ home directories in DFS.

- 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 customers business is growing. 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, 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.

- DCE with HACMP/6000

  This section gives instructions on how to set up the security or CDS server in an HACMP/6000 cluster, to make them available for write access at any time.

4.1 Configuring a Cell

This section discusses the following topics:

- Preparing for DCE installation and configuration
- Installing DCE
- Configuring DCE core server and clients
- Starting and stopping servers
- Replicating the core servers

Chapter 3, "Implementing DCE Cells" on page 43 gives step-by-step instructions for configuration of specific scenarios.

4.1.1 Preparing for DCE Configuration

The purpose of this section is to guide you through all the necessary preparation steps that are required and the same 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

4.1.1.1 Preparing Disk Space

Installation and configuration of DCE servers and clients requires some reserved disk space which 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 at least 100MB allocated for paging space.

   ```
   #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 64MB for paging space. Since more than 70% is used we suggest you increase it. To increase both disks to 60MB each, we have to add 7 partitions of 4MB:

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

2. Disk space for /var/dce:

   The /var file system is used by the operating to store various files which can grow in size and number, such as the print spool and trace files. On the other hand DCE also has some files which use more and more disk space, for instance, core databases and credential files. It is important for AIX and DCE not to disturb 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 20MB 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:

   Creating this file system is helpful on a DFS client side to avoid getting stuck because of an incorrectly defined cache that could fill up the /var/dce file system or because of files underneath /var/dce using up space meant to be reserved for disk cache. The following example makes room for a 10MB cache on a 12MB 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. Disk space for /var/dce/rpc/socket:

   DCE uses a large number of inodes to create sockets. Creating this file system can be helpful.

```
#crfs -v jfs -g'rootvg' -a size='10000' -m'/var/dce/rpc/socket'\
 -A'yes' -p'rw'

#mount /var/dce/rpc/socket
```

Note

It was a bug in the beta code we were using that these zero size socket files were not cleaned up. Try it without creating this separate file system, but keep an eye on the number of these files.

5. 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.

4.1.1.2 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) 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).

4.1.1.3 Checking Network Routing

Before configuring DCE, make sure that all machines in your network communicate correctly with each other using TCP/IP protocol. You can use the `#ping` command to test all connections. 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 it 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.

4.1.1.4 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
xt0   576   <Link>                   3     0        3  0     0
xt0   576   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 xt0, 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 did 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 1.3 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 like this before you configure any server:

```
 ..
RPC_UNSUPPORTED_NETIFS=xt0
export RPC_UNSUPPORTED_NETIFS
 ..
```

If you exclude xt0 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.

4.1.1.5 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
```

4.1.2 Installing the DCE Code

Installing DCE is the procedure of loading the software on to 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.

If you have not done this yet, go back to 4.1.1, “Preparing for DCE Configuration” on page 84.

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 to backup all your DCE databases prior to the upgrade. See 4.4, “Backup/Restore and Other Housekeeping Tasks” on page 142 for details.

4.1.3 Configuring the Core Services

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

1. Configure the Security server machine
2. Configure the CDS server machine
3. Configure the DTS server machine

The following steps to configure a cell are common to all scenarios we have defined in Chapter 3, “Implementing DCE Cells” on page 43. We use the cell name *itsc.austin.ibm.com* for all tasks performed.

4.1.3.1 Configuring the Security Service

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]
   PRINCIPALS Lowest possible UNIX ID              [100]
   PRINCIPALS Lowest possible UNIX ID              [100]
   GROUPS Lowest possible UNIX ID                  [100]
   GROUPS Lowest possible UNIX ID                  [100]
   ORGANIZATIONS Lowest possible UNIX ID           [100]
   ORGANIZATION Lowest possible UNIX ID            [100]
   MAXIMUM possible UNIX ID                        [32767]
   MAXIMUM possible UNIX ID                        [32767]
```

```
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
```

When you configure the security server *sec_srv*, the security client *sec_cl* is automatically configured for you. At this point you can start to configure a CDS server machine. 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.

4.1.3.2 Configuring the CDS Server

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 = mkdcecds)
```

```
                      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]<<--Fill this.
* Cell ADMINISTRATOR's account                    [cell_admin]
* LAN PROFILE                                     [/.:/lan-profile]
```

```
Enter password for DCE account cell_admin:

Password must be changed!
Configuring Initial CDS Server (cds_srv)...
Configuring CDS Clerk (cds_cl)...
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)
```

The following command has the same effect:

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

When you configure the CDS server machine, the CDS client or, more accurately, the CDS clerk is automatically configured.

4.1.3.3 Configuring the DTS Server

Call SMIT and follow the indicated path or call SMIT with the fastpath name:

```
#smitty dce
   -> Configure DCE/DFS
      -> 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]
```

```
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)
```

The following command has the same effect:

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

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

4.1.3.4 Configuring Multiple Servers at One Time

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
```

4.1.4 Configuring the DCE Clients

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

In AIX DCE 1.3 we have the ability to split the configuration: 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 features 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 by themselves or to give away their password. As long as all machines are in the same (trusted) LAN you can remotely login to each client.

- 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 want to discuss our experiences with the split configuration.

4.1.4.1 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_ev4]
      The following fields are used
      ONLY if a DFS client is configured
      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_ev4
           Modifying acls on hosts/dce_ev4/self
           Modifying acls on hosts/dce_ev4/cds-clerk
           Modifying acls on hosts/dce_ev4/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_ev4 -s ev1 all_cl
```

At this point all DCE core clients are configured. See 4.2, "Configuring DFS" on page 101 to understand how to configure a DFS server machine and a DFS client machine.

4.1.4.2 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?

Steps to be performed by cell_admin:

The following are the tasks to do 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 *ev4*:

```
#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_ev4]
 * Client Machine IDENTIFIER             [ev4]
 * LAN PROFILE                           [/.../itsc.austin.ibm.com/lan-profile]
```

```
Enter password for DCE account cell_admin:
Configuring Security Client (sec_cl) for dce_host dce_ev4 on
  machine ev4 ...
Password must be changed!
Completed admin configuration of Security Client (sec_cl) for
  dce_host dce_ev4 on machine ev4
Configuring Security Client (sec_cl) for dce_host dce_ev4 on
  machine ev4 ...
Completed admin configuration of Security Client (sec_cl) for
  dce_host dce_ev4 on machine ev4
Configuring CDS Clerk (cds_cl) for dce_host dce_ev4 on
  machine ev4 ...
           Modifying acls on hosts/dce_ev4
           Modifying acls on hosts/dce_ev4/self
           Modifying acls on hosts/dce_ev4/cds-clerk
           Modifying acls on hosts/dce_ev4/profile
           Modifying acls on /.:/lan-profile

Completed admin configuration of CDS Clerk (cds_cl) for
  dce_host dce_ev4 on machine ev4

Cell administrator's portion of client configuration has completed
  successfully. Root administrator for ev4 should now
  complete the client configuration on that machine.
          Press Enter to continue
```

You must specify two machine names, which is new with AIX DCE 1.3:

```
  ...
  Client Machine DCE_HOSTNAME          [dce_ev4]
* Client Machine IDENTIFIER            [ev4]
  ...
```

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 to see 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 5.1.2, “Split Configuration” on page 220.

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_ev4 -i ev4 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.

Steps to be performed by the local system administrator:

The following shows how the local system administrator needs to configure his 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 dfs_cl]
* SECURITY Server                            [ev1]
  CDS Server (If in a separate network)      [ev1]
* Client Machine DCE_HOSTNAME                [dce_ev4]
     The following fields are used
     ONLY if a DFS client is configured
     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
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
          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_ev4** all_cl

4.1.4.3 Experience with Split Configuration

The new feature is easy to use and understand. It is very useful to administer DCE client machine configurations in a large DCE cell. Owner of client workstations could 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* from the local hostname for the DCE hostname and tried to authenticate as principal hosts/ev8/self which does not exist.

We recommend to always specify the DCE hostname for both sides of the split configuration.

4.1.5 Starting and Stopping DCE

You can start DCE manually at your convenience or do it automatically at machine boot time:

1. You can manually start and stop DCE processes at any time:

   For instance the CDS server and client on a CDS server machine:

   ```
   #/etc/dce.clean cds
   #/etc/rc.dce cds
   ```

   Or all DCE services, including DFS:

```
#/etc/dce.clean
#/etc/rc.dce
```

The dfsd process cannot be stopped, though. You must reboot the machine, if you need to restart dfsd.

2. You can also start DCE automatically at machine boot time.

   You must put the /etc/rc.dce script file into the /etc/inittab file. Following are the lines you need to add in the inittab file:

```
 ...
srcmstr:2:respawn:/etc/srcmstr          # System Resource Controller
rctcpip:2:wait:/etc/rc.tcpip > /dev/console 2>&1 # Start TCP/IP daemons
rcdce:2:wait:/etc/dce/rc.dce  core > /dev/console 2>&1:
rcdfs:2:wait:/etc/dce/rc.dfs  > /dev/console 2>&1
rcnfs:2:wait:/etc/rc.nfs > /dev/console 2>&1 # Start NFS Daemons
rcdfsnfs:2:wait:/etc/dce/rc.dfsnfs > /dev/console 2>&1
 ...
```

   You can put /etc/rc.dfs and /etc/rc.dfsnfs in the inittab file if you want to start them automatically.

4.1.6 Replicating DCE Core Services

DCE core services that can be replicated are CDS server, security server and DTS server. For the DTS server the term replication is not appropriate, we would rather use the term redundancy. For DFS replication see 4.2.3, “Replicating DFS Server” on page 107.

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. The security server replicates its entire registry database as a whole, whereas the CDS database can be distributed.

In this section we will discuss:

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

According to Figure 18 on page 48 we replicate the CDS server and the security server on machine *ev4*.

4.1.6.1 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 *ev4*:

```
#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.ccom]
* SECURITY Server                                    [ev1]
  Initial CDS Server (If in a separate network)      []
* Cell ADMINISTRATOR's account                       [cell_admin]
* LAN PROFILE                                        [/.:/lan-profile]
```

```
Enter password for DCE account cell_admin:

Password must be changed!
Configuring Additional CDS Server (cds_second)...
           Modifying acls on ev4_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 on this machine (*ev4*) an additional CDS server. The component is called *cds_second*.

The following command has the same effect:

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

At this point, even if you have configured the machine to become a CDS replication server, you have not achieved replication of any CDS data. You have to explicitly specify, which directory to replicate. We have provided a shell script `copy_CH` which replicates all CDS directories to the new CDS replication server. This script is on the diskette associated with this publication. It is well documented and can also be consulted to see how a directory can be replicated.

4.1.6.2 Replicating the Security Server

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
            -> 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]
  PRINCIPALS Lowest possible UNIX ID                [100]
* REPLICA name                                      [ev4]
  GROUPS Lowest possible UNIX ID                    [100]
  ORGANIZATIONS Lowest possible UNIX ID             [100]
* SECURITY Server                                   [ev1]
* LAN PROFILE                                       [/.:/lan-profile]
  MAXIMUM possible UNIX ID                          [32767]
```

```
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
cds_second   COMPLETE   Additional CDS Server
dts_cl       COMPLETE   DTS Clerk
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
sec_srv      COMPLETE   Security Server (Replica)
```

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 you try to issue the command: `#sec_admin` you will see that the default replica on the machine is *ev4*.

```
Default replica:  /.../itsc.austin.ibm.com/subsys/dce/sec/ev4
Default cell:     /.../itsc.austin.ibm.com
```

If the security master server goes down, you can continue to login into the cell, but it may take a long time. Response time depends on which machine you issue the `dce_login` command. For example on *ev4* you can login immediately into the cell, because clients always use the local server. Other machines in the cell, which are bound to the master security server, will experience a timeout before they look for an alternative to get tickets from.

4.2 Configuring DFS

The Distributed File System (DFS) is one of the first real DCE applications, it is not considered a core service. See 1.3.3.6, “Distributed File System” on page 16 in this publication for a short description on DFS.

The Distributed File System (DFS) for AIX/6000 has more details on DFS. This section gives a short step-by-step configuration guideline on how to set up a DFS server with a couple of filesets and a DCE client. It also discusses experiences with fileset replication, a feature which was not available before, and considerations about having the users′ home directories in DFS.

The following topics are discussed in this section:

- Configuring a DFS server
- Configuring a DFS client
- Configuring a DFS replication server
- Defining home directories in DFS

Timing problem

If a security replica server is installed in the cell, calls to the security service are evenly split between the master and replica servers. When a registry account is added and the local keytab entry is generated in the same command, there is a timing problem, if the connection goes to a replica security server. You′ll get:

```
?(rgy_edit) Unable to add key - Registry object not found (dce / sec)
?(rgy_edit) Unable to add key - Requested key is unavailable (dce / sec)
```

The addition of the keytab entry comes too quickly after the creation of the account. The account is not found yet on the security replica. `mkdfs` often fails for this reason. If this happens, you must delete the principal and the CDS entries, and then you *must* reboot the system, otherwise the `bosserver` will fail to start.

To circumvent this problem, make sure the master security server is the first entry in the file /var/dce/security/pe_site and export the BIND_PE_SITE variable. After all DFS servers are installed use `unset BIND_PE_SITE` to release the forced binding to the master security server.

4.2.1 Configuring a DFS Server

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

Before starting to configure the DFS server make sure that all DCE core services are configured. Below is a short summary description of the different DFS components 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 servers. A DFS domain is the set of machines that is controlled by one SCM, respectively by the same lists of DFS administrators. There can be multiple domains in a DCE cell.

- Fileset database machine

  A Fileset database machine stores the Fileset Location Database (FLDB). The purpose of the FLDB is to take a pathname for a file that is located in the DFS namespace and determine the location of the file server that houses that file.

- File server machine

  A file server machine is used to store and export DCE 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. In our example we will use LFS file systems.

After having configured these servers, we can export the root.dfs fileset and other filesets. Below are the steps to make available the DFS filespace. If you plan to replicate a fileset, you can also replicate this fileset immediately after having created the read-write fileset, which would prevent any DFS clients from obtaining access to the read-write fileset via a regular mount point. See 4.2.3, “Replicating DFS Server” on page 107 for a discussion on when to replicate filesets and to create mount points.

4.2.1.1 Configuring an System Control Machine

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                                     [/.:/lan-profile]
```

```
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 has the same effect:

```
#mkdfs dfs_scm
```

4.2.1.2 Configuring a Fileset Location Database Machine

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                                        [/.:/lan-profile]
```

```
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.
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 has the same effect:

```
#mkdfs dfs_fldb
```

4.2.1.3 Configuring the DFS File Server Machine

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                                         [/.:/lan-profile]
```

```
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 has the same effect:

```
#mkdfs -e dfs_srv
```

4.2.1.4 Configuring a DFS root Fileset

1. Prepare an aggregate to house the root.dfs fileset:

   Note that this aggregate need not be very large. Creating a logical volume with one block (4MB) 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
     ```

   - Make the /dev/lfsroot logical volume a 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 one JFS.

2. Create the DFS root directory

   This step performs multiple steps at one time. It exports the aggregrate /dev/lfsroot, creates the root.dfs fileset, and mounts it at the DFS junction.

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

```
        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. Before being able to access the DFS namespace, you have to configure a DFS client. See 4.2.2, “Configuring a DFS Client” on page 106 to understand how to configure a DFS client. Or you can just type the following command to configure the DFS client machine:

```
#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
```

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 yet to wait for. When you type the following command for the first time:

```
#cd /:
```

you will get a message after a while that the DFS server is in TSR (Token State Recovery) mode. Normally after a minute or two, you can access the DFS space.

4.2.1.5 Adding Another Fileset

This step is optional. We want to add another fileset to the DFS filespace. We can add many filesets within a single LFS aggregate. However, we suggest not adding another fileset to the rootlfs aggregate. We suggest creating another aggregate for other filesets. Adding other filesets can be done on every DFS Server machine.

Suppose we want to add another fileset usrbin.ft:

1. Create a logical volume *usrbin* with five blocks:

   ```
   #mklv -t lfs -y usrbin rootvg 5
   ```

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

   ```
   #newaggr -aggregate /dev/usrbin -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/rusrbin: Marked as not a BSD file system any more.
   Done.  /dev/rusrbin is now an Episode aggregate.
   ```

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

```
#newaggr -aggregate /dev/usrbin -bl 8192 -fr 1024 -overwrite
```

3. Export the aggregate:

```
#mkdfslfs -d /dev/usrbin -n usrbin
```

4. Create a fileset with a mount point:

```
#mkdfslfs -f usrbin.ft -m /:/usrbin -n usrbin

        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      usrbin 0:00:00 hosts/ev1      <nil>
Fileset 0,,4 created on aggregate usrbin of /.:/hosts/ev1
```

If you forgot to login to DCE and the following message appears:

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

you should login as cell_admin and enter the following command:

```
#fts crmount -dir /:/usrbin -fileset usrbin.ft
```

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

```
usrbin.ft
        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      usrbin 0:00:00 hosts/ev1      <nil>

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      root.dfs 0:00:00 hosts/ev1      <nil>
----------------------
Total FLDB entries that were successfully enumerated: 2
```

4.2.2 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 4.1.1.1, “Preparing Disk Space” on page 84. The separate file system for the DFS cache has to be mounted before you begin. If you want to choose another cache size than 10MB (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
```

Before being able to access the /: directory, you will have to wait some time. Then try this command:

```
#cd /:
```

You might experience the following error message:

```
/bin/ksh: /::  does not exist
```

You will have to wait a moment and try again. You can try the following command to see if the fileset(s) are available:

```
#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       lfs.root 0:00:00 hosts/ev1  <nil>

usrbin
        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       usrbinlv 0:00:00 hosts/ev1      <nil>

----------------------
Total FLDB entries that were successfully enumerated: 2 (0 failed; 0
wrong aggr type)
```

If this message is displayed, it means that everything is alright.

4.2.3 Replicating DFS Server

Before reading this section, we recommend you read the discussion about DFS replication in 5.2, “DFS Replication” on page 224:

- 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 3, “Implementing DCE Cells” on page 43. 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 18 on page 48. 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.

4.2.3.1 Setting Up Replication

Before starting a fileset replication, we must have previously replicated the root.dfs fileset on the same machine that physically houses this root.dfs fileset.

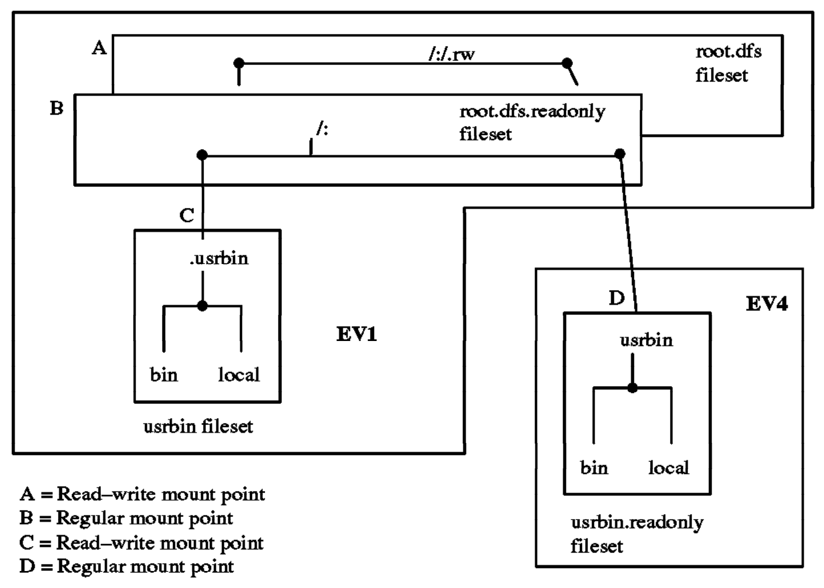


Figure 25. DFS Replication Fileset

In Figure 25:

- /:/.rw is a read-write mount point for the root.dfs fileset. The directory is physically housed on *ev1*.
- /: is a regular mount point or read-only directory for the DFS root directory. This directory is housed on *ev1*.
- /:/.usrbin is a read-write mount point for the usrbin fileset, which is physically housed on *ev1*. This pathname accesses the read-write fileset. So does the pathname /:/.rw/usrbin. We want to replicate this source fileset on *ev4*.
- /:/usrbin is a read-only directory, which is physically housed on *ev4*. The regular mount point accesses the read-only fileset.

The following two sections give you the steps to perform a replication. We start by replicating the root.dfs fileset and then we can replicate another fileset.

4.2.3.2 Setting up Replication for the root.dfs Fileset

The root.dfs fileset is the fileset that houses the root directory for DFS.

We assume that root.dfs is properly configured on an LFS fileset. See 4.2.1, "Configuring a DFS Server" on page 101 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. Configure and start the replication server:

```
#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 -s /.:/hosts/ev1 dfs_repsrv
```

2. Create a 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
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*.

4. Define the same machine as a replication site:

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

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

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

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
```

At this point, you can start to replicate other filesets. In this example we want to replicate the usrbin fileset on *ev4*.

Verify the status of the fileset in the FLDB:

```
#fts lsfldb

root.dfs
        readWrite   ID 0,,1  valid
        readOnly    ID 0,,3  valid
        backup      ID 0,,4  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 version fileset is 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.

4.2.3.3 Setting Up Replication for Another Fileset

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

For all types of replication (release or schedule), you must have previously:

- Installed and configured file server machines to hold the replicas. See in 4.2.1, "Configuring a DFS Server" on page 101 how to configure a DFS server.
- Made sure that the replication server (repserver process) is configured and running.

  This command configures and starts the replication server:

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

  This command can also be accessed from SMIT as was done on *ev1*.

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

We assume in this example, that *ev1* is the system control machine. According to Figure 25 on page 108, we assume we have a fileset called *usrbin* that we want to replicate on another machine called *ev4*. The usrbin fileset is physical ly housed on *ev1*. We assume also the regular mount point of the usrbin fileset is /:/usrbin before and after replication.

High availability of read-only filesets

In order to get highly available read access to the fileset usrbin.ft.readonly, all filesets above it in the fileset hierarchy need to be replicated also. In other words, root.dfs should be replicated on *ev4* to make sure /:/usrbin can be accessed, if *ev1* breaks.

Configuring fileset using released replication:

On the server system that has the read-write fileset you must do the following tasks for release replication:

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

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

   b. Define the same machine as a replication site:

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

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

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

2. On a target machine (*ev4*):
   a. Configure this machine as a DFS server, see 4.2.1, "Configuring a DFS Server" on page 101. Or you can 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                    [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]
```

This command has the same effect:

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

b. Configure this machine as a replication server:

   This command configures and starts the replication server:

```
#mkdfs -s /.:/hosts/ev1 dfs_repsrv
```

c. Create an aggregate large enough to house the fileset:

   Suppose the aggregate *usrbin* has five blocks of 4MB. We have to provide at least that same size on the target machine.

   Create a logical volume as large as on *ev1*:

```
#mklv -t lfs -y usrbin rootvg 5
```

d. Create an aggregate on /dev/usrbin:

```
#newaggr -aggreg /dev/usrbin -bl 8192 -fr 1024 -overwrite
```

e. Export the aggregate:

```
#mkdfslfs -d /dev/usrbin -n usrbin
```

f. Define the new replication site:

```
#fts addsite -fileset usrbin.ft -server /.:/hosts/ev4 -aggr usrbin
```

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

```
#fts release -fileset usrbin.ft
Released fileset usrbin.ft successfully
```

   Since the /:/usrbin directory with the regular mount point existed before, we need not force an update of the /: parent directory at this point.

h. Leave the /:/usrbin 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 usrbin.ft.

```
cd /:
```

i. Force the local cache manager to read the new fileset information:

```
#cm checkfilesets
```

j. Check whether you can create a file in /:/usrbin now:

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

You are not 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
```

k. Update the read-only copy of root.dfs to make the /:/.usrbin directory available:

```
#fts rel root.dfs
```

l. 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.

Configuring fileset using scheduled replication:

1. On *ev1*, which physically houses the fileset:

   a. Set the replication parameters for scheduled replication:

   ```
   #fts setrepinfo -fileset usrbin.ft -sched
   ```

   You do not need to use the command `fts addsite` on this machine, but you need to use it on the target machine. You may use it, however, if you want to create a read-only copy on this primary machine.

2. On a target machine (ev4):

   These are the steps to be done on each target machine:

   a. Configure this machine as a DFS server as you did above for release replication:

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

   b. Configure this machine as a replication server:

   This command configures and starts the replication server:

   ```
   #mkdfs -s /.:/hosts/ev1 dfs_repsrv
   ```

   c. Create an aggregate large enough to house the fileset:

   Assume the aggregate *usrbin* has five blocks of 4MB. Create a logical volume as large as on *ev1*:

   ```
   #mklv -t lfs -y usrbin rootvg 5
   ```

   d. Create an aggregate on /dev/usrbin:

   ```
   #newaggr -aggreg /dev/usrbin -bl 8192 -fr 1024 -overwrite
   ```

   e. Export the aggregate:

   ```
   #mkdfslfs -d /dev/usrbin -n usrbin
   ```

   f. Define the new replication site:

   ```
   #fts addsite -fileset usrbin.ft -server /.:/hosts/ev4 -aggr usrbin
   ```

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

```
#fts update -fileset usrbin.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 /:/usrbin directory with the regular mount point existed before, we need not force an update of the /: parent directory at this point.

h. Leave the /:/usrbin 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 usrbin.ft.

```
cd /:
```

i. Force the local cache manager to read the new fileset information:

```
#cm checkfilesets
```

j. Check whether you can create a file in /:/usrbin now:

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

You are not 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
```

k. Update the read-only copy of root.dfs to make the /:/.usrbin directory available:

```
#fts rel root.dfs
```

l. 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.

4.2.3.4 Check Your Work for Accuracy

You can use these 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>

usrbin.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    usrbin  0:00:00 hosts/ev1      <nil>
ev4                RO       usrbin  0:00:00 hosts/ev4      <nil>
```

The *readOnly* fileset is now marked *valid*.

- Access to DFS filespace:

  `#dce_login cell_admin mypasswd`

  `#cd /:/usrbin`

4.2.4 Defining Home Directories in DFS

Defining the home directory for DCE users is not trivial, because at the time the user logs in to AIX, they have no network credentials yet, in other words, they are not DCE authenticated yet. So, they do not have the necessary access permissions to enter into their home directory.

Let′s assume that user joe is to have a home directory in DFS, namely /:/dfshome/joe. The first difficulty is to define this home directory in the /etc/passwd file. Since the colon (:) is used as a field separator, we must enter the global directory name:

```
# cat /etc/passwd | grep joe
joe:!:302:1::/.../itsc.austin.ibm.com/fs/dfshome/joe:/bin/ksh
```

If a user logs in to AIX and does not have a valid home directory, their home directory is set to /usr/guest or /home/guest. However, they do not have sufficient permission there to actually use that directory.

We can use the .profile in /home/guest to take the DCE users through to a DCE login. At the same time we can reassign the home directory:

```
$ pg /home/guest/.profile
export EDITOR=vi
# Extract the Home directory name:
export HOME=`cat /etc/passwd | grep $USER | cut -f6 -d:`
export ENV=/$HOME/.kshrc
# DCE Login
echo ″\nLogin to DCE --> \c″
dce_login $USER
```

Set permission such that all users can run the .profile:

```
$ chmod o+rx .profile
```

User joe gets immediately to the dce_login command, when he logs in to AIX. Since a new Korn shell is started after a successful dce_login, we can use the .kshrc startup file to issue a cd command, so that joe′s current directory is his DFS home directory.

```
# cat /.../itsc.austin.ibm.com/fs/dfshome/joe/.kshrc
cd
```

If user joe now logs in, he has to specify his password twice, once for AIX and once for DCE. If that operation is successful, he is in DCE and his home directory in DFS is his current directory:

```
IBM AIX Version 3 for RISC System/6000
(C) Copyrights by IBM and by others 1982, 1991.
login: joe
Enter password for joe: secret
Standard UNIX authentication successful.
******************************************************************************
*  Welcome to IBM AIX Version 3.2!                                           *
******************************************************************************
1 unsuccessful login attempt since last login
Last unsuccessful login: Thu Sep 22 13:23:49 CDT 1994 on pts/6
Last login: Thu Sep 22 13:16:01 CDT 1994 on hft/0
3004-614 Unable to change directory to "/.../itsc.austin.ibm.com/fs/dfshome/joe".
        You are in "/home/guest" instead.

Login to DCE --> Enter Password: secret
$ pwd
/.../itsc.austin.ibm.com/fs/dfshome/joe
$
```

Please note that this procedure is only necessary, if you are using the standard dce_login command.

If you use Single Login/6000 (see 5.4, “Single Login/6000” on page 235) and our user management tool as described in 5.5, “User (and ACL) Management” on page 242, defining a home directory in DFS is much easier.

- The DFS home directory is defined in the DCE registry account
- Use our user management tool to define the DCE user and account
- It can be defined as /:/dfshome/joe
- Both the registry and our user management tools support the notation with the colon (:)
- Single Login/6000 uses the values in the DCE registry account in contrast to dce_login, which ignores them and uses the home directory and initial program of the local /etc/passwd file

4.3 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 is 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 document.

4.3.1 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 more information which is important for migration of users and files see the following sections:

- 5.5, “User (and ACL) Management” on page 242
- 4.7.3, “Migrating NFS Files to DCE/DFS” on page 208
- 2.4, “Planning the User Namespace” on page 33

The following list describes a general procedure to move users from one cell to a new cell. The user management tools would have to be extended to also support groups and maybe Single Login/6000.

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 the 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. Backup 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 deleted UDFs/GDFs. 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 4.3.4, “Moving Services Within the Cell” on page 129 on how to achieve this.

10. Install the DCE and DFS servers in the new cell.

11. Move the UDFs 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 no change.

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 Single Login/6000, 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, such 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 Single Login/6000, 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.

4.3.2 Joining Cells

For a general procedure on how to join cells or move parts of an existing cell to another cell, see 4.3.1, "Splitting Cells" on page 117. 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, which you had originally designed for unique UIDs and principal names across multiple cells as discussed in 2.4, "Planning the User Namespace" on page 33, 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 all UDFs 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 users:

1. Find a new user name and/or UID
2. Recursively change their DFS file ownerships to the new UID
3. Find file or directory names that contain their old user name and change them to the new 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.

4.3.3 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 for the involved machines such as LAN interface configuration, name server entry, and routing. As a summary of the considerations presented in the following section, 4.3.3.1, "RPC Binding Information or CDS Towers" on page 120, we can say 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 4.4.5, "Managing Caches on Client Machines" on page 156:

`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 will describe:

- Binding handles overview
- Description of each shell script
- Generalized procedure on how an IP address change is performed with the help of our scripts
- Our experiences

4.3.3.1 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 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.

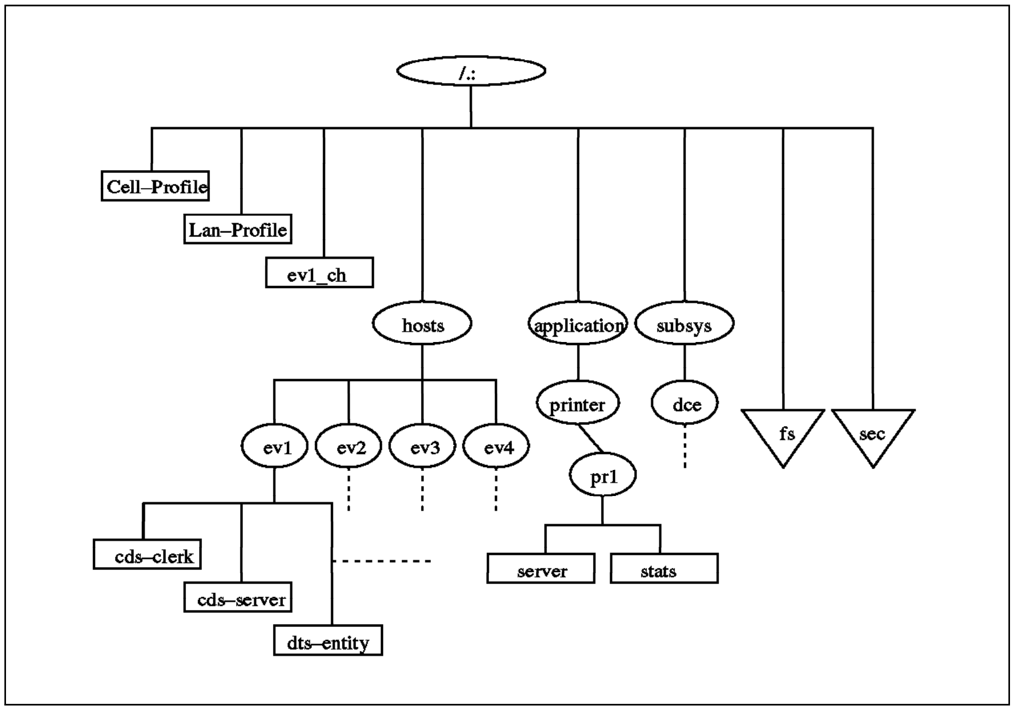


Figure 26. Extract of a CDS Namespace

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, they memorize (cache) it, so they do not have to call the operator again for the same information. If the number changes, we have to tell that to each client which memorized it.

We need to change every occurrence of an IP address in CDS and on all clients′ caches.

4.3.3.2 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
```

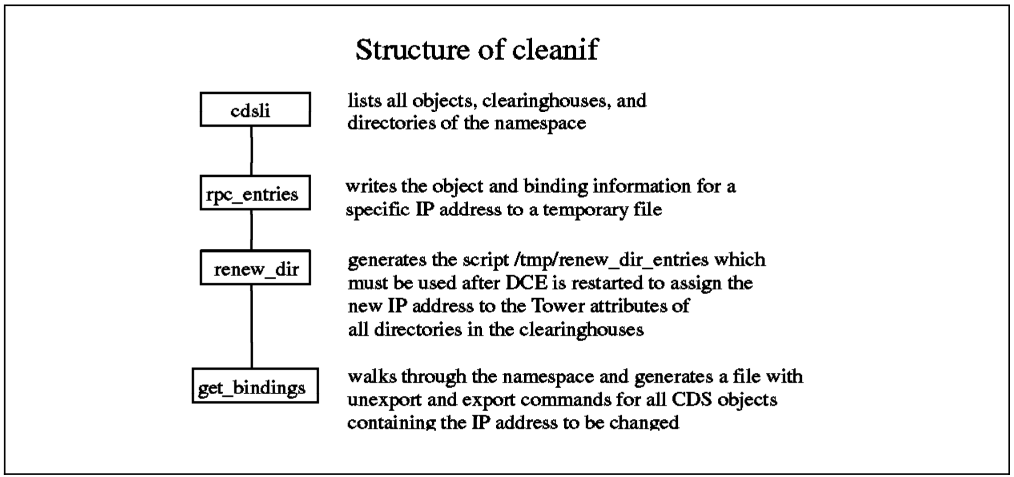

Structure of cleanif
cdsli
lists all objects, clearinghouses, and directories of the namespace
rpc_entries
writes the object and binding information for a specific IP address to a temporary file
renew_dir
generates the script /tmp/renew_dir_entries which must be used after DCE is restarted to assign the new IP address to the Tower attributes of all directories in the clearinghouses
get_bindings
walks through the namespace and generates a file with unexport and export commands for all CDS objects containing the IP address to be changed

Figure 27. 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 which contain binding handles with that IP address. Since `unexport` can only remove an entire interface, which might also contain other 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 it creates another shell script called /tmp/renew_dir_entries. This procedure is explained in 4.3.3.4, "renew_dir_entries Procedure" 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 which 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!

4.3.3.3 cleanup_ip Procedure

The script cleanup_ip takes the rpccp input file previously generated by cleanif and executes all the 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>
```

4.3.3.4 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, which 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 in all directories of that server has to be changed.

renew_dir_entries is a shell script 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 [ 0 -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.

4.3.3.5 Generalized Procedure to Change an IP Address

Figure 28 on page 125 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 what type of DCE/DFS server has been changed.

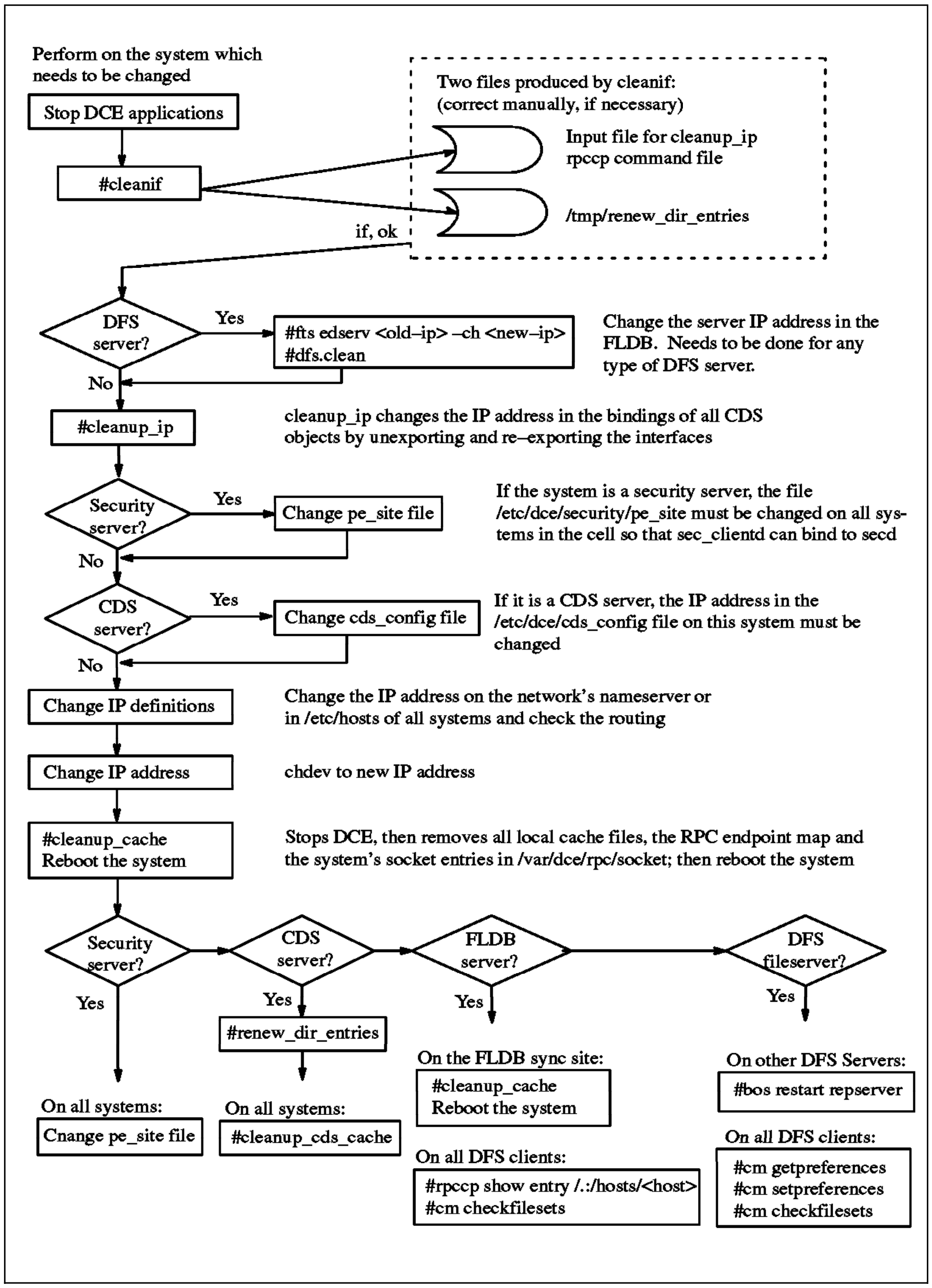

Perform on the system which needs to be changed
Stop DCE applications
#cleanif
Two files produced by cleanif: (correct manually, if necessary)
Input file for cleanup_ip rpccp command file
/tmp/renew_dir_entries
if, ok
DFS server?
Yes
No
#fts edserv <old–ip> –ch <new–ip> #dfs.clean
Change the server IP address in the FLDB. Needs to be done for any type of DFS server.
#cleanup_ip
cleanup_ip changes the IP address in the bindings of all CDS objects by unexporting and re–exporting the interfaces
Security server?
Change pe_site file
If the system is a security server, the file /etc/dce/security/pe_site must be changed on all systems in the cell so that sec_clientd can bind to secd
CDS server?
Change cds_config file
If it is a CDS server, the IP address in the /etc/dce/cds_config file on this system must be changed
Change IP definitions
Change the IP address on the network's nameserver or in /etc/hosts of all systems and check the routing
Change IP address
chdev to new IP address
#cleanup_cache Reboot the system
Stops DCE, then removes all local cache files, the RPC endpoint map and the system's socket entries in /var/dce/rpc/socket; then reboot the system
Security server?
CDS server?
FLDB server?
DFS fileserver?
Yes
#renew_dir_entries
On all systems:
Cnange pe_site file
On all systems:
#cleanup_cds_cache
On the FLDB sync site:
#cleanup_cache Reboot the system
On all DFS clients:
#rpccp show entry /.:/hosts/<host> #cm checkfilesets
On other DFS Servers:
#bos restart repserver
On all DFS clients:
#cm getpreferences #cm setpreferences #cm checkfilesets

Figure 28. Generalized Workflow Description to Change an IP Address

4.3.3.6 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 has actually 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.

Depending on what type of server is running on the system of which the IP address is changed, some or all of the following additional steps are necessary as outlined also in Figure 28 on page 125:

DCE client only:

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 cash 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.

DCE security server:

The /etc/dce/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` to update the pe_site file. Otherwise, for example, if you have changed your IP address from 9.3.1.68 to 9.3.1.120:

```
/.../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[]
```

Change the above entries with a text editor to:

```
/.../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[]
```

Applications might have security server bindings cached. In particular, long running applications which automatically renew their ticket tend to cache a binding handle. For instance sec_clientd is one. It keeps the machine principal valid. When sec_clientd is stopped and restarted with the following commands, it destroys all existing credentials:

```
# dce.clean sec_clientd
# sec_clientd -purge
```

One of the next commands should also be executed to refresh the clerk cache entry for the security server, depending on whether it was the master or a replica server:

```
# rpccp show entry /.:/subsys/dce/sec/master -u
# rpccp show entry /.:/subsys/dce/sec/rep<number> -u
```

If you do not know which other applications behave like sec_clientd, stop all applications and run `cleanup_cache` to destroy credentials and cache entries; see 4.4.5, “Managing Caches on Client Machines” on page 156.

CDS server:

Follow the procedure. We have tested it 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 4.3.3.4, “renew_dir_entries Procedure” on page 123). However, check the contents of renew_dir_entries, before you execute it, to be sure the replica sets are correctly generated.

Do not forget to change the IP address in the file /etc/dce/cds_config.

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.

DFS FLDB:

The IP address of each FLDB server is stored in the FLDB. Change this address and regularly stop DFS on the old address to allow for correct DFS token management before you run cleanup_ip.

Also check which of the FLDB servers is the sync site:

```
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 his command for each FLDB server defined in the /.:/fs RPC group. If another FLDB server is the sync site, you must run cleanup_cache and reboot that server as well. Otherwise it keeps the old IP address in the FLDB and ubik tries to synchronize the FLDB on the old address. Just restarting the flserver on the sync site was not sufficient when we tested. 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:

```
# 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 timeouts on the DFS clients, you may want to refresh this CDS entry on all DFS clients:

```
# 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 timeout, when the old binding information is tried.

Run the `cm check` command to refresh the DFS clients′ binding information to the FLDB and the 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. 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:

```
# 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 which did not restart 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, 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 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.

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 much application dependent.

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.

4.3.4 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

4.3.4.1 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 numeric 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 allow 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. It 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 local RPC mappings for that application, if the server is not going to be rebooted; a well behaving application does that by unregistering itself from rpcd
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 timeouts:
   - Either the entire cache with the procedures described in 4.4.5, “Managing Caches on Client Machines” on page 156
   - Or force the CDS client to read the CDS object from CDS rather than from the local cache with:

     ```
     rpccp show entry <entry_name> -u
     ```

4.3.4.2 DFS Services

DFS has various machine roles which can be on the same or different machines. We discuss them separately. For more details consult the ITSO publication *The Distributed File System (DFS) for AIX/6000*.

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:

```
# 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 read-write filesets 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 4.2.3, “Replicating DFS Server” on page 107 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 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 which 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 according 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 adminstrative 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:

   ```
   # bos stop -s <scm_client_system> -p upclient.scm
   # bos delete -s <scm_client_system> -p upclient.scm

   # bos create -s <scm_client_system> -p upclient.scm -type simple -cmd '/op
   t/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 /o
   pt/dcelocal/var/dfs/admin.bak'
   ```

5. On the old SCM site, delete the upserver process:

   ```
   # bos stop -s /.:/hosts/<old_scm> -p upserver
   # bos delete -s /.:/hosts/<old_scm> -p upserver
   ```

   Then add an upclient.scm entry as you did in the previous step.

6. Check with the following command:

   ```
   # bos status /.:/hosts/<scm_client_system> upclient.scm -long
   ```

7. Delete the SCM machine on the old site:

   `rmdfs dfs_scm`

   However, if you have other DFS servers running, you will destroy adminstrative lists with this step and it will take a while until they are recreated by the upclient process.

4.3.4.3 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 want to 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 directory
- Clearinghouse
- CDS server

After all these changes a refresh of all CDS clerk caches is useful to avoid timeouts because a client still tries to access outdated information. The clients become aware of the change through advertisements of 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 add the cached server definitions.

To list, delete, or add the cached server definition on *ev4*, which is connected via X.25, you use the following commands:

```
cdscp show cached server ev*
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 4.4.5, “Managing Caches on Client Machines” on page 156.

Listing Directories: The IBM provided option is `cdsli -rd`, which lists all directories in the namespace. With the OSF provided command you can list the directories that are present in a certain clearinghouse:

```
# cdscp show object /.:/ev1_ch | grep Name | cut -f2 -d=
```

Finding all Replicas and the Master: The following command looks up a certain directory and prints which clearinghouses contain the master and read-only copies of that directory:

```
# 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 command provides a more direct method:

```
# cdscp show replica /.:/hosts clearinghouse /.:/ev1_ch  | grep \
CDS_ReplicaType | cut -f2 -d=

  master
```

Relocating a Master Directory:

As mentioned earlier, each occurrence of the same directory is called a replica. One of them must be defined as the master. Before you can relocate an existing master directory, for instance /.:/hosts, you must have a replica directory. Check which replicas are defined and where they are. If none are defined, you have to create them first:

1. Optionally install a secondary CDS server, which automatically creates a clearinghouse. If you did so, skip the next step.
2. Create another clearinghouse locally on the system that should house it, if you did not want to install another CDS server before:

   ```
   # cdscp create clearinghouse /.:/xyz_ch
   ```

   If this command fails, refresh the local CDS clerk cache and try again:

   ```
   # <tool_dir>/cleanup_cds_cache
   ```

3. Create a replica of the directory:

   ```
   # cdscp create replica /.:/hosts clearinghouse /.:/xyz_ch
   ```

   Then 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* and a read-only was on *ev2*:

   ```
   # cdscp set directory /.:/hosts to new epoch master /.:/xyz_ch \
   readonly /.:/ev1_ch /.:/ev2_ch
   ```

   To move the master back to /.:/ev1_ch and define the other two as read-only:

   ```
   # cdscp set directory /.:/hosts to new epoch master /.:/ev1_ch \
   readonly /.:/xyz_ch /.:/ev2_ch
   ```

Relocating a Clearinghouse:

The steps described hereafter backup the whole clearinghouse and restore it to another *existing* CDS server. If the new system does not have a CDS server yet, you must first install one. For example, you may want to move a clearinghouse when:

- You need to temporarily disconnect the host server system from the network for repair or for other reasons.
- You no longer want the current host system to function as a CDS server.

- You want to move the clearinghouse to a server system that is physically closer on the network to the user groups and applications that use the information contained in the clearinghouse.

Assuming you want to relocate the clearinghouse on *ev1* to *ev2*, follow these steps:

1. Install a CDS server on *ev2*, if there is none yet:

   ```
   mkdce -n <cell_name> -s <sec_srv_name> cds_second
   ```

2. Disconnect the clearinghouse from the server where it is currently running. To do so you can use the `cdscp clear clearinghouse /.:/ev1_ch` command on *ev1* which updates the clearinghouse files and ensures the files are consistent.

3. Copy the clearinghouse database files from their current location (source server system) to their new location (target server system).

   To backup the ev1_ch clearinghouse on *ev1*:

   ```
   # tar -cvf/dev/rmt0 /var/dce/directory/cds/*ev1_ch.* \
   /etc/dce/cds_attributes
   ```

4. Perform the next steps on the target system (*ev2*):

   a. Restore the files.

   b. Create a new clearinghouse:

      ```
      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. Change the Tower information in /.:/ev1_ch:

      List the interface definition and note the interface UUID and object UUID:

      ```
      # rpccp show entry /.:/ev1_ch

      objects:

        004a4f0c-951b-1e9a-b254-10005a4f15da

      binding information:

        <interface id>   257df1c9-c6d3-11ca-8554-08002b1c8f1f,1.0
        <string binding> ncacn_unix_stream:[/var/dce/rpc/socket/006bd654-ec
      84-1e9b-ba25-10005a4f4629]
        <string binding> ncadg_ip_udp:9.3.1.68[]
        <string binding> ncacn_ip_tcp:9.3.1.68[]
      ```

      To change the IP address to the address 9.3.1.120 of *ev2*, unexport and re-export the interface definition:

      ```
      rpccp unexport -i 257df1c9-c6d3-11ca-8554-08002b1c8f1f,1.0 \
      -o   004a4f0c-951b-1e9a-b2 54-10005a4f15da /.:/ev1_ch

      rpccp export -b ncadg_ip_udp:9.3.1.120[] \
      -i 257df1c9-c6d3-11ca-8554-08002b1c8f1f,1.0 \
      -o 004a4f0c-951b-1e9a-b254-10005a4f15da /.:/ev1_ch
      ```

```
rpccp export -b ncacn_ip_tcp:9.3.1.120[] \
-i 257df1c9-c6d3-11ca-8554-08002b1c8f1f,1.0 \
-o 004a4f0c-951b-1e9a-b254-10005a4f15da /.:/ev1_ch
```

Do not export the local socket interface. If you need to export another LAN interface, add two more export commands for the additional IP address.

d. Refresh the CDS clerk cache:

```
# <tools_dir>/cleanup_cds_cache
```

e. Update the Tower information for all directories in /.:/ev1_ch:

```
# cleanif -i 9.3.1.68 -n 9.3.1.120 -f/tmp/out.cds
# /tmp/renew_dir_entries
```

After the clearinghouse is created on the new location, it still has the old binding information for all directories. The first command above searches the whole clearinghouse for the address 9.3.1.68 and creates the `renew_dir_entries` command. Executing this generated shell script recreates the replica sets of all directories, which forces an update of the IP addresses to 9.3.1.120. See 4.3.3, "Changing IP Addresses" on page 119 for information on the above two commands.

f. 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.

```
# cdscp delete clearinghouse /.:/ev2_ch
```

5. 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 switch between the two systems.

To reflect this fact in your cell configuration, we must edit the /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 on *ev1*:

```
rmdce cds_second
```

6. Refresh the CDS clerk caches on all systems now:

```
# <tools_dir>/cleanup_cds_cache
```

Relocating a CDS Server:

This task can be performed by relocating all clearinghouses of a CDS server to another CDS server and then deleting the old CDS server, as described above. What you get on the new CDS server are all the clearinghouses of the old server with the original names. You cannot change the names. You get a clearinghouse name which does not correspond to the system name, which in fact does not matter. Another problem with that method is, that it does not merge the relocated clearinghouse with one that might already exist on the target server. If it does not bother you that the clearinghouse name is different from the host name or you might have two clearinghouses, the method mentioned above may be simpler, depending on how the namespace is structured.

If you want to merge with an existing clearinghouse or if you want the new clearinghouse to correspond to the system name, you have to perform many manual steps. You basically have to replicate directories to the target clearinghouse and delete them in the old one. You have to take into considerations that further replicas might exist in other CDS servers. Assume moving the CDS server on *ev1* to node *ev2*, where a (possibly empty) /.:/ev2_ch clearinghouse exists. You want to get rid of /.:/ev1_ch. This means, it needs to be merged with /.:/ev2_ch. These are the generalized steps:

- Create a list of directories in /.:/ev1_ch and /.:/ev2_ch and compare the directories
- Consider first the ones which are in both clearinghouses
  - Read-only replicas in the /.:/ev1_ch need no further considerations
  - All master replicas in /.:/ev1_ch need to be moved to /.:/ev2_ch

    Find further possible replicas of these directories and issue the following command for each directory as shown for /.:/hosts:

    ```
    # cdscp set directory /.:/hosts to new epoch master /.:/ev2_ch \
    readonly /.:/ev1_ch  [/.:/further_ch]
    ```
- Directories which are in /.:/ev2_ch only need no further considerations
- Directories which are in /.:/ev1_ch only need following steps:
  - For read-only directories create a new replica on /.:/ev2_ch.

    Then find all other replicas and the master of each directory and issue the `cdscp new epoch master` command to make the new read-only known to the others.
  - For master directories create a new replica on /.:/ev2_ch

    Find all other replicas of each directory and issue the the `cdscp new epoch master` command as above with /.:/ev2_ch as the master.

  Remember that you have to specify *all* existing replicas with the command `cdscp set dir to epoch new master`.
- Remove the CDS server

  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 /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” on page 133. Then this secondary CDS server can be removed with:

  ```
  rmdce cds_second
  ```

4.3.4.4 Relocating the Primary Security Server

The following part is an extract of the latest DCE security server release notes that are shipped with PTF#U431018 for AIX DCE 1.2 or with AIX DCE 1.3. We have not tested it within our test scenarios. The procedure consists of many manual steps. Most of them cannot be performed remotely. We strongly recommend relocating the primary security server only in extreme situations. For example when you must upgrade your system for performance reasons or your system is defective and can no longer be repaired.

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 machine may be an existing client or a security secondary machine. Relocation steps are described for the following scenarios:

1. Primary to client (with one or more secondaries running in the cell)
2. Primary to secondary (with no other secondaries running in the cell)

In the instructions below, you may see passwords, such as -dce-, or replica names, such as repl1, referenced. Replace these values with the values that your environment is configured with.

Please note:

1. It is advisable to unset the BIND_PE_SITE environmental variable. Ensure that the rpccp cache on the /.:/sec groups has the current information. To execute this update issue the `rpccp show group /.:/sec -u` command on each machine. To avoid problems with old ticket information, first exit from all the old dce_login shells by invoking the `ps` command. Then exit from all shells until only one is left.
2. It is recommended that you maintain a backup of the security database under /opt/dcelocal/var/security so that you can reconstruct the primary server in case of a failure, see 4.4.1.2, “Backing Up the Files of the Security Server” on page 143.

Primary to Client:

Following is the minimum initial cell configuration for this type of relocation procedure. A single, three-machine cell that has:

- A node configured as the primary security server (the original primary)
- A node configured as the secondary security server (the secondary)
- A node configured as a DCE client (the new primary).

If the primary server is up and running, ensure that the latest information is written to the files by performing all steps logged in as local user root on the specified machine.

1. On the original primary machine:

   Perform the following steps:

   ```
   dce_login cell_admin
   > sec_admin -s /.:/subsys/dce/sec/master
   sec_admin> state -m
   sec_admin> state -s
   ```

   Back up the registry database and support files using the AIX `tar` command.

```
sec_admin> quit
> cd /opt/dcelocal/var/security
> tar cvf <tarfile> ./.mkey ./rgy_data/*
```

Notes:
The <tarfile> specified can be a tape drive or other backup device, or it can be a regular file that you can send in binary mode by means of the rcp or ftp command to the new primary Security Server machine. If the original primary node is up and running during primary relocation, you can transfer the files individually.

2. On the original primary machine:

   If secd is still running, stop it by using the AIX kill -2 command. An alternate method is to perform dce_login as cell_admin and use the stop subcommand of sec_admin to stop the primary (make sure sec_admin is bound to the primary site).

   Notes:
   If you stop the secd daemon with the AIX kill command using any flag except SIGINT (-2), you can cause secd to exit without performing cleanup routines. If this is the case (as may well be if secd dies unexpectedly), these cleanup steps will have to be done manually (see step 3). Otherwise, go to step 4.

3. On the original primary machine:

   Clean up the local endpoint map if the secd bindings are still registered. This may happen if secd is killed with a signal that does not have a signal handler or if it ends unexpectedly. Enter the following:

   ```
   > rpccp remove mapping \
   -b <string_binding> \
   -i <interface_id> \
   -o <object_uuid>
   ```

   Notes:
   Perform this for each secd entry in the endpoint map. These are the entries that have the following notation when you issue the rpccp show mapping command:

   ```
   <annotation> "DCE user registry"
   ```

4. On the original primary machine:

   Remove the registry database and support files with the following:

   ```
   > rm /opt/dcelocal/var/security/.mkey
   > rm -r /opt/dcelocal/var/security/rgy_data*
   > rm /opt/dcelocal/var/security/tmp/krb5kdc_rcache
   > exit
   ```

   Comment out the line in /etc/rc.dce that will restart secd upon rc.dce by adding a pound sign (#) to the beginning of the line.
   Before:

   ```
   daemonrunning $DCELOCAL/bin/secd
   ```

   After:

   ```
   #daemonrunning $DCELOCAL/bin/secd
   ```

   Delete the line in /etc/mkdce.data that indicates the primary security server is running on the local machine.

   ```
   sec_srv     COMPLETE      Security Server
   ```

5. On the new primary node (the original client):

   Restore the backed up files by issuing the following:

   ```
   > cd /opt/dcelocal/var/security
   > tar xvf <tarfile_created_in_step1>
   ```

   Verify that the permission set on the restored rgy_data directory is 755 and the permission set on all restored files (.mkey and all files under rgy_data) is 600.

6. On every machine in the cell:

   Update the primary server′s entry in /opt/dcelocal/etc/security/pe_site to reflect the client as the new primary (update the IP address). To do this, replace the bindings containing the original primary replica address with the address of the new primary (the original client). Ensure that these bindings are always the first in the file. The IP address of the machine can be found by issuing the AIX host command:

   ```
   > host <machine_name>
   ```

   Update the entry in /krb5/krb.conf to reflect the client as the new primary machine.

7. On the new primary machine:

   Ensure that you are running under only one KSH shell.
   Start secd by issuing:

   ```
   > secd -d -v
   ```

   This will start secd in the foreground. Wait approximately three minutes until secd starts exporting its bindings to the namespace. Uncomment the line in /etc/rc.dce so that secd will be restarted on re.dce.
   Before:

   ```
   #daemonrunning $DCELOCAL/bin/secd
   ```

   After:

   ```
   daemonrunning $DCELOCAL/bin/secd
   ```

   Add the following line to /etc/mkdce.data to indicate that the primary security server is running on the local machine:

   ```
   sec_srv     COMPLETE     Security Server
   ```

   secd may not start at the first invocation; you may see the following error messages:

   ```
   Registry: Fatal Error - Cannot establish dce registry identity
   at line 837 of
   file ../../../../../src/security/server/rs/rs.c - 0x1712207b -
   Registry server unavailable (dce / sec)
   Registry: Fatal Error - DT exiting with an exception
   at line 793 of file ../../../../../src/security/server/rs
     /rs_main.c -
   0x1712207
   b - Registry server unavailable (dce / sec)
   Registry: Fatal error in main thread; exiting
   ```

   If secd does not start, restart it; it should come up on a subsequent try. It is recommended that you start secd in debug/verbose mode (using the -d and -v flags); when you see the message that secd is exporting its bindings to /.../<cellname>/subsys/dce/sec/master, you can assume it has started properly. Starting secd may take several minutes. If it fails to export its bindings after several minutes, use cdsdel to delete the /.:/subsys/dce/sec/master object from the namespace and use rpccp to remove member /.:/sec -m /.:/subsys/dce/sec/master. Then try to start secd again.

8. After secd has started successfully, refresh the CDS cache on every machine in the cell as follows:

   ```
   > kill -9 <PIDs of all CDS daemons currently running on
     the machine>
   > cd /opt/dcelocal/var/adm/directory/cds
   > rm cds_cache.*
   > rm cdsclerk_*
   > rc.dce cds
   ```

   If you do not refresh the CDS cache, you may experience binding failures to the new primary machine. It is important to use the AIX kill command with the sigkill (-9) flag so that the CDS daemons do not write in-memory information to disk; do not use dce.clean or kill -2 in this step.

9. On every machine in the cell:

   Stop and restart sec_clientd so that it will rebind to the new primary machine. Use the -purge option so that existing credentials will be destroyed and recreated.

   ```
   > dce.clean sec_clientd
   > sec_clientd -purge
   ```

   If you did not refresh the CDS cache (in step 8), sec_clientd may fail to restart and the following messages may be displayed:

   ```
   sec_clientd 09/25/93 21:31:27 - Unable to validate machine
   context ... Registry server unavailable (dce / sec)

   sec_clientd 09/25/93 21:31:27 -
   Unable to establish valid machine context creds
   during initialization...ABORTING ... Registry server
   unavailable (dce / sec)
   ```

   Other long-running applications that may have security server binding handles cached in their runtime should also be stopped and restarted. After all these steps are successful, you may want to issue dce.clean secd and restart normally with rc.dce secd. Run chpesite on all machines in the cell to update the /opt/dcelocal/etc/security/pe_site file.

Primary to a Secondary:

Following is the minimum initial cell configuration for this type of relocation procedure.

A single, two-machine cell that has the following:

- A node configured as primary security server (the original primary)
- A node configured as a secondary security server to be the new primary machine (called repl1)

If the primary replica is running, ensure that the latest information is written to the files by performing all steps while logged in as local user root on the specified machine.

1. Exactly the same step as in *Primary to Client*

2. Exactly the same step as in *Primary to Client*
3. Exactly the same step as in *Primary to Client*
4. Exactly the same step as in *Primary to Client*
5. On the new primary node (currently the secondary machine):

   Remove references to the security primary in the namespace:

   ```
   > dce_login cell_admin
   > cdscp delete obj /.:/subsys/dce/sec/master
   ```

   Also remove /.:/subsys/dce/sec/master from the rpc groups:

   ```
   > rpccp remove member /.:/sec -m /.:/subsys/dce/sec/master
   > rpccp remove member /.:/sec-v1 -m /.:/subsys/dce/sec/master
   ```

   The `cdscp delete object` and `rpccp` operations can take several minutes. Now destroy the secondary machine:

   ```
   > sec_admin -s /.:/subsys/dce/sec/repl1
   sec_admin> destroy subsys/dce/sec/repl1
   sec_admin> quit
   > exit
   ```

   Remove the .mkey file that remains:

   ```
   > rm /opt/dcelocal/var/security/.mkey
   ```

   Verify that the /opt/dcelocal/var/security/rgy_data directory is empty. Restore the backed up files:

   ```
   > cd /opt/dcelocal/var/security
   > tar xvf <tarfile_created_in_step1>
   ```

   Verify that the permission set on the restored rgy_data directory is 755 and the permission set on all restored files (.mkey and all files under rgy_data) is 600.

6. Exactly the same step as in *Primary to Client*
7. On the new primary machine:

   Start secd by issuing:

   ```
   > secd -d -v
   ```

   This will start secd in the foreground. Wait approximately three minutes until secd starts exporting its bindings to the namespace.

   secd may not start at the first invocation; you may see the following error messages:

   ```
   Registry: Fatal Error - Cannot establish dce registry identity
   at line 837 of file ../../../../../src/security/server/rs/rs.c -
   0x1712207b - Registry server unavailable (dce / sec)

   Registry: Fatal Error - DT exiting with an exception
   at line 793 of file ../../../../../src/security/server/rs
     /rs_main.c -
   0x1712207 b - Registry server unavailable (dce / sec)

   Registry: Fatal error in main thread; exiting
   ```

   If secd does not start, restart it; it should come up on a subsequent try. It is recommend that you start secd in debug/verbose mode (using the -d and -v flags); when you see the messages that secd is exporting its bindings to /.../<cellname>/susbys/dce/sec/master, you can assume it has started properly. Starting secd can take several minutes. If it fails to export its bindings after several minutes, use cdsdel to delete the /.:/subsys/dce/sec/master object from the namespace and try to restart secd again.

Wait for three minutes after secd has exported its bindings to the namespace and propagated its database to any secondaries, and then perform the following steps to complete cleanup of the old secondary machine:

```
> dce_login cell_admin
> sec_admin -s /.:/subsys/dce/sec/master
sec_admin> delrep subsys/dce/sec/repl1 -f
sec_admin> quit
> exit
```

8. On every machine in the cell (except for the new primary machine):

   Refresh the cds cache as follows:

   ```
   > kill -9 <PIDs of all CDS daemons currently running on
     the machine>
   > cd /opt/dcelocal/var/adm/directory/cds
   > rm cds_cache.*
   > rm cdsclerk_*
   > rc.dce cds
   ```

   Also execute the following commands to update the local caches used by rpccp:

   ```
   > rpccp show group /.:/sec -u
   > rpccp show group /.:/sec-v1 -u
   ```

   If you do not refresh the CDS cache, you may experience binding failures to the new primary machine. It is important to use the AIX kill command with the sigkill (-9) flag so that the CDS daemons do not write in-memory information to disk; do not use dce.clean or kill -2 in this step.

9. Exactly the same step as in *Primary to Client*

4.4 Backup/Restore and Other Housekeeping Tasks

As with any operating system or application programs 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 online backup. However, there may be cases where replication is not sufficient and traditional backups are still needed or at least recommended such as:

- Accidental removal of a series of DCE accounts
- Accidental removal of the CDS namespace contents
- Release upgrades
- Attempt to change the cellname
- Unsuccessful change of an IP address
- Accidental removal of database files

The following sections cover:

1. Backup/restore of DCE core services databases
2. Backup/restore of DFS databases

3. Backup/restore of DFS data
4. Controlling system created files that may fill disk space
5. Managing caches

4.4.1 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 show the lack of need to backup RPC data, and show the way in which to save the security service and the CDS service files.

4.4.1.1 No RPC Defined Mapping to Backup

The RPC component of DCE has no static data to be taken into account.

When the DCE processes of a cell restart, all the RPC mappings will be repopulated by the starting DCE servers of a system. Following are the important RPC files:

`/var/dce/rpc/rpcdep.data`	This is the endpoint map which is stored on disk so that the RPC daemon can be stopped and restarted without requiring servers to reregister with the RPC 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 RPC daemon starts. This is done by the script /etc/rc.dce which starts up all the DCE processes.
`/var/dce/rpc/rpcdllb.dat`	Along with the endpoint map database rpcdep.dat, `rpcd` creates the NCS 1.5.1 local location broker file. Like rpcdep.dat, rpcdllb.dat needs to be deleted at system boot time.
`/var/dce/rpc/socket`	Is a directory which contains the local sockets for the system. A server which is going to unregister, deletes its entries in this directory. There is no special maintenance needed.

4.4.1.2 Backing Up the Files of the Security Server

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. Login 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. Backup 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
```

or if stopped restart it (`rc.dce sec_srv`)

4.4.1.3 Restoring the Files of the Security Server

1. Login 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
   ```

4.4.1.4 Backing Up the Files of a CDS Server

The CDS namespace is divided into one or several clearinghouses. This section looks at how to backup the files associated with one specific clearinghouse. We have tested three 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 read-only copy for that clearinghouse. The description of this procedure assumes that there is only one CDS server in the cell with all master directories. These steps are automated in the shell script backup_CH.

   Of course this is not realistic for a production environment. However, the description might be helpful to give ideas on how such a requirement could actually be met in more complex environments.
3. The third case provides a solution where uninterrupted write access to the clearinghouse is required. This procedure is very similar to the second one above, but it creates a write-able copy of the clearinghouse. The same restrictions apply.

Backup by Disabling the Service:

This is the normal procedure as it is also 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 backup all clearinghouses which contain one or more master replicas of any directory. Backup 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 recreate consistency upon restore.

Before you begin to backup clearinghouses you must know where your master replicas are. See also 4.3.4.3, “Moving CDS Resources” on page 132 to find some helpful commands to find all master and read-only replicas.

1. Login to the CDS server on which you want to backup a clearinghouse

2. Login as cell_admin
3. Disable the clearinghouse you want to backup, for instance /.:/ev1_ch, with one of the three following methods:
   - With cdsdiag:

     ```
     # cdsd_diag >
     manage dc /.:/ev1_ch
     ```

   - Disabling the CDS server, which would affect all clearinghouses

     ```
     # dce.clean cds
     ```

     The same is achieved with `cdscp disable clerk` followed by `cdscp disable server`.
   - Disconnecting a specific clearinghouse from the CDS server:

     ```
     # 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. Backup all files associated with clearinghouse ev1_ch:

   ```
   # tar -cvf/dev/rmt0 /var/dce/directory/cds/*ev1_ch.* \
   /var/dce/directory/cds/cds_files /etc/dce/cds_attributes \
   /etc/dce/cds_config  /etc/dce/cds.conf
   ```

5. Reactivate the clearinghouse according to the method you had stopped activities with before:
   - With cds_diag:

     ```
     # cdsd_diag >
     manage ec /.:/ev1_ch
     ```

   - Restarting the CDS server:

     ```
     # /etc/rc.dce cds
     ```

     Or with OSF commands:

     ```
     # cdsadv
     # cdsd
     ```

   - Reconnecting the clearinghouse with the CDS server

     ```
     # cdscp create clearinghouse /.:/ev1_ch
     ```

Keeping a Read-only Copy Active:

What we describe here is a procedure on how to create a read-only copy of a clearinghouse to be able to backup the primary clearinghouse while information is still available from the new read-only clearinghouse. We have provided the shell script backup_CH for this case. It works in a very simple environment with just one CDS server housing all master replicas. However, we consider the script useful for an understanding on how this could be done even in a distributed and replicated environment.

This procedure does the following steps:

1. Creates a temporary clearinghouse /.:/xxx_ch
2. Creates a replica for each directory in the clearinghouse and defines the replica set with:

```
# cdscp create replica /.:/hosts clearinghouse /.:/xxx_ch
# cdscp set directory /.:/hosts to new epoch master /.:/ev1_ch \
readonly /.:/xxx_ch
```

3. Then backup /.:/ev1_ch as described above in "Backup by Disabling the Service" on page 144
4. Once /.:/ev1_ch is backed up and running, the temporary clearinghouse can be deleted:

   ```
   # cdscp delete clearinghouse /.:/xxx_ch
   ```

The command `cdscp set dir to new epoch master` defines the replica set, which assigns the role of each replica of a certain directory to either master, read-only or excluded. Since you have to specify all existing replicas with this command, you have to know or find them. This is not implemented in our script.

Furthermore, in a more complex environment you probably have replication already implemented to a certain extent. If all directories of a clearinghouse are replicated, you need not care about read-only availability, it is there. If this is not the case, you need to find all master directories which do not have a read-only replica and copy only those.

The script backup_CH is self-explaining and can be found in the diskette that comes with this publication.

Keeping the Clearinghouse Active for Write-able Access:

If write access is really needed at all times, then this procedure should be looked at. Again, this procedure describes a case where there is only one CDS server. If there are already multiple CDS servers, you need to adjust the steps that execute the command `cdscp set dir to new epoch master` in such a way that you find all replicas for a replica set and specify all of them in this command.

What you basically would have to do for each clearinghouse that contains master replicas is:

1. Create a temporary clearinghouse
2. Find all master directories, for each:
   a. Create a replica in the new clearinghouse
   b. Find all read-only replicas that might exist
   c. Create the new replica set with the master replica on the temporary clearinghouse; exclude the one on the original clearinghouse to prevent updates
3. Backup the original clearinghouse as outlined in "Backup by Disabling the Service" on page 144

   You want to backup this clearinghouse and not the temporary one, because you want to keep its name.
4. After the backup redefine the replica set for all directories in the temporary clearinghouse so that the master is back on the original
5. Delete the temporary clearinghouse

The following is an outline of the procedure we tested in a scenario with just one CDS server:

1. Select the directories of the clearinghouse that must be kept active with read write access while backing up the files.
2. Replicate each of those directories which have their master copy in this clearinghouse.

   A skulk, that is, an update of the read-only copy of a CDS directory, is performed when the cdscp set to new epoch command is executed on that directory. By excluding the directory just after this skulk we can prevent another skulk to occur later, if someone updates the master of a directory before the clearinghouse is disabled.

   Three steps need to be applied on each directory :

   ```
   # cdscp create replica /.:/dir_name/sudir1 clearinghouse /.:/xxx_ch
   ```

   This first creates the directory copy (called read-only replica) into the second clearinghouse (/.:/xxx_ch).

   ```
   # cdscp set directory /.:/dir_name/subdir1 to new epoch master  \
   /.:/xxx_ch readonly /.:/ev1_ch
   ```

   Secondly the directory copy in the second clearinghouse is populated and switched to be the master or read-write copy (called master replica). The original directory is put in read-only state (read-only replica).

   ```
   # cdscp set directory /.:/dir_name/subdir1 to new epoch master  \
   /.:/xxx_ch exclude /.:/ev1_ch
   ```

   Instead of keeping the read-only copy online for the backup, this third step excludes the directory replica from the namespace. The contents of this directory copy cannot be affected by any changes done to the master. When it will be brought back to activity after the backup, it will be populated with the possible changes that might have occurred in the meantime to the temporary master in /.:/xxx_ch.
3. Disable the clearinghouse(s) associated with the server to backup:

   ```
   # cdsd_diag >
   manage dc /.:/ev1_ch
   ```
4. Backup the associated files as in the preceding case
5. Reactivate the clearinghouse(s):

   ```
   # cdsd_diag >
   manage ec /.:/ev1_ch
   ```
6. Bring back to activity the directories that may have been excluded from the namespace. The dual operation of the example shown there becomes:

   ```
   # cdscp set directory /.:/dir_name/subdir1 to new epoch master  \
   /.:/xxx_ch readonly /.:/ev1_ch
   ```

   This step updates the read-only replica that was explicitly excluded and updates it with the changes that may have occurred in the meantime to the master replica within clearinghouse /.:/xxx_ch.
7. Move back the master copy of each directory replica to the initial clearinghouse (/.:/ev1_ch).

   ```
   # cdscp set directory /.:/subsys to new epoch master /.:/ev1_ch  \
   readonly /.:/xxx_ch
   ```

   For an application that must be kept in permanent activity, the administrator may choose to keep the associated entries in a specific set of clearinghouses distributed across the cell and back them up more frequently.

8. Switch back the states of each directory replica with their master copy back in the initial clearinghouse (/.:/ev1_ch).

4.4.1.5 Restoring the Files of a CDS Server

To restore any of the clearinghouses perform the following steps:

1. Stop DCE
2. Restore the database(s)
3. Restart DCE
4. Clean all clerk caches with `cleanup_cds_cache` (see 4.4.5, "Managing Caches on Client Machines" on page 156)

4.4.2 Backing Up DFS Servers Related Information

There are two types of entities that need backup considerations. The first consists of the two databases that store fileset locations (FLDB) and configuration of the DFS backup subsystem, the backup database. The fileset data builds the second type, which is discussed in 4.4.3, "Backing Up and Restoring DFS Data" on page 152.

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 ITSO publication *The Distributed File System (DFS) for AIX/6000* or InfoExplorer*.

In the following subsections we want to provide a short recapitulation on how the FLDB is backed up and restored.

4.4.2.1 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 to lose 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 worst cases it might make sense to backup the FLDB in regular intervals, especially if a lot of fileset replication is defined or you are using 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:

   ```
   # bos status /.:/hosts/ev8 -localauth
   Instance upserver, currently running normally.
   Instance flserver, currently running normally.
   Instance ftserver, currently running normally.
   # bos stop /.:/hosts/ev8 flserver -localauth
   # 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 /var/dce/dfs/fldb*
   -rw-------   1 root  145472 Jun 10 14:24 /var/dce/dfs/fldb.DB0
   -rw-------   1 root      64 Jun 10 14:24 /var/dce/dfs/fldb.DBSYS1
   # cp  -v /var/dce/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
   # 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.

4.4.2.2 Restoring an FLDB

Please Note

The steps outlined here are a summary of ideas based on our experiences in this project and were not 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 would 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 4.4.2.1, "Backing up the FLDB" on page 148; 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

4.4.2.3 Recreating an FLDB

If the files of the FLDB servers are corrupted or accidentally destroyed, the flserver can be started anyway, but 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 :

   `fts crserverentry` 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:

```
# 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 checks that the FLDB correctly records every fileset whose fileset header is marked on-line. It 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 FL Server 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.

Because non-LFS filesets do not have fileset headers, the fts syncfldb and fts syncserv have limited effectiveness on non-LFS filesets.

4.4.3 Backing Up and Restoring DFS Data

There are basically two methods to backup DFS data:

- Using the DFS backup subsystem with the bak command suite.

  As mentioned above in 4.4.2.1, "Backing up the FLDB" on page 148 the backup service allows for setting up sophisticated automated backup procedures. You define tape coordinator machine(s) that have tapes 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 ITSO publication *The Distributed File System (DFS) for AIX/6000* or InfoExplorer.

4.4.4 Controlling Disk Space: System Created Files

DCE is creating a lot of files which it uses for its operation:

- 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 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 /var directory. The /var file system is also used for other variable size system files. It is important to create separate file systems for DCE to decouple DCE from the operating system as advised in 4.1.1, “Preparing for DCE Configuration” on page 84.

4.4.4.1 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 of about 1KB per user.

- CDS server: /opt/dcelocal/var/directory/cds

  The database files mainly grow with the number of client workstations (approximately 1.4KB per client), but also with the number of application servers. If you are using Single Login/6000, the number of database entries also grows with the number of users, which might become critical. Each user has an object of approximately 2.1KB in CDS.

- RPC daemon: /opt/dcelocal/var/rpc

  The endpoint map files grow with the number of application servers. The number of socket files 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 31 for information on disk space requirements.

4.4.4.2 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 larger than 90% of the logical volume size. For a new cache size to take effect, the system must be rebooted.

See also 4.4.5, “Managing Caches on Client Machines” on page 156 for more information about cache management.

4.4.4.3 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 /var/dce/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 /var/dce/security/creds | wc -w
     13
```

Put the command rmxcred into crontab so that credentials are cleaned up at regular intervals:

1. Login 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

4.4.4.4 Socket Files for Local RPC

For a description of local RPC see also 5.1.3, "Local RPCs" on page 221. If DCE clients and servers are on the same machine, they use the ncacn_unix_stream protocol sequence which communicates over a local socket special file.

The endpoint for a ncacn_unix_stream binding handle is represented as a full pathname to a UNIX socket file. When allowing the RPC runtime to assign an endpoint to an ncacn_unix_stream binding, the binding will be an object UUID, which is guaranteed to be unique. This means there is no chance that a socket file will ever be used over again on two invocations of a DCE application.

For a newly configured cell with the following services:

```
<ev1:root>cat /etc/mkdce.data
cds_cl       COMPLETE   CDS Clerk
cds_srv      COMPLETE   Initial CDS Server
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
dts_local    COMPLETE   Local DTS Server
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
sec_srv      COMPLETE   Security Server
```

The number of sockets observed is 152.

```
<ev1:root>ls /var/dce/rpc/socket | wc -w
    152
```

For each DCE command a new file is created:

```
# dce_login -c cell_admin dce
Password must be changed!
# ls /var/dce/rpc/socket | wc -w
     153
```

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, if it had been system created.

When for any reason these files are not cleaned up, as was the case with the DCE build we were using, the number of files can reach the limit of the file system. All DCE activity will be frozen due to lack of i-nodes. We observed an increase up to 2463 inodes after less than two weeks of cell activity.

```
# ls /var/dce/rpc/socket | wc -w
   2463
```

If this happens, the following utility can be run to delete stale socket files:

```
rpc.clean -p /var/dce/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 /var/dce/rpc/socket could also be manually deleted.

Please note

This happened to us because we were using very early code. The socket files are deleted by the endpoint map, if an application correctly unregisters upon termination. Only if applications do not unregister correctly, you might experience a growing number of stale socket files.

4.4.5 Managing Caches on Client Machines

Client systems house to 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 timeouts occur, or to completely hang, if one of the following occurs:

- They point to a server which is out of order
- Information is outdated
- Cache is corrupted or destroyed by a mistaken operator intervention

This section covers the aspects of refreshing these caches. See also 4.4.4.2, "Cache Files" on page 153 for limiting the cache sizes.

4.4.5.1 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 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 timeouts 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 timeout 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 157 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, which 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 timeouts on client machines for a couple of 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 such as:

```
rpccp show entry /.:/<some_entry_name> -u
```

The rest of this section covers the following topics:

1. An InfoExplorer 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. It is the role of the client program to ensure that the information which it receives from the CDS is current enough for its uses. 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 InfoExplorer:

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:

Since binding handles are cached in each CDS clerk cache, we have to refresh it on all systems, if binding handles become invalid. Otherwise we may experience nasty timeouts. 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)

CDS_SERVER=NO

ps -ef | grep cdsd >/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″
```

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 which 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, CDS cache is kept while rebooting or restarting DCE. RPC end point 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.
cat << EOI
If you want to continue, you have to restart all your DCE Applications
and user 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

#kill dce
/etc/dfs.clean
/etc/dce.clean

#Remove the socket files
rm /opt/dcelocal/var/rpc/socket/*  >/dev/null 2>&1

#Remove the rpc config files
rm /opt/dcelocal/var/rpc/rpcdep.dat >/dev/null 2>&1
rm /opt/dcelocal/var/rpc/rpcdllb.dat >/dev/null 2>&1

#Remove everyone′s credentials files
rm /opt/dcelocal/var/security/creds/* >/dev/null 2>&1

#Remove everyone′s cdscache files
rm /var/dce/adm/directory/cds/cds_cache.* >/dev/null 2>&1
rm /var/dce/adm/directory/cds/cdsclerk_* >/dev/null 2>&1
```

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 which is not in the same network as the CDS server you call mkdce with the -c flag. This instructs mkdce to setup 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 find the security
server via the pe_site file.
EOI
echo "Is /etc/dce/security/pe_site up to date? [y/n]: \c"
read a

if [ X$a != X"y" ]
then
  echo "\n>> Please edit /etc/dce/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.

4.4.5.2 Managing the DFS Client Cache

This topic is comprehensively described in the ITSO publication *The Distributed File System (DFS) for AIX/6000* and in InfoExplorer. 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. 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, 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.

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.

4.5 Administering Users and Groups

User management encompasses tasks such as adding, modifying, deleting users, accounts, and groups in the DCE security registry. Current DCE implementations lack tools to perform these in a large scale. Managing single users is nicely supported within SMIT. Easy to use menus allow one to add, modify, and delete users, accounts, groups and organizations. The second problem after mass DCE user management is that DCE login is not integrated into AIX login. This means the user has to login to AIX first and then to DCE. The administrator has to define all AIX/DCE users in at least one AIX local system and in the DCE registry.

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 5.5, “User (and ACL) Management” on page 242 we describe a set of tools for user management which we have designed and implemented during this project
- In 5.4, “Single Login/6000” on page 235 we give a short summary description of an integrated login package available from the IBM lab in Boeblingen, Germany

The user management tools are designed to also manage user related ACLs. This is important to support tasks such as migration from other environments to DCE or splitting/joining of existing DCE cells.

This section focuses on the use of the user management tools and explains how to configure Single Login/6000:

1. Adding users
2. Modifying users
3. Moving users
4. Deleting users
5. Aliases
6. A test with adding 32,000 users
7. Configuring Single Login/6000

The concepts and structure of the tools as well as the details about the commands are described in 5.5, “User (and ACL) Management” on page 242.

Make sure the management tools have been installed. See Appendix A, “Installing the Tools” on page 277 for instructions how to install the tools.

Assume we have a directory /umgt which is to hold our user information database and you have restored the shell script into this directory. In order to use the commands correctly you need to change into that directory and make sure your PATH variable contains the current directory:

```
# cd /umgt
# PATH=$PATH::
# echo "PATH=\$PATH::" >> /etc/environment
```

4.5.1 Adding Users

The concepts and the syntax of the commands used to add new users is described in 5.5.4, "Adding Users: add_users" on page 258 and 5.5.5, "Enabling Users for DCE Login: rgy_enable_users" on page 263.

This section gives a few examples of the usage:

1. Create users from list of user names
2. Create a user with specific UID
3. Create a user for Single Login/6000

DCE account attributes

We define attributes in the DCE registry for each user such as the home directory and the initial program. However, as long as the user needs to have a local /etc/passwd entry and is required to login there first, these attributes have no effect. The local attributes are used.

Defining the home directory in DFS is another problem, because at AIX login time the user does not have their credentials yet. See 4.2.4, "Defining Home Directories in DFS" on page 115 for instructions on how to define the home directory in DFS.

These issues are solved with Single Login/6000. See 5.4, "Single Login/6000" on page 235 for more details.

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 "barry brice ben" > /tmp/users
   ```

2. Call `add_users`: to create the principals and a default accounts which 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. Login 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 barry ...ok      (file will be created)
  Checking brice ...ok      (file will be created)
  Checking ben ...ok      (file will be created)

You are going to add       3 users
Starting to work with rgy_edit ...

Please provide your password:


   Adding principal barry ...ok
   Adding principal brice ...ok
   Adding principal ben ...ok

*** Ended to add users in DCE
```

4. Check the principals that were created:

```
#rgy_edit -p -v | grep ^b
bin                                          3
barry                                      349
brice                                      350
ben                                        351
```

5. Check the accounts that were created:

```
#rgy_edit -a -v | grep ^b
bin [bin none]:*:3:3::/bin::
barry [none none]:*:349:12::/::
brice [none none]:*:350:12::/::
ben [none none]:*:351:12::/::
```

6. The UDF of user barry looks as follows:

```
# cat dce_users/barry

# -- !!!!!!!!!!!!!  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 barry
homedir=/:/dfs_home/barry
size=
initprog=/bin/ksh
expir_date=95/09/15
good_since=94/09/15
# --- ACL_INI info:
# --- ACL info:
# --- State and last access:
state=SUSPENDED
last_time_access=Thu Sep 15 13:31:04 CDT 1994  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, his 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 *Barry White, Dept 99S*.

```
# echo ″ADD_gecos=Barry White, Dept 99S″ >> dce_users/barry
# cat dce_users barry
```

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 barry ...ok
  Checking brice ...ok
  Checking ben ...ok

You are going to rgy_enable       3 users
Starting to work with rgy_edit ...

barry brice ben
   RGY-enabling principal barry ...  ok
   RGY-enabling principal brice ...  ok
   RGY-enabling principal ben ...  ok

*** Ended to rgy_enable users in DCE
```

8. List the DCE accounts:

```
#rgy_edit -a -v | grep ^b
bin [bin none]:*:3:3::/bin::
barry [none none]:*:349:12:Barry White, Dept 99S:/:/dfs_home/barry:/bin/ksh:
brice [none none]:*:350:12:Account for brice:/:/dfs_home/brice:/bin/ksh:
ben [none none]:*:351:12:Account for ben:/:/dfs_home/ben:/bin/ksh:
```

All the account information is now filled in. Note the GECOS field for user barry.

9. List the UDF file for barry again, his GECOS field is now changed and the ADD_gecos has disappeared. The state is now RGY_ENABLED and his account valid:

```
# cat dce_users/barry

# -- !!!!!!!!!!!!!  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=Barry White, Dept 99S
homedir=/:/dfs_home/barry
size=
initprog=/bin/ksh
expir_date=95/09/15
good_since=94/09/15
# --- ACL_INI info:
```

```
# --- ACL info:
# --- State and last access:
state=RGY_ENABLED
last_time_access=Thu Sep 15 13:52:15 CDT 1994  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 us assume we want to add a user 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 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:

```
# rgy_edit -p -v felix | sed /Current/d
felix                                          1001
# rgy_edit -a -v felix | sed /Current/d
felix [none none]:*:1001:12::/::
```

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:

```
# rgy_edit -a -v felix | sed /Current/d
felix [none none]:*:1001:12:Account for felix:/:/dfs_home/felix:/bin/ksh:
```

8. Also check the UDF file. The account is valid now and its state is RGY_ENABLED.

Adding Users for Single Login/6000:

Single Login/6000 supports a structured user namespace. Departments or regional offices can be defined. If you decide to implement departments, then each DCE node needs to be defined into a department. When users are added, you need to specify to which department they belong. They can then login to any machine in their department. If they are defined as global users, they can even login on any machine in the cell.

Single Login/6000 users have a CDS entry to control their current location and number of logins. The Single Login/6000 CDS namespace is also structured according to the departments. Each department has a CDS directory and its users have a CDS object in that directory. The DCE principal name reflects this structure, it is a qualified name consisting of the department name and the user name.

See also 4.5.6, “Configuring Single Login/6000” on page 180 on how to configure Single Login/6000.

So for instance user sal in the regional office Munich would have the principal name munich/sal. The UDF supports this structure. Although you could define another user sal in Frankfurt, we do not recommend this, if sal is a global user who is allowed to login at any location. Since Single Login/6000 creates an /etc/passwd file entry on the system at which the user logs in, frankfurt/sal and munich/sal would both become sal on that system.

Let us add user sal in regional office munich:

1. Add the user:

```
# add_users munich/sal
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to add ...
  Checking munich/sal ...ok      (file will be created)

You are going to add       1 users
Starting to work with rgy_edit ...

Please provide your password:


   Adding principal munich/sal ...ok

*** Ended to add users in DCE
```

2. List his UDF. Note that his filename is munich%sal. This is to avoid subdirectories in the repository:

```
# cat dce_users/munich%sal
# -- !!!!!!!!!!!!!!  Do NOT change manually the first part  !!!!!!
# --- Principal info:
uuid=00000161-e86c-2e79-a100-02608c2fff91
uid=353
groups=none
# --- Account info:
group=none
org=none
valid=NO
gecos=Account for munich%sal
homedir=/:/dfs_home/munich/sal
size=
initprog=/bin/ksh
expir_date=95/09/16
good_since=94/09/16
# --- ACL_INI info:
# --- ACL info:
# --- State and last access:
state=SUSPENDED
last_time_access=Fri Sep 16 13:48:18 CDT 1994  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):
```

3. Use the rgy_enable_user command on the user. The rest works the same as for the other users without department names.

   The DFS home directory also reflects the department structure. You have to create the directories accordingly.

See 4.5.6.5, "Configuring Single Login/6000 Users" on page 186 on how you configure this user as a Single Login/6000 user.

4.5.2 Modifying Users

Modifying users means changing some of their definitions in an already existing account. For this purpose we have to bring the user into the SUSPENDED state, to modify their attributes. 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 barry

   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 to suspend users first.

Changing the Primary Group of a User:

In order to assign user barry a new primary group, g7, we must perform the following steps

1. Check the DCE account information; the primary group is none:

```
# rgy_edit -a -v barry | sed /Current/d
barry [none none]:*:349:110:Barry White, Dept 99S:/:/dfs_home/barry:/bin/ksh:
```

2. Suspend user barry:

```
# susp_users barry
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to suspend ...
  Checking barry ...ok

You are going to suspend       1 users
Starting to work with rgy_edit ...

   Suspending account barry ...ok

*** Ended to suspend users in DCE
```

3. Add the necessary instructions into his UDF:

```
# echo ″ADD_newgrp=g7″ >> dce_users/barry
```

   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 barry again:

```
# rgy_enable_users barry
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to rgy_enable ...
  Checking barry ...ok

You are going to rgy_enable       1 users
Starting to work with rgy_edit ...

   RGY-enabling principal barry ...ok

*** Ended to rgy_enable users in DCE
```

5. Check the DCE account information:

```
# rgy_edit -a -v barry | sed /Current/d
barry [g7 none]:*:349:110:Barry White, Dept 99S:/:/dfs_home/barry:/bin/ksh:
```

6. Check the UDF and the principal definition of barry:

```
# rgy_edit -p
rgy_edit=> v barry -m
  Member of 2 groups:
    none, g7
rgy_edit=> q
# cat dce_users/barry
# --- Principal info:
uuid=0000015d-92c2-2e78-a100-02608c2fff91
uid=349
groups=none g7
  ...
```

barry was a member in the group none before, which was his primary group. Adding a new primary group does not delete his 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 barry
# echo "DEL_groups=none" >> dce_users/barry
# rgy_enable_users barry
# rgy_edit -p
rgy_edit=> v barry -m
  Member of 1 groups:
    g7
rgy_edit=> q
# cat dce_users/barry
# -- !!!!!!!!!!!!!!  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 be logged in as cell_admin.

   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):

```
# cat << EOI >> dce_users/barry
ADD_ACL_INI=dfs#/:/dfs_home/barry#mask_obj:r-x---
ADD_ACL_INI=dfs#/:/dfs_home/barry#user_obj:rwxcid
ADD_ACL_INI=dfs#/:/dfs_home/barry#group_obj:rwx---
ADD_ACL_INI=dfs#/:/dfs_home/barry#other_obj:r-x---
ADD_ACL_INI_OC=dfs#/:/dfs_home/barry#mask_obj:r-x---
ADD_ACL_INI_OC=dfs#/:/dfs_home/barry#user_obj:rwxcid
ADD_ACL_INI_OC=dfs#/:/dfs_home/barry#group_obj:rwx---
ADD_ACL_INI_OC=dfs#/:/dfs_home/barry#other_obj:r-x---
ADD_ACL_INI_CC=dfs#/:/dfs_home/barry#mask_obj:r-x---
ADD_ACL_INI_CC=dfs#/:/dfs_home/barry#user_obj:rwxcid
ADD_ACL_INI_CC=dfs#/:/dfs_home/barry#group_obj:rwx---
ADD_ACL_INI_CC=dfs#/:/dfs_home/barry#other_obj:r-x---
EOI
```

3. Set these ACLs which also contain the initial object and container creation ACLs:

```
# dfs_enable_users barry
Checking to be sure you are cell_admin ...ok
Creating a candidate list of users to dfs_enable ...
  Checking barry ...ok

You are going to dfs_enable       1 users
Starting to work with rgy_edit ...

barry
   DFS-enabling principal barry ...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 users with the following set of entries:

```
ADD_ACL_INI=dfs#/:/dfs_home/barry#group:g99:rwx---
ADD_ACL_INI_OC=dfs#/:/dfs_home/barry#group:g99:rwx---
ADD_ACL_INI_CC=dfs#/:/dfs_home/barry#group:g99:rwx---
```

With instructions like DEL_ACL_INI some of these entries can be deleted. The ACLs for owner, group, and others can 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. Thus, if the user is deleted, the object will probably also be removed by the administrator later on. The other type of ACL entries in the UDFs pull all permissions together which a specific user has on CDS and DFS objects. This makes it possible to remove a user or group with their specific ACL defined anywhere.

To define this other type 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.

4.5.3 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 remove 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 backup 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 us 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 not login 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      5 users
   Starting to work with rgy_edit ...

      Suspending account ben ...ok
      Suspending account brice ...ok
      Suspending account chuck ...ok
      Suspending account felix ...ok
      Suspending account kurt8 ...failed  (Account does not exist)

   *** Ended to suspend users in DCE
   ```

   Apparently there was a UDF in dce_users for which no user exists (anymore). Nevertheless its state will changed to SUSPENDED, and in the next step it will be deleted.

2. Then delete all users:

   ```
   Checking to be sure you are cell_admin ...ok
   Creating a candidate list of users to delete ...

   You are going to delete       6 users
   Starting to work with rgy_edit ...

      Deleting barry ...ACLs...Principal... ok
      Deleting ben ...ACLs...Principal... ok
      Deleting brice ...ACLs...Principal... ok
      Deleting chuck ...ACLs...Principal... ok
      Deleting felix ...ACLs...Principal... ok
      Deleting kurt8 ...ACLs...Principal... failed  (Account does not exist)

   *** Ended to delete users in DCE
   ```

3. The files are now in directory cemetary_users:

```
# cat dce_users/ben
cat: 0652-050 Cannot open dce_users/ben.
# cat cemetary_users/ben
 !!!!!!!!!!!!!!  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 ben
homedir=/:/dfs_home/ben
size=
initprog=/bin/ksh
expir_date=95/09/16
good_since=94/09/16
# --- ACL_INI info:
# --- ACL info:
# --- State and last access:
state=DELETED
last_time_access=Fri Sep 16 11:55:51 CDT 1994  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.

```
rgy_edit -p -v ben
Current site is: registry server at /.../itsc.austin.ibm.com/subsys/dce/sec/master
?(rgy_edit) Cannot retrieve entry for ben - Entry not found (Registry ...
```

4.5.4 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 us create an alias sal for the existing principal pier:

```
# rgy_edit
Current site is: registry server at /.../itsc/subsys/dce/sec/master
rgy_edit=> do p
Domain changed to: principal
 v pier -f
pier                                      107
  Uuid:       0000006b-bc86-2def-b300-10005a4f4165
  Primary:  pr   Reserved:   --
  Quota: unlimited
rgy_edit=> add -al
Add Principal=> Enter name: sal
Enter UNIX number: (auto assign) 107
Enter full name: ()
Enter object creation quota: (unlimited)
rgy_edit=> v sal -f
sal                                       107
  Uuid:       0000006b-bc86-2def-b300-10005a4f4165
  Primary:  al   Reserved:   --
  Quota: unlimited
rgy_edit=>
```

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.

User pier is member of the group staff and the organization itsc:

```
# rgy_edit
rgy_edit=> rgy_edit=> view pier
pier [staff itsc]:*:107:5::/::
rgy_edit=>
```

User sal can now be added as a member of another group and/or organization:

```
rgy_edit=> add sal -g dev -o itsc -pw xyz -mp klr
rgy_edit=> view sal
sal [dev itsc]:*:107:111::/::
rgy_edit=>
```

The only difference is in the group identifier number (5 for pier; 111 for sal). Now modify an ACL for the object /.:/fs:

```
acl_edit -e /.:/fs
sec_acl_edit> m user:pier:r----
sec_acl_edit> m user:sal:rw---
sec_acl_edit> l

# SEC_ACL for /.:/fs:
# Default cell = /.../itsc
unauthenticated:r--t-
user:cell_admin:rwdtc
user:pier:rw---
group:subsys/dce/cds-admin:rwdtc
group:subsys/dce/cds-server:rwdtc
group:subsys/dce/dfs-fs-servers:rwdtc
group:subsys/dce/dfs-admin:rwdtc
any_other:r--t-
sec_acl_edit>
```

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 where the users sal and pier belong:

```
# acl_edit -e /.:/fs
sec_acl_edit> m group:staff:rw---
sec_acl_edit> m group:dev:r----
sec_acl_edit> l

# SEC_ACL for /.:/fs:
# Default cell = /.../itsc
unauthenticated:r--t-
user:cell_admin:rwdtc
user:pier:rw---
group:subsys/dce/cds-admin:rwdtc
group:subsys/dce/cds-server:rwdtc
group:subsys/dce/dfs-fs-servers:rwdtc
group:subsys/dce/dfs-admin:rwdtc
```

```
group:staff:rw---
group:dev:r----
any_other:r--t-
sec_acl_edit>
```

The group staff has read and write permissions while the group dev 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.

4.5.5 A Test with Adding 32,000 Users

The purpose of this section is to describe how our scripts can be used and what experiences we had when we added thousands of users into the DCE security registry.

4.5.5.1 Goals and Results

In this task we added 32,908 users. The reason we decided to add so many users was because we wanted to reach the *Maximum possible UID: 32767* in the registry and see what kind of problems we would find and how we would fix them. This limit is defined in the POSIX standard and is defined as the least common denominator for the different UNIX implementations. Some UNIX implementations use a signed 2-byte integer for the UID.

By adding so many users at once we expected to get a good feeling on how the registry database and related programs such as dce_login behave. In a few words:

- The registry works very well
- We encountered a few problems, which will be discussed
- The overall performance was very good in scenario 1
- When you plan to add so many users, the first thing to do is to extend the default ticket lifetime
- In scenario 1, to add a user took four seconds on average

4.5.5.2 Extending cell_admin′s Ticket Lifetime

Extension of the ticket lifetime can be accomplished in several ways, one is using SMIT:

```
smitty dce ->
  DCE Security & Users Administration ->
     Registry Policies and Properties ->
        Authenticated Policies and Properties
```

and change the following fields from 8h to 1w

```
  DEFAULT ticket lifetime (in hours)                [1w]
  MAXIMUM ticket lifetime (in hours)                [[1w]]
```

If you work on another system and SMIT is not available, use the rgy_edit command as follows:

1. Login as cell_admin
2. Check the current ticket lifetime:

```
klist cell_admin
  ......
Identity Info Expires: 94/06/01:03:39:26
  ......
```

3. Call rgy_edit to change the lifetime:

```
$ rgy_edit
Current site is: registry server at /.../itsc.austin.ibm.com/subsys/dce/sec/master
rgy_edit=> pro
  Properties:
    Properties for Registry at:              /.../itsc.austin.ibm.com
    Registry is NOT read-only
    Certificates to this server may be generated at any site.
    Encrypted passwords are hidden
    ...............
    Default certificate lifetime             10h
Do you wish to make changes [y/n]? (n) y
    .............
Default certificate lifetime (hours): (10h) 168
rgy_edit=> au
  Authentication Policy:
    Max certificate lifetime:                1d
    Max renewable lifetime:                  4w
Do you wish to make changes [y/n]? (n) y
Enter maximum certificate lifetime in hours or 'forever': (1d) 168
Enter maximum certificate-renewable lifetime in hours or 'forever': (4w)
Do you wish to make changes [y/n]? (n)
rgy_edit=> quit
$
```

4. Logout from DCE with the exit command and login again with the dce_login command. Running the kinit command would not do the job. You must exit from the current session and dce_login again, if you want to renew the ticket with the current extended lifetime.

5. Check the current ticket lifetime again. The lifetime has been extended to 1 week.

```
# date
Wed Jun  1 08:31:13 CDT 1994
# klist
  ......
Identity Info Expires: 94/06/08:08:31:06
  .....
```

4.5.5.3 Running the add_users Procedure

The add_users procedure is fully described in 5.5.4, "Adding Users: add_users" on page 258.

In order for our test to create 33,000 user names, we used the dictionary file /usr/dict/words which has more than 24,000 entries and created the file /tmp/words. Then we added the remaining 8800 by duplicating and appending a "z" to as many words from the same file. We have used roughly the same method as the Center for Information Technology Integration (CITI) at the University of Michigan to add 50,200 entries in the DCE Namespace.

We called our script:

```
add_users /tmp/words
```

Our add_users script stopped four times for different reasons, some of these were beginners′ problems, but we will still share them with the reader.

4.5.5.4 The First Problem: /var/dce File System Full

The first time it stopped, was because the /var/dce file system reached the limit. The last successful entry was:

```
# rgy_edit -v -p
  .....
dropout                                   7002
# klist dropout
  ......
Good since date: 1994/05/28.03:19
  ......
```

Then it started to produce error messages for further entries like the following one:

```
?(rgy_edit) Unable to contact the registry - Registry server unavailable
(Registry Edit Kernel) (dce / sad)
drosophila 84
```

The first entry added by the add_users script was:

```
# rgy_edit -v -p
  .....
AAA                                        123
# klist AAA
  ......
Good since date: 1994/05/27.18:51
  ......
```

The 6879 entries we just added plus the initial 27 entries makes a total of 6906 entries.

The /var/dce filesystem was 20MB and we extended it to 60MB. If we presume that all the other files and directories in the /var/dce directory did not grow during this time, we can conclude, that a default entry for a user account in the registry database is almost 1K bytes.

The elapsed time between the first and the last user creation can be derived by comparing the `Good since date` that the `klist` command delivers for the first and the last user. So it took about four seconds per user creation.

The following is a small caveat that shows how the files grow and how the security registry works.

Management of the Security Registry Files:

The file that initially grows in the /var/dce/security/rgy_data directory is the update_log file. The other files are only updated, when secd checkpoints its in-memory registry to disk. The disk files that compose the security registry are acct, acl, group, org, persons, replicas, rgy, rgy_state.

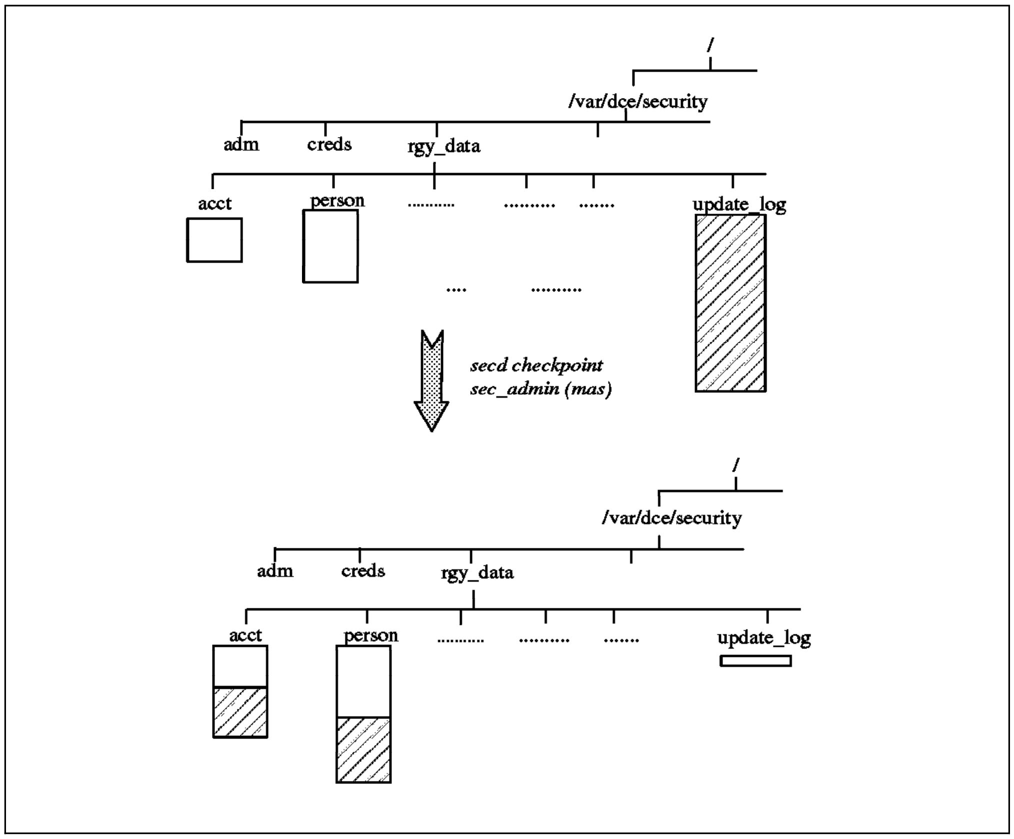


Figure 29. Security Registry Files

secd keeps and updates its database in memory. Changes are logged into the update_log file on disk. The database files are checkpointed from memory to disk every two hours and the update_log is adjusted. In case of a system crash changes since the last checkpoint could be replayed.

You can either await the checkpoint due every two hours for the disk files to be updated or you can run sec_admin with the master_key subcommand to force the update:

```
# sec_admin
Default replica:  /.../itsc/subsys/dce/sec/master
Default cell:     /.../itsc
sec_admin> mas
sec_admin> quit
bye.
```

Taking secd into maintenance mode has the same effect.

4.5.5.5 The Second Problem: The Ticket Lifetime

In order to restart the add_users at the point where it failed before, we had to remove the already successfully added entries from the users file. The second time the procedure then stopped because the ticket lifetime expired on the run. It was set to its default of eight hours. So the script ended after having added another 7481 users (taking four seconds each) with the following error message:

```
?(rgy_edit) Warning - binding is not authenticated - Cant establish
authentication to registry (Registry Edit Kernel) (dce / sad)
Domain changed to: principal
?(rgy_edit) Unable to add principal  "Mississippi" - Principal quota
exhausted (dce / sec )
```

Some additional information which shows with which UIDs this step began and ended, as well as how long it took are shown here:

```
# rgy_edit -v -p
drosophila                                  7145
  .....
  .....
mission                                    14626
#
# klist drosophila
Good since date: 1994/05/29.11:59
# klist mission
  ......
Good since date: 1994/05/29.21:27
  ......
```

It is unclear why from the first stop and the second one the rgy_edit command left out 142 UIDs and started to add the next entry at 7145 instead of 7003. Those UIDs left out from 7003 to 7144 were reused when at the end the rgy_edit arrived at the hard limit of 32,767 accounts. We will describe this later in the discussion of the fourth problem encountered.

We increased the ticket lifetime to 1 week as explained in 4.5.5.2, “Extending cell_admin's Ticket Lifetime” on page 174.

4.5.5.6 The Third Problem: Lack of Paging Space

The third time the script stopped because of lack of paging space. During the third run we got the message:

```
INIT: Paging space low
```

We increased the paging space from 80MB to 108MB

```
# lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto
Type
hd61        hdisk1            rootvg          40MB      83     yes   yes lv
hd6         hdisk0            rootvg          40MB      99     yes   yes lv
# chps -s'5' hd61
# chps -s'2' hd6
# lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto
Type
hd61        hdisk1            rootvg          60MB      58     yes   yes lv
hd6         hdisk0            rootvg          48MB      85     yes   yes lv
```

Since we increased the paging space right in time before the system started to kill processes, we could restart the procedure. After a few more (this time unattended) hours the system started to give the message INIT: Paging space low again on the console and a ps command showed that several kproc processes were in suspended (S) state :

```
# ps
   PID    TTY STAT  TIME COMMAND
     0      - S     1:29 swapper
     1      - S     3:18 /etc/init
   514      - R   904:39 kproc
   771      - S     6:39 kproc
  1028      - S     0:00 kproc
  1352      - S     0:17 /usr/sbin/snmpd
  1614      - S     0:06 /etc/cron
  2150      - S     0:00 kproc
  ....
#
```

After this the rgy_edit started to display the following messages:

```
Current site is: registry server at
/.../itsc.austin.ibm.com/subsys/dce/sec/master
Domain changed to: account
bye.
cyclotronz 0
Cygnusz -0.00390625
cylinderz -0.00390625
cylindricz -0.00390625
  ......
```

Unfortunately not even the shutdown was able to run anymore, so we had to do it the hard way by pushing the reset button of the system. Once the system was up and running again we increased the paging space to 128MB.

4.5.5.7 The Fourth Problem: The Maximum Allowable User ID

So we started the script again and the last problem was when we reached 32,767 accounts. After having reached that limit rgy_edit started to look for unused UIDs in the range of 0..32767. It used what was left over between 7003 and 7144 from the first and second trial. After everything was exhausted it started to display the following error message:

```
Current site is: registry server at
/.../itsc.austin.ibm.com/subsys/dce/sec/master
Domain changed to: principal
?(rgy_edit) Unable to add principal  "floricanz" - UNIX id space for domain
has been exhausted (dce / sec)
bye.
```

After changing the Maximum allowable unix ids the rgy_edit started to add further accounts and reached the limit of 32,908 accounts we had set with no problems.

```
# rgy_edit
rgy_edit=> prop
  Properties:
    ........
    Maximum possible UID:                     32767
    .......
Do you wish to make changes [y/n]? (n) y
  ........
Maximum allowable unix id: (32767) 60000
  .......
rgy_edit=> quit
#
```

4.5.5.8 Conclusions

All the commands like dce_login and rgy_edit were working just fine, the response time was really good even with so many users. The only time we could not run dce_login or rgy_edit was when the /var/dce file system and the paging file system were running out of space. In all other cases we had no problems.

4.5.6 Configuring Single Login/6000

Single Login/6000 is a program offering for AIX DCE. It implements a user login procedure that integrates the standard AIX login and the DCE standard dce_login. See 5.4, “Single Login/6000” on page 235 for product information and distribution information.

This section provides the configuration steps and practical hints based on our testing experience.

In a nutshell, Single Login/6000 provides the users with standard cell wide DCE credentials. Based on these credentials it automatically initiates AIX personal user attributes on the specific local system where the user chooses to login from. As a result the user is concurrently logged in with DCE and with the local AIX under the same UID. The user home directory can be defined to be location independent, that is transparently accessed from any place in the cell, thanks to the DCE/DFS filespace.

For the user administrator, user entries are to be defined only at the cell level with no dependencies on the local AIX systems. This facilitates the management of users as there is no need to define them locally anymore. Single Login/6000 functions take care of this. The cost of this advantage is to manage more entries in the cell namespace for the Single Login/6000 components to work cell wide.

All the configuration operations are supported from SMIT panels associated with Single Login/6000:

```
# smit
  -> Communications Applications and Services
     ->IBM Single Login / 6000
       (fast_path = si_login)
```

This section covers the following topics:

1. Prerequisites and planning information
2. Configuring the Single Login/6000 server
3. Configuring a Single Login/6000 client machine
4. Configuring Single Login/6000 users
5. Release compatibility
6. Using Single Login/6000 as a user
7. User password
8. DFS home directory
9. Password rules customization
10. Failed login and current sessions control
11. Checklist for solving access denied situations

4.5.6.1 Prerequisites and Planning Information

Before you start to configure Single Login/6000 be sure that the following prerequisites are met:

- Your DCE cell is working correctly

- If you are using a free of charge IBM internal version from the AIXTOOLS disk, your cellname must be in the form of an IP domain name ending in ibm.com.

  ```
  # getcellname
  /.../itsc.austin.ibm.com
  ```

- Each Single Login/6000 user has an object in the CDS namespace that takes up 2.1KB disk space per user. If you intend to add a lot of users, be sure the disk space containing the directory /opt/dcelocal/var/directory/cds is large enough.
- If you decide to implement a structured namespace with departments as outlined in 5.4, “Single Login/6000” on page 235, then each department has its own directory in CDS. The users belonging to a certain department have their CDS object in that directory.

  If your user community is really large you might decide to replicate those directories and move the master replica to a department CDS server, provided that your cell core services are structured like that.

  However, distributing CDS directories can be done at any time later on.

4.5.6.2 Configuring the Single Login/6000 Server

There is one machine in the cell which coordinates user login information such as:

- Number of concurrent logins allowed
- Current number of logins
- Locations and client machines of current logins
- Whether a user is allowed to login from all departments (global user)
- Number of recent failed logins

These parameters are configurable and are kept in CDS, see 4.5.6.11, “Failed Login and Current Sessions Control” on page 192.

The task of configuring the Single Login/6000 server consists of:

1. Configuring the server process
2. Starting the server process

After having configured the server process you should follow all steps necessary for client configuration also on the server machine.

Configuring the Server Process:

Call SMIT and follow the indicated path or call SMIT with the fastpath name:

```
#smit
  -> Communications Applications and Services
     -> IBM Single Login / 6000
        -> Configure Single Login / 6000
           -> Configure Single Login / 6000 server machine
              (fast_path = si_cfg_server)
```

```
                 Configure Single Login / 6000 server machine

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

                                                       [Entry Fields]
   Department ID to which this machine belongs?      [dep99]
   (or null)
```

The department ID is for the Single Login/6000 client machine that is automatically configured with the server. The server process has to run on one machine in the cell, the clients run on each machine from which users can login.

Entering a NULL string at the request for a department name is fine, if the cell administrator does not have to split the DCE users community into several subarea′s such as departments or branches. However, departments are useful, if you want to distinguish between users who are allowed to login only from a certain group of machines, the departmental machines, and others who are allowed on all machines in the cell, the global users.

Instead of using SMIT the server can be configured by executing the /usr/bin/mksiserver script:

```
# mksiserver dep99
Enter Your PASSWORD:dce
Enter initial PASSWORD for single/si_client:sil
Enter initial PASSWORD for single/si_server:sil
Current site is: registry server at /.../itsc.austin.ibm.com/subsys/dce/sec/master
Domain changed to: principal
Current site is: registry server at /.../itsc.austin.ibm.com/subsys/dce/sec/master
Domain changed to: principal
Create account for single/si_client
Current site is: registry server at /.../itsc.austin.ibm.com/subsys/dce/sec/master
Domain changed to: account
Create account for single/si_server
Current site is: registry server at /.../itsc.austin.ibm.com/subsys/dce/sec/master
Domain changed to: account
Create key for single/si_client
Current site is: registry server at /.../itsc.austin.ibm.com/subsys/dce/sec/master
Domain changed to: account
Create key for single/si_server
Current site is: registry server at /.../itsc.austin.ibm.com/subsys/dce/sec/master
Domain changed to: account
Create subsystem SI_LOGIN in cds
Create ACL′s for ′/.:/subsys/SI_LOGIN ′
Create ACL′s for ′/.:/subsys/SI_LOGIN - initial container′
Create ACL′s for ′/.:/subsys/SI_LOGIN - initial object′

 - IBM Internal use only -

 Single Login / 6000 - AIX/DCE integrated login

 - IBM Internal use only -

 Press Enter to continue
```

This mksiserver configuration command produces a set of entries in CDS, two new entries in the security registry and two DCE keys for the authentication of the client and server single_login processes as shown by the next three commands :

```
# cdsli -Rod /.:/subsys/SI_LOGIN
/.:/subsys/SI_LOGIN/configuration
/.:/subsys/SI_LOGIN/configuration/password
/.:/subsys/SI_LOGIN/server
/.:/subsys/SI_LOGIN/user

# rgy_edit -v | grep single
single/si_client [none none]:*:10005:12::/::
single/si_server ]none none]:*:10006:12::/::

# echo ktl | rgy_edit | grep single
/.../itsc.austin.ibm.com/single/si_client 1
/.../itsc.austin.ibm.com/single/si_server 1
```

The department name of a client machine is stored in the file /opt/si_login/department.

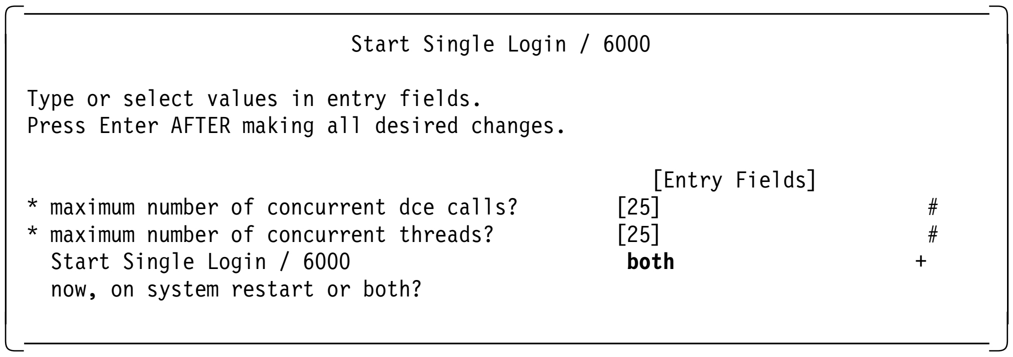
```
$ cat /opt/si_login/department
dep99
```

Starting the Server Process:

The Single Login/6000 service is made of a server process /opt/si_login/single_login_server that permanently runs for the whole cell and of single login clients /opt/si_login/single_login launched when users login with via Single Login/6000.

The SMIT panel for launching the server part (only on the server system) is

```
# smit
   -> Communications Applications and Services
      -> IBM Single Login / 6000
         -> Start Single Login / 6000
            (fastpath = si_start)
```



```
                          Start Single Login / 6000

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

                                                  [Entry Fields]
 * maximum number of concurrent dce calls?       [25]                   #
 * maximum number of concurrent threads?         [25]                   #
   Start Single Login / 6000                      both                  +
   now, on system restart or both?
```

If you expect a lot of users to login at the exact same time you may want to increase the number concurrent threads. However, for each thread the RPC runtime can queue eight requests and the login procedure does not take a long time. The worst case is the user has to login again, when the call fails because the queue is full. If you specify *both* as shown above, the Single Login/6000 server is automatically started on reboot of the machine. If you start DCE automatically, you can do the same with Single Login/6000.

The following command has the same effect:

```
# chgsi_login -U -c25 -t25 -N
```

The -U flag stands for *up* as opposed to the -D flag, which would be used to stop the Single Login/6000 server.

The output of the execution shows :

```
single_login_server: Got the following bindings:
single_login_server: This is binding # 0:
single_login_server: ncacn_ip_tcp:9.3.1.127[2970]
single_login_server: This is binding # 1:
single_login_server: ncadg_ip_udp:9.3.1.127[1185]
single_login_server: This is binding # 2:
single_login_server: ncacn_unix_stream:[/var/dce/rpc/socket/000da232-
d8ec-1df3-987d-10005aa8c755]

single_login_server: Listening ...
```

The command /usr/bin/chgsi_login actually runs as a script that starts the single_login_server process.

```
# ps -ef | grep single
    root  1554  4953   3 21:27:04  pts/1  0:00 grep single
    root 13117     1   0 21:18:13      -  0:00 /opt/si_login/single_logi
n_server -c 25 -t 25
```

The server entries in the namespace are like these :

```
# rpccp show group /.:/subsys/SI_LOGIN/server/SINGLE_GROUP
group members:
  /.../ev8.itsc.austin.ibm.com/subsys/SI_LOGIN/server/single_login_srv
#
# cdscp show object /.:/subsys/SI_LOGIN/server/single_login_srv

              SHOW
              OBJECT   /.../ev8.itsc.austin.ibm.com/subsys/SI_LOGIN/server/s
ingle_login_srv
                  AT   1994-06-06-23:13:37
    RPC_ClassVersion = 0100
             CDS_CTS = 1994-06-07-02:18:20.374048100/10-00-5a-a8-c7-55
             CDS_UTS = 1994-06-07-02:18:21.137307100/10-00-5a-a8-c7-55
           CDS_Class = RPC_Entry
    CDS_ClassVersion = 1.0
          CDS_Towers = :
               Tower = ncacn_ip_tcp:9.3.1.127[]
          CDS_Towers = :
               Tower = ncadg_ip_udp:9.3.1.127[]
```

4.5.6.3 Configuring a Single Login/6000 Client Machine

Configuration of a Single Login/6000 client machine involves the following steps:

1. Configuring the department name and client password
2. Configuring an AIX global user entry

Configuring the Department Name and Client Password:

This particular step has already been done on the Single Login/6000 server machine. So it has to be done only on all pure clients.

Call SMIT and follow the indicated path or call SMIT with the fastpath name:

```
# smit
   -> Communications Applications and Services
      -> IBM Single Login / 6000
         -> Configure Single Login / 6000
            -> Configure Single Login / 6000 client machine
               (fast path = si_cfg_client)
```

```
                   Configure Single Login / 6000 client machine

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

                                                         [Entry Fields]
   Department ID to which this machine belongs?       [dep99]
   (or null)
```

The action behind this panel is run by the script:

```
# mksiclient dep99
Create key for single/si_client
Enter PASSWORD of single/si_client
secret
Current site is: registry server at /.../itsc.austin.ibm.com/subsys/dce/sec/mast
er
Domain changed to: account
          Press Enter to continue
```

If a department name is given, this name is kept in the local file /opt/si_login/department.

Configuration of the AIX Global User Entry:

This configuration step is very much DCE independent. It configures an AIX login entry to which all users first login. It is an AIX account which runs with root authority. The initial program of this global AIX user is the Single Login/6000 client process. There is no chance for the user to get shell access with root permissions.

Call SMIT and follow the indicated path or call SMIT with the fastpath name:

```
# smit
   -> Communications Applications and Services
      -> IBM Single Login / 6000
         -> Single Login / 6000 user administration
            -> Create global AIX login userid
               (fastpath = mksiglobal)
```

```
                          Create global AIX login userid

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

                                                        [Entry Fields]
 * User NAME                                          [dce]
   HOME directory                                     [/home/dce]
```

This panel runs the AIX mkuser command that creates the needed entry in the local AIX user definition files. The associated user entry in /etc/passwd shows:

```
# grep dce /etc/passwd
dce:!:203:200:SingleLogin/6000 global userid:/home/dce:/opt/si_login/single_login
```

4.5.6.4 Defining Departments

If you choose to implement the department concept, you have to define the departments before you can define any users. Department names are reflected in the DCE principal names of users. So, for instance user jacques in a regional office brussels needs to be defined as DCE pricipal brussels/jacques. Since each user gets a CDS object in a CDS directory with the department′s name, this CDS directory must exist first. This is what the following step does.

Call SMIT and follow the indicated path or call SMIT with the fastpath name:

```
# smit
   -> Communications Applications and Services
      -> IBM Single Login / 6000
         -> Single Login / 6000 user administration
            -> Create department
               (fastpath = mksidept)
```

```
                                Create department

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

                                                      [Entry Fields]
 * Department ID?                                   [dep99]
```

The following command has the same effect:

```
cdscp create dir /.:/subsys/SI_LOGIN/user/dep99
```

4.5.6.5 Configuring Single Login/6000 Users

If you choose to implement the department concept, you have to define the departments before you can define any users. See 4.5.6.4, "Defining Departments" on how to define departments. The user should be defined in DCE first. For instance for user jacques in a regional office brussels you need to define the DCE pricipal brussels/jacques. See "Adding Users for Single Login/6000" on page 166 for an example on how to use our user management tool with Single Login/6000 users.

Call SMIT and follow the indicated path or call SMIT with the fastpath name:

```
# smit
   -> Communications Applications and Services
      -> IBM Single Login / 6000
         -> Single Login / 6000 user administration
            -> Create user
               (fastpath = mksiuser)
```

```
                                  Create user

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

                                              [Entry Fields]
 * user name (department/userid)?          [brussels/jacques]
 * max. number of logins (0..50)?          [1]                          #
   global login allowed?                    yes                         +
```

The global attribute in the user definitions specifies whether a user is allowed to login from any machine in the cell or even from a foreign cell. If this is set to yes, he can login from wherever he wants. If it is set to no, he can only login from any client machine of his own department (brussels).

The following command line option adds user jim and allows him to be concurrently logged in 5 times:

```
# mksiuser -u dep99/jim -m 5 -gyes
```

This is how this Single Login/6000 information is recorded for each user in an associated object in the CDS namespace:

```
# cdscp show object /.:/subsys/SI_LOGIN/user/dep99/jim

                SHOW
              OBJECT   /.../itsc.austin.ibm.com/subsys/SI_LOGIN/user/dep99/jim
                  AT   1994-06-21-10:55:21
             CDS_CTS = 1994-06-21-15:54:35.701247100/10-00-5a-a8-c7-55
             CDS_UTS = 1994-06-21-15:54:36.231790100/10-00-5a-a8-c7-55
      Login_MaxLogin = 5
  Login_CurrentLogin = 0
    Login_AccessTime = none
     Login_DataChain = none
        Login_global = yes
  Login_invalid_Logins = 0
```

4.5.6.6 Release Compatibility

If the Single Login/6000 compiled for AIX DCE 1.2 is used with AIX DCE V.1.3, we encountered the problem that on the Single Login/6000 server machine users could not login, because the Single Login/6000 client could not deal with the local socket.

This might not be a problem anymore by now, but we want to mention it just in case. We were using Single Login V.1.05.

This is an example of what happens:

```
# su - dce
 ...
Login:
Cellname [without /.../]:
Department:dep99
Username:jim
Password:

Sorry
Sorry
```

The error reported for this problem shows the cause of the problem:

```
# errpt -a | pg

---------------------------------------------------------------------------
ERROR LABEL:    SI_LOGIN_CLIENT
ERROR ID:       244E48C4

Date/Time:       Mon Jun  6 21:42:02
Sequence Number: 18050
Machine Id:      000005081800
Node Id:         ev8
Error Class:     S
Error Type:      TEMP
Resource Name:   Single Login

Error Description
SOFTWARE PROGRAM ERROR

Probable Causes
SOFTWARE PROGRAM

Failure Causes
SOFTWARE PROGRAM

        Recommended Actions
        REVIEW DETAILED DATA
        CORRECT CONFIGURATION

Detail Data
Additional information:
Single Login 01.0x  Communication error with si_login_start() - comm_st = 16c9a02c
---------------------------------------------------------------------------
```

This error appears because the Single Login/6000 code we ran was not compiled for AIX DCE Version 1.3. So the usage of stream socket binding information was not available. The way around this is to remove the third binding in the endpoint mapper associated with Single Login/6000. It shows:

```
# rpccp show mapping | tail -5
  <object>         nil
  <interface id>   00693246-d4b0-1a98-8579-10005a4fde08,1.0
  <string binding> ncacn_unix_stream:[/var/dce/rpc/socket/000da232
-d8ec-1df3-987d-10005aa8c755]
  <annotation>     SI_SERVER
```

The removal can be done manually as shown here or automatically by running the complementory script `rmsi_stream_mapping` delivered with the diskette in this publication.

```
# rpccp remove mapping -i 00693246-d4b0-1a98-8579-10005a4fde08,1.0 -b \
ncacn_unix_stream:[/var/dce/rpc/socket/000da232-d8ec-1df3-987d-\
10005aa8c755]

>>> all matching mappings removed
```

4.5.6.7 Using Single Login/6000 as a User

This section summarizes the Single Login/6000 procedure as perceived by the user.

Let us assume that the global AIX user ID is dce without a password:

```
AIX Version 3
(C) Copyrights by IBM and by others 1982, 1993.
login: dce
```

This AIX global user account is permanent in the /etc/password file of all systems where Single Login/6000 users are authorized to login from.

```
# cat /etc/passwd | grep dce
dce:!:203:200:SingleLogin/6000 global userid:/home/dce:/opt/si_login/single_login
```

The associated initial program is the Single Login/6000 client program. The client part communicates directly with the DCE security server to obtain network credentials and then communicates with the Single Login/6000 server to update the CDS attributes. This program runs with root permissions. If interrupted, the user gets back to the AIX login prompt. They cannot obtain root authority in a shell.

/opt/si_login/single_login prompts the user for the cellname (for the local cell just press **enter**) the department name (if the user has no department, just press **enter**) and the username, as illustrated here:

```
Login:
        Cellname [without /.../]:
        Department: dep99
        Username: jim
```

The DCE account for this user had been defined by the administrator as follows :

```
$ rgy_edit -v c1
dep99/jim [none none]:*:2006:12:This is jim of
dep99:/:/dfs_home/dep99/jim:/bin/ksh:
```

Single Login/6000 automatically brings the user into the DCE defined home directory and gives the user in AIX the same UID as defined in DCE:

```
$ pwd
/:/dfs_home/dep99/jim
$ id
uid=2006(jim) gid=12
$ klist | grep P
        Global Principal: /.../itsc.austin.ibm.com/dep99/jim
        Principal: 000007d6-b58f-2df3-b100-10005aa8c755 dep99/jim
Passwd Expires:        never
```

The coincidence of the UIDs achieved by automatically creating the equivalent entry in the /etc/passwd file as shown here:

```
$ cat /etc/passwd | grep c1
c1:*:2006:12:SingleLogin/6000 defined user::
```

Thus the user becomes logged in with AIX and with DCE as shown by the session attributes:

```
$ who am i;id
c1          pts/3       Jun 06 22:11          (ev8)
uid=2006(c1) gid=12
```

This is important for accounting and auditing on each local AIX system. See also 5.4.3, “AIX and Single Login/6000” on page 238 for more information about these topics.

When the users login for the first time in a certain system, the DCE account information is stored in the local registry. Should the security server not be available, such users can login to the local system via the local registry. However, they do not have network credentials and their home directory is not accessible if defined in DFS.

4.5.6.8 DFS Home Directory

When working with Single Login/6000 definition of user home directories in DFS is possible, because at the point when the login procedure makes the home directory the current directory, it is accessible by the user. They have already obtained their network credentials.

On the other hand a DFS home directory is also needed to present the same environment to a user no matter from which system he actually logs in.

It is no problem to define a DFS directory for a Single Login/6000 user. The directory has to be specified in the DCE account and it has to be created by the DFS administrator before the user first logs in.

4.5.6.9 Password Rules Customization

The Single Login package provides a set of password rules whose restrictions are set up by the administrator via the next SMIT panel:

```
# smit
   -> Communications Applications and Services
      -> IBM Single Login / 6000
         -> Configure Single Login / 6000
            -> Change password restrictions
               (fast path = si_chpwcfg)
```

```
                             Change password restrictions

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


 * minimum length of password?                         [0]
 * minimum number of different characters?             [0]
 * minimum number of alpha characters?                 [0]
 * minimum number of numeric characters?               [0]
 * maximum number of repetitions?                      [9]
```

To list the current password restrictions use SMIT (fastpath =lspwconf) or simply run the command `lspwconf` as shown here:

```
# lspwconf
 ...
Login_Passwd_mindiff    = 0
  Login_Passwd_minalpha   = 0
  Login_Passwd_minnumber  = 0
  Login_Passwd_minlen     = 0
  Login_Passwd_maxrepeats = 9
```

The parameters of the rules are kept in the CDS object named /.:/subsys/SI_LOGIN/configuration/password.

```
#  cdscp show object /.:/subsys/SI_LOGIN/configuration/password

                    SHOW
                  OBJECT   /.../itsc.austin.ibm.com/subsys/SI_LOGIN/config
uration/password
                      AT   1994-06-22-14:29:53
                 CDS_CTS = 1994-06-21-15:51:43.883875100/10-00-5a-a8-c7-55
                 CDS_UTS = 1994-06-21-15:51:44.362058100/10-00-5a-a8-c7-55
    Login_Passwd_mindiff = 0
   Login_Passwd_minalpha = 0
  Login_Passwd_minnumber = 0
     Login_Passwd_minlen = 0
 Login_Passwd_maxrepeats = 9
```

The password rules are global within a cell, they apply to all users. Before Single Login/6000 grants access, it checks the expiration date of the password as defined in the DCE account. If it is expired, the user is prompted to change the password by Single Login/6000. The standard dce_login displays a message but does not force any change. Before the new password is set, Single Login/6000 checks all the above password rules and continues to request a new password until the rules are met.

4.5.6.10 User Password

The user password in the Single Login/6000 environment is kept in the DCE Security registry and not in local AIX user files. When a user or the administrator has to change a user password, the SMIT DCE change password panel has to be used, which is accessible from two paths:

```
# smit
   -> IBM Single Login / 6000
      -> Single Login / 6000 user administration
         -> DCE Passwords
            (fastpath = chpass)
```

```
# smit
   -> DCE (Distributed Computing Environment)
      -> DCE Security & Users Administration
         -> Passwords
            (fastpath = chpass)
```

The local AIX password does not exist and does not have to be taken into account. If the user tries to set or change a local AIX password he will not be authorized to do so, because there is no valid old password for the change.

```
$ id
uid=2007(c2) gid=12
$ passwd
Changing password for ″c2″
c2′s Old password:
3004-604 Your entry does not match the old password.
3004-664 You are not authorized to change ″c2′s″ password.
```

If the administrator has to provide the user with a command line level operation for changing the password, the smit.script file illustrates the script commands to be executed. Basically a read of the current and new password, a check of the DCE identity and a `rgy_edit` command similar to the next line can be required:

```
cat << EOF | rgy_edit
domain account
change -p $principal_name -g $grp_name -o $org_name -pw ″$PASSWD″ -mp ″$MYPASSWD″
quit
EOF
```

There is usually no need for a user to explicitly request a change for their password, unless they feel the password is not safe enough. Single Login/6000 checks the expiration date as defined in the DCE account and requests a new password from the user if necessary. The password rules as configured in 4.5.6.9, "Password Rules Customization" on page 190 are enforced through Single Login/6000. The traditional `dce_login` never enforces a change.

4.5.6.11 Failed Login and Current Sessions Control

For the administrator, the Single Login/6000 package provides information about the number of cell wide failed logins for each user and the current number and location of sessions used by a user. This information is stored as attributes to each user′s CDS object as shown by the following output:

```
$ cdscp show object /.:/subsys/SI_LOGIN/user/c1

                SHOW
              OBJECT   /.../aix.itsc.austin.ibm.com/subsys/SI_LOGIN/user/c1
                  AT   1994-06-23-10:16:07
             CDS_CTS = 1994-06-22-22:57:13.809865100/02-60-8c-2e-c8-23
            CDS_UTS2 = 1994-06-23-15:15:19.546378100/02-60-8c-2e-c8-23
      Login_MaxLogin = 15
  Login_CurrentLogin = 3
    Login_AccessTime = Thu Jun 23 10:15:19 1994
     Login_DataChain = ev7.itsc.austin.ibm.com since Wed Jun 22 18:00:08 1994
on pts/7 (tx9);ev7.itsc.austin.ibm.com since Wed Jun 22 18:06:30 1994 on pts/
9 (tx9);ev7.itsc.austin.ibm.com since Thu Jun 23 10:15:19 1994 on pts/10 (tx9);
        Login_global = yes
Login_invalid_Logins = 0
```

- Login_CurrentLogin

  This attribute shows that user c1 has 3 current sessions in the cell.

- Login_AccessTime

  This attribute shows the access time of the last login session the user had. If the user cancels the last session, the value of the attribute is not changed as it keeps showing the last time the user logged in even if he logged out from the session.

- Login_DataChain

This attribute shows information about the currently running sessions of the user. This useful information presents where the user is working from. It displays cellname, client hostname, and terminal information as seen on the client host (for instance pts/10 (tx9)). The starting time of each session is also given. Thanks to this cell wide location information, the administrator can track down all user activities in the cell.

- Login_invalid_Logins

  This attribute shows the administrator, if the user failed to login due to specifying a wrong password and how often he unsuccessfully tried. As soon as the user presents the correct password and gets a login session, the Login_invalid_Logins is reset.

  The user may have unsuccessful logins for configuration reasons not related to the password. Those non-password related unsuccessful trials are not counted in the invalid login counter. See the checklist that follows in the next section for hints on managing potential configuration problems.

4.5.6.12 Checklist for Solving Access Denied Situations

Login can fail for a variety of reasons, but Single Login/6000 does not display any reason, it just says *sorry*. From a security point of view, this is correct. So a hacker who tries to break into a system does not get any hints as to whether they have found an existing account name or whether they just missed giving a correct password.

However, from a debugging point of view, it would be desirable to get a reason for the failure. Once Single Login/6000 is configured and working correctly, no other errors than typing errors by users should occur. If other errors such as communication errors occur, Single Login/6000 creates an entry to the AIX error log. Check this log by typing:

```
errpt -a
```

For your convenience we provide a checklist you can follow to find problems associated with failed user logins that are not just typing errors or an incorrect password.

1. Is the DCE account still valid?

   ```
   # dce_login c1 dce
   ```

2. Is the user entry defined in CDS?

   ```
   # cdsli -o /.:/subsys/SI_LOGIN/user/c1
   ```

3. Are the attributes of the object OK?

   ```
   # cdscp show object /.:/subsys/SI_LOGIN/user/c1
   ```

4. Are the ACLs set to the object OK?

   ```
   # acl_edit -e /.:/subsys/SI_LOGIN/user/c1 -l
   ```

5. Is the user home directory defined with their DCE account?

   ```
   # rgy_edit -v c1
   ```

6. Is the user home directory created?

   ```
   # ls -d /:/si_users/global/c1
   ```

7. Is the user home directory owned by the user or UID of the DCE user account?

   ```
   # ls -ld /:/si_users/global/c1 ; rgy_edit -v c1
   ```

8. Is the user initial program (for example /bin/ksh) defined with their DCE account?

   ```
   # rgy_edit -v c1
   ```

9. Is the Single Login server registered in the Namespace?

   ```
   # cdsli -Ro | grep single_login_srv
   ```

10. Is the Single Login server process running on the server machine?

    ```
    # ps -ef | grep single_login_server
    ```

11. Has the Single Login client been configured?

    ```
    # echo ktl | rgy_edit
    ```

12. Is the password key of the Single Login client valid?

    ```
    # dce_login /.../itsc.austin.ibm.com/single/si_client
    ```

13. Is the local department definition correct?

    ```
    # cat /opt/si_login/department
    ```

14. Is your Single Login/6000 version compiled for AIX DCE 1.3 and if no, is the local stream socket removed?

    ```
    # rpccp show mapping | pg
    ```

4.6 Managing the cell_admin Account

cell_admin is per 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 has accidentally been removed
3. Adding new cell_admin accounts

This last procedure also shows you where cell_admin needs to be defined to have all its rights. If you do not like the fact that one single account is omnipotent, you can assign the rights to several other special accounts.

4.6.1 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:

1. Kill the security daemon secd:

   ```
   kill -9 `ps xv | grep secd | grep -v grep | awk '{print $1}'`
   ```

2. Call 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 to start the dce_login session.

4. Login to DCE with the new password:

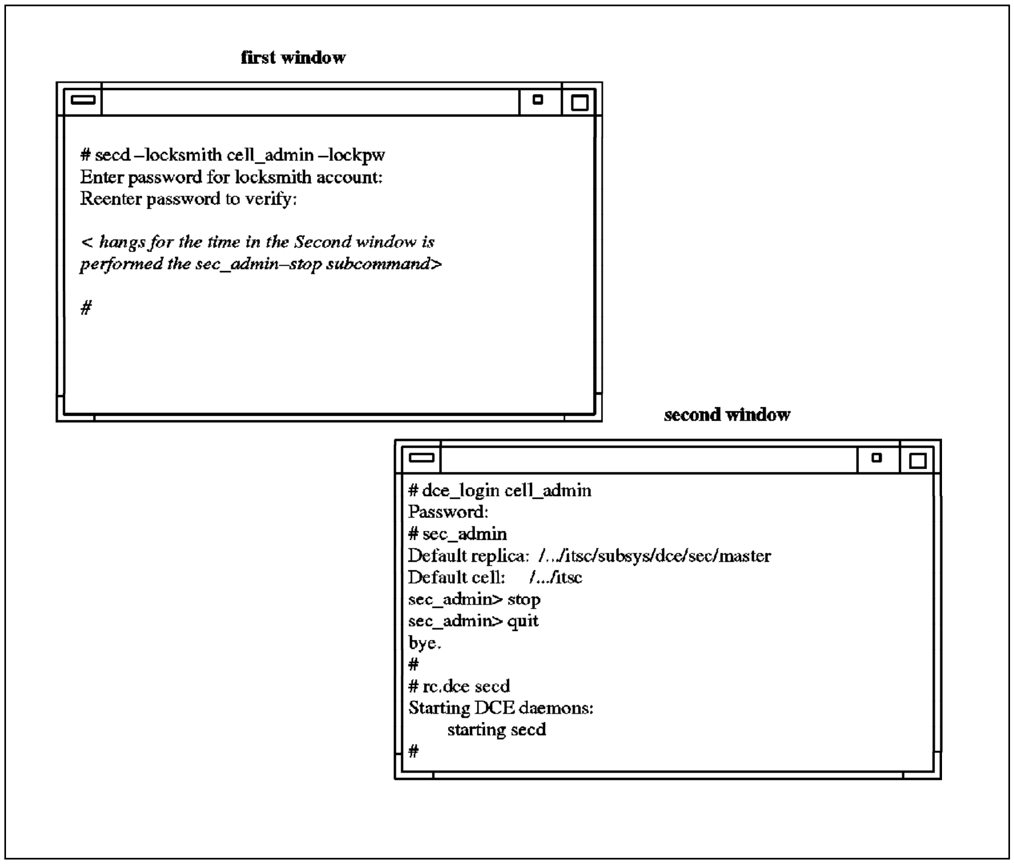
```
dce_login cell_admin
Enter Password: <NEW PASSWORD: not-echoed password>
```

5. You must stop secd, which is still running in the background:

```
# sec_admin
Default replica:  /.../itsc/subsys/dce/sec/master
Default cell:     /.../itsc
sec_admin> stop
sec_admin> quit
bye.
#
```

6. Restart secd from SMIT, with `rc.dce secd` or simply calling secd from the command line.

7. Login again with cell_admin using the new password.



```
# secd –locksmith cell_admin –lockpw
Enter password for locksmith account:
Reenter password to verify:

< hangs for the time in the Second window is
performed the sec_admin–stop subcommand>

#
```

```
# dce_login cell_admin
Password:
# sec_admin
Default replica:  /.../itsc/subsys/dce/sec/master
Default cell:     /.../itsc
sec_admin> stop
sec_admin> quit
bye.
#
# rc.dce secd
Starting DCE daemons:
      starting secd
#
```

Figure 30. How to Restore a Lost cell_admin Password

4.6.2 Cell Administrator Accidentally Removed

Accidentally a cell administrator might remove their own user ID:

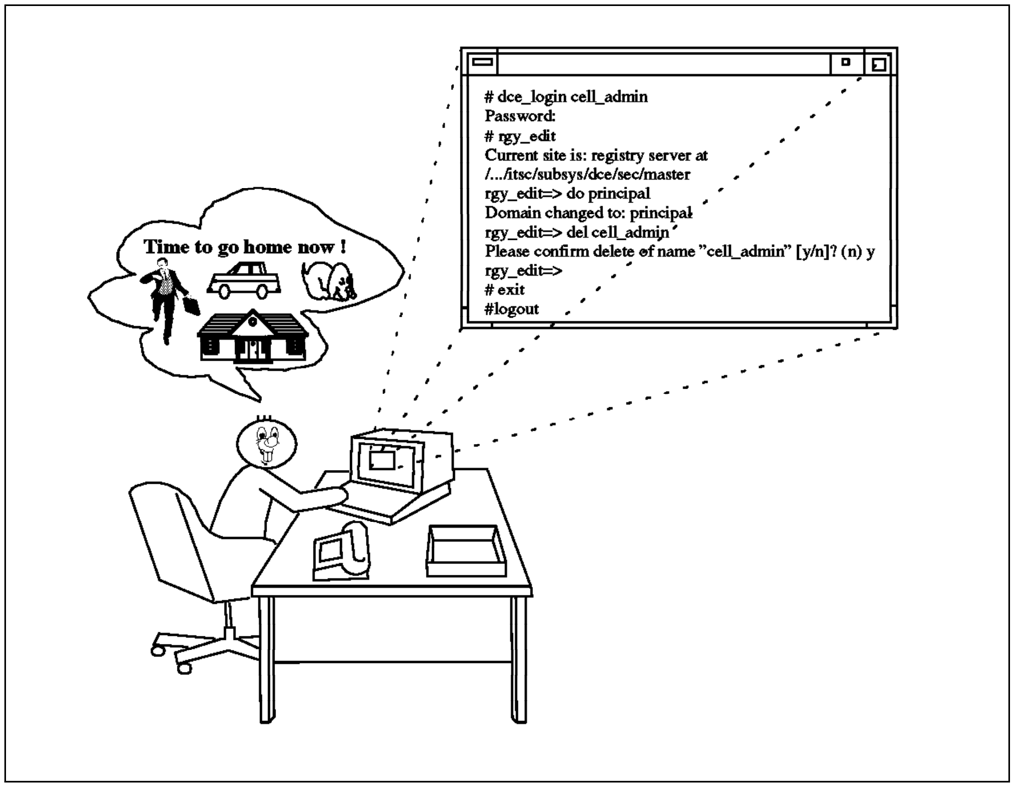
```
# dce_login cell_admin
Password:
# rgy_edit
Current site is: registry server at /.../itsc/subsys/dce/sec/master
rgy_edit=> do principal
Domain changed to: principal
```

```
rgy_edit=> del cell_admin
Please confirm delete of name ″cell_admin″ [y/n]? (n) y
rgy_edit=>
# exit
#
```

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)
#
```

Suppose that you as cell administrator delete yourself and go home:



```
# dce_login cell_admin
Password:
# rgy_edit
Current site is: registry server at
/.../itsc/subsys/dce/sec/master
rgy_edit=> do principal
Domain changed to: principal
rgy_edit=> del cell_admin
Please confirm delete of name "cell_admin" [y/n]? (n) y
rgy_edit=>
# exit
#logout
```

Figure 31. What to Do When cell_admin is Deleted?

The day after, you realize what happened and find yourself in a bad situation.

Let us 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 must be recreated now:

```
# rgy_edit
rgy_edit=> do p
Domain changed to: principal
rgy_edit=> view -members
nobody                                     -2
  Member of 1 groups:
    nogroup
            ...........
cell_admin                                100
  Member of 7 groups:
    none, acct-admin, subsys/dce/sec-admin, subsys/dce/cds-admin
    subsys/dce/dts-admin, subsys/dce/dfs-admin, subsys/dce/dskl-admin
# rgy_edit=> do a
Domain changed to: account
rgy_edit=> view -f cell_admin
cell_admin [none none]:*:100:12::/::
  created by: /.../itsc/cell_admin  1994/06/03.13:38
  changed by: /.../itsc/cell_admin  1994/06/03.13:38
  password is: NOT valid, was last changed: 1994/06/03.13:38
  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: 1994/06/03.13:38
  Max certificate lifetime: default-policy
  Max renewable lifetime: default-policy
rgy_edit=>
```

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
/.:/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 5.5, "User (and ACL) Management" on page 242 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.

Here are the steps you have to perform in order to recover from this situation:

1. Be sure you are root on the local system.
2. Kill the security daemon on the security server:

   ```
   kill -9 `ps xv | grep secd | grep -v grep | awk '{print $1}'`
   ```

3. 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>
```

4. 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 to start the dce_login session.

5. Run dce_login command for the cell_admin user:

```
# dce_login cell_admin
Enter Password:  <NEW PASSWORD: not-echoed password>
#
```

6. Run the rgy_edit command and start to update the user cell_admin. A cell administrator has been created from the maintenance option locksmith as follows:

```
rgy_edit=> view cell_admin -f
cell_admin [none none]:*:104:12::/::
  created by: /.../itsc/dce-rgy  1994/06/03.12:02
  changed by: /.../itsc/dce-rgy  1994/06/03.12:02
  password is: valid, was last changed: 1994/06/03.12:02
  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: 1994/06/03.12:02
  Max certificate lifetime: default-policy
  Max renewable lifetime: default-policy
rgy_edit=>
```

You might notice that the user has been created by the dce-rgy principal and not by cell_admin.

7. Make cell_admin a member of the groups listed below:

   - acct-admin
   - subsys/dce/sec-admin
   - subsys/dce/cds-admin
   - subsys/dce/dts-admin
   - subsys/dce/dfs-admin
   - subsys/dce/dskl-admin

```
rgy_edit=> do group
Domain changed to: group
rgy_edit=> m acct-admin
Enter name to add: cell_admin
Enter name to add:
Enter name to remove:
rgy_edit=>
rgy_edit=> m subsys/dce/sec-admin
Enter name to add: cell_admin
Enter name to add:
Enter name to remove:
rgy_edit=> m subsys/dce/cds-admin
```

```
Enter name to add: cell_admin
Enter name to add:
Enter name to remove:
rgy_edit=> m subsys/dce/dts-admin
Enter name to add: cell_admin
Enter name to add:
Enter name to remove:
rgy_edit=> m subsys/dce/dfs-admin
Enter name to add: cell_admin
Enter name to add:
Enter name to remove:
rgy_edit=> m subsys/dce/dskl-admin
Enter name to add: cell_admin
Enter name to add:
Enter name to remove:
rgy_edit=>
rgy_edit=> quit
#
```

8. Exit from your current session and login again as cell_admin:

```
# exit
# dce_login cell_admin
Password:
#
```

9. Update the ACL entries for all the directories and objects with the following commands:

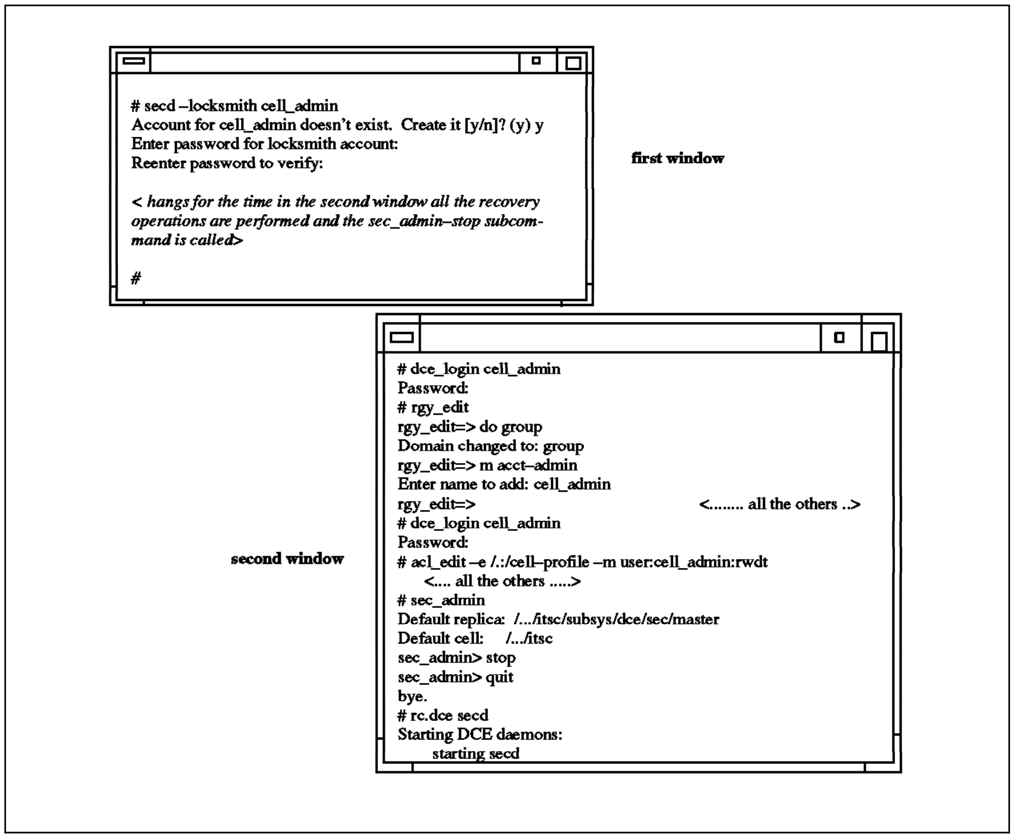
```
# acl_edit -e /.:/cell-profile -m user:cell_admin:rwdtc
# acl_edit -e /.:/fs -m user:cell_admin:rwdtc
# acl_edit -e /.:/lan-profile -m user:cell_admin:rwdtc
# acl_edit -e /.:/sec -m user:cell_admin:rwdtc
# acl_edit -e /.:/sec-v1 -m user:cell_admin:rwdtc
# acl_edit -e /.:/hosts -m user:cell_admin:rwdtc
# acl_edit -e /.:/hosts/hostname -m user:cell_admin:rwdtc
# acl_edit -e /.:/hosts/hostname/cds-clerk -m user:cell_admin:rwdtc
# acl_edit -e /.:/hosts/hostname/cds-server -m user:cell_admin:rwdtc
# acl_edit -e /.:/hosts/hostname/profile -m user:cell_admin:rwdtc
# acl_edit -e /.:/hosts/hostname/self -m user:cell_admin:rwdtc
# acl_edit -e /.:/subsys -m user:cell_admin:rwdtc
# acl_edit -e /.:/subsys/dce -m user:cell_admin:rwdtc
# acl_edit -e /.:/subsys/dce/dfs -m user:cell_admin:rwdtc
# acl_edit -e /.:/subsys/dce/dfs/bak -m user:cell_admin:rwdtc
# acl_edit -e /.:/subsys/dce/sec -m user:cell_admin:rwdtc
```

10. Run sec_admin command with the stop subcommand to stop the security daemon which is still running in maintenance mode in the other window or in the background:

```
# sec_admin
Default replica:  /.../itsc/subsys/dce/sec/master
Default cell:     /.../itsc
sec_admin> stop
sec_admin> quit
bye.
#
```

11. Start up the security daemon again and start working in normal mode:

```
# rc.dce secd
Starting DCE daemons:
        starting secd
#
```



```
# secd –locksmith cell_admin
Account for cell_admin doesn't exist.  Create it [y/n]? (y) y
Enter password for locksmith account:
Reenter password to verify:

< hangs for the time in the second window all the recovery
operations are performed and the sec_admin–stop subcom-
mand is called>

#
```

```
# dce_login cell_admin
Password:
# rgy_edit
rgy_edit=> do group
Domain changed to: group
rgy_edit=> m acct–admin
Enter name to add: cell_admin
rgy_edit=>                              <........ all the others ..>
# dce_login cell_admin
Password:
# acl_edit –e /.:/cell–profile –m user:cell_admin:rwdt
     <..... all the others .....>
# sec_admin
Default replica:  /.../itsc/subsys/dce/sec/master
Default cell:     /.../itsc
sec_admin> stop
sec_admin> quit
bye.
# rc.dce secd
Starting DCE daemons:
      starting secd
```

Figure 32. Recreating the cell_admin Account

4.6.3 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.

4.6.2, "Cell Administrator Accidentally Removed" on page 195 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 5.5, "User (and ACL) Management" on page 242 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 create the new_admin account.

To add a new cell_administrator do the following:

1. Login as cell_admin:

2. Add a new principal:

```
# rgy_edit
rgy_edit=> do p
Domain changed to: principal
rgy_edit=> add new_admin
```

3. Add a new account:

```
rgy_edit=> do a
Domain changed to: account
rgy_edit=> add
Add Account=> Enter account id [pname]: new_admin
Enter account group [gname]: none
Enter account organization [oname]: none
Enter password:
Retype password:
Enter your password:
Enter misc info: () New Administrator
Enter home directory: (/) /home/new_admin
Enter shell: () /bin/ksh
Password valid [y/n]? (y)
Enter expiration date [yy/mm/dd or 'none']: (none)
Allow account to be server principal [y/n]? (y)
Allow account to be client principal [y/n]? (y)
Account valid for login [y/n]? (y)
Allow account to obtain post-dated certificates [y/n]? (n)
Allow account to obtain forwardable certificates [y/n]? (y)
Allow certificates to this account to be issued via TGT authentication [y/n]? (y
)
Allow account to obtain renewable certificates [y/n]? (y)
Allow account to obtain proxiable certificates [y/n]? (n)
Allow account to obtain duplicate session keys [y/n]? (n)
Good since date [yy/mm/dd]: (1994/06/03.16:28)
Create/Change auth policy for this acct [y/n]? (n)
Add Account=> Enter account id [pname]:
```

4. Add new_admin to the necessary groups:

```
rgy_edit=> do group
Domain changed to: group
rgy_edit=> m acct-admin
Enter name to add: new_admin
Enter name to add:
Enter name to remove:
rgy_edit=>
rgy_edit=> m subsys/dce/sec-admin
Enter name to add: new_admin
Enter name to add:
Enter name to remove:
rgy_edit=> m subsys/dce/cds-admin
Enter name to add: new_admin
Enter name to add:
Enter name to remove:
rgy_edit=> m subsys/dce/dts-admin
Enter name to add: new_admin
Enter name to add:
Enter name to remove:
rgy_edit=> m subsys/dce/dfs-admin
Enter name to add: new_admin
Enter name to add:
Enter name to remove:
rgy_edit=> m subsys/dce/dskl-admin
```

```
Enter name to add: new_admin
Enter name to add:
Enter name to remove:
rgy_edit=>
rgy_edit=> quit
```

5. Create ACL entries for the following CDS objects:

```
# acl_edit -e /.:/cell-profile -m user:new_admin:rwdtc
# acl_edit -e /.:/fs -m user:new_admin:rwdtc
# acl_edit -e /.:/lan-profile -m user:new_admin:rwdtc
# acl_edit -e /.:/sec -m user:new_admin:rwdtc
# acl_edit -e /.:/sec-v1 -m user:new_admin:rwdtc
# acl_edit -e /.:/hosts -m user:new_admin:rwdtc
# acl_edit -e /.:/hosts/hostname -m user:new_admin:rwdtc
# acl_edit -e /.:/hosts/hostname/cds-clerk -m user:new_admin:rwdtc
# acl_edit -e /.:/hosts/hostname/cds-server -m user:new_admin:rwdtc
# acl_edit -e /.:/hosts/hostname/profile -m user:new_admin:rwdtc
# acl_edit -e /.:/hosts/hostname/self -m user:new_admin:rwdtc
# acl_edit -e /.:/subsys -m user:new_admin:rwdtc
# acl_edit -e /.:/subsys/dce -m user:new_admin:rwdtc
# acl_edit -e /.:/subsys/dce/dfs -m user:new_admin:rwdtc
# acl_edit -e /.:/subsys/dce/dfs/bak -m user:new_admin:rwdtc
# acl_edit -e /.:/subsys/dce/sec -m user:new_admin:rwdtc
```

6. Now you could delete the old cell_admin:

```
dce_login new_admin
Enter Password:
# rgy_edit
Current site is: registry server at /.../itsc/subsys/dce/sec/master
rgy_edit=> Domain changed to: principal
rgy_edit=> do p
rgy_edit=> del cell_admin
Please confirm delete of name "cell_admin" [y/n]? (n) y
rgy_edit=> quit
bye.
```

From now on the cell_admin is not part of your DCE environment:

```
# dce_login cell_admin
Sorry.
User Identification Failure. - Registry object not found (dce / sec)
#
```

4.7 Integrating an NFS/NIS Environment

The Network Information System (NIS) and Network File System (NFS) are network services which 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 is 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 there local /etc/hosts file.

NFS is a distributed file system. An NFS server has one or more file systems exported, which 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. See 1.3.4, “NIS/NFS” on page 23 for a description of NFS/NIS and how it compares to DCE/DFS.

As outlined in the above referenced section of chapter 1, DCE/DFS serves the same purpose as NFS/NIS but has many advantages. However, it is relatively new and is just about to establish itself as a standard platform for 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

4.7.1 Migrating from NIS Domains to DCE cells

NIS can centrally manage configuration files usually needed on each single system. The files like for instance an /etc/passwd file is present on each system, but it contains 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 lookup 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 which 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 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. Unfortunately DCE is not well enough integrated into AIX yet so that two logins are required, whereas NFS/NFS is fully integrated. There is an optional program offering, Single Login/6000, which uses the DCE registry as a central user repository and offers additional functions for login integration of AIX and DCE. Single Login/6000 is being discussed in more detail in 5.4, “Single Login/6000” on page 235. In 4.7.2, “Migrating Users from NIS to DCE” on page 205 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. /etc/services or /etc/protocols 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 TCP/IP definitions 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.

4.7.1.1 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 an upclient process. 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 could 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 ITSO publication *The Distributed File System (DFS) for AIX/6000* or InfoExplorer.

4.7.1.2 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.

4.7.1.3 Distribution via CDS Namespace

Although it is not recommended to store data in the CDS namespace, there is the possibility of keeping information which has to be consistent throughout the cell in a CDS object. Following is an example for the file /etc/services:

1. Create an object in the namespace:

   ```
   cdscp create object /.:/services
   ```

2. Add a new attribute to the file /etc/dce/cds_attribute:

   ```
   echo "1.3.22.1.3.60   CDS_TCPIP_SERVICES  char" >> /etc/dce/cds_attribute
   ```

   This defines the attribute CDS_TCPIP_SERVICES of type character to be used within your namespace. /etc/dce/cds_attribute is also a file which needs to be consistent on all the systems within the cell.

3. 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

   ```
   # cdscp add object /.:/services CDS_TCPIP_SERVICES=domain:53/tcp:nameserver
   # cdscp add object /.:/services CDS_TCPIP_SERVICES=domain:53/udp:nameserver
   ```

   The colons (:) are used as field delimiters. Process each entry of /etc/services in this way.

4. 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 to 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.

4.7.2 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 5.5, “User (and ACL) Management” on page 242.

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 4.7.2.3, “The nis2dce_users Procedure” on page 207.

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

4.7.2.1 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 for login or to trace activities or access rights. 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 pretty 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 while DCE IDs do not match AIX IDs.

The following is 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.

4.7.2.2 Moving User Accounts and Groups into DCE

Moving 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 5.5, “User (and ACL) Management” on page 242 for explanations on UDF/GDF and the tools.

Assuming you installed the user management scripts in directory /umgt and directory temp_users does not exist yet, 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 above, 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.

4.7.2.3 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 5.5, “User (and ACL) Management” on page 242 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 use WRITE_UDF to write the upper part of the file and eventually write 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"
```

4.7.3 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 new the 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 3, "Implementing DCE Cells" on page 43 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.

4.7.3.1 DFS ACL Inheritance

ACL inheritance is the method by which a file or a directory is given an ACL when they are created. Certain values, we call them Initial Creation ACLs, are defined on the parent directory and are passed 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.

There are three things to consider when the ACL is created for a new object or directory:

1. 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 use the open() system call with permissions `rw-rw-rw`, if they are creating a new file. 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. Initial Creation ACL set for the parent directory

3. 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. For the corresponding entries user_obj, group_obj, mask_obj, and other_obj 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` 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`.

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:

```
cd /:
/: #acl_edit testdir -l -io

# Initial SEC_ACL for objects created under: testdir:
# Default cell = /.../itsc.austin.ibm.com
user_obj:rwxc--
group_obj:r-x---
other_obj:r-x---
/: #umask
022
```

The umask 022 means that write permissions for group and others are masked out. So this ACL corresponds to the umask, which makes it likely that ACLs have not been set. To check, we need to change the umask with the `umask` command and list the ACL again:

```
/: #umask 0
/: #umask
00
/: #touch testdir/t
/: #rm testdir/t
/: #acl_edit testdir -l -io

# Initial SEC_ACL for objects created under: testdir:
# Default cell = /.../itsc.austin.ibm.com
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 `acl_edit` to reflect the new umask value.

We recommend setting the Initial Creation ACLs on the top directory which will receive 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.

4.7.3.2 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 his own fileset. The following generalized example outlines the necessary steps:

1. Login 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. DCE login 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 which 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 which 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.

4.7.4 Configuring DFS Access from NFS Clients

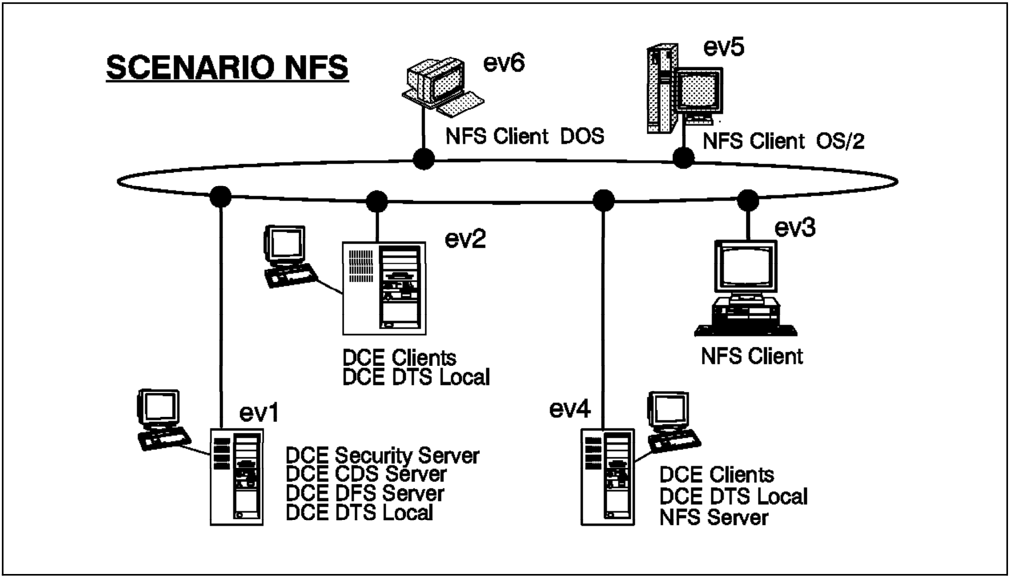


Figure 33. Scenario with Coexistence of NFS Clients and DCE/DFS

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 5.3, "NFS to DFS Authenticating Gateway" on page 227 to understand the basics about the translator.

According to Figure 33 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 4.2.4, "Defining Home Directories in DFS" on page 115 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

4.7.4.1 Preparation Steps

See 4.1.1, "Preparing for DCE Configuration" on page 84. To install the DCE cell follow the configuration steps exactly in 3.1.1, "Scenario 1: All Servers on One Machine without Replicas" on page 45 for machines *ev1, ev2, ev4*.

Install and test TCP/IP and NFS on the OS/2 and the DOS/Windows machines. Please follow the appropriate system documentation.

4.7.4.2 NFS/DFS Translator Configuration Steps

Following are all the configuration steps for this scenario.

***Creating and mounting the additional filesets on** ev1*

1. Create a logical volume /dev/dfshome with 5 blocks of 4MB:

   ```
   #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. Create the dfshome fileset with mount point:

   ```
   #mkdfslfs -f dfshome.ft -m /:/dfshome -n dfshome
   ```

5. Create brice′s fileset with mount point in the same aggregate:

   ```
   #mkdfslfs -f hbrice.ft -m /:/dfshome/brice -n dfshome
   ```

6. Try to see if the filesets are correctly exported:

   ```
   #fts lsfldb
   ```

7. Make brice the owner of the new fileset:

   ```
   chown brice.staff /:/dfshome/brice
   ```

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 yet done. Normally this step is to be done by the owner of the directory.

1. Login as DCE brice principal on *ev4*:

   ```
   $dce_login brice brice_passwd
   ```

2. Apply ACLs on the directory /:/dfshome/brice:

   ```
   $acl_edit /:/brice
   ```

   Inherited rights when you create another subdirectory:

   ```
   $acl_edit /:/brice -ic
   ```

   Inherited rights when you create files underneath this point in the file tree:

   ```
   $acl_edit /:/brice -io
   ```

   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.

Configuring and starting the NFS/DFS Translator on** ev4**:

This step has to be executed by a UNIX system administrator. Be sure that:

- DFS client is configured and running on the machine
- 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.
```

Note

If you do not see this message, the reason might be a bad version of the nfs.ext file in the /etc/drivers directory. This happened to us, but should not happen with the release level product.

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.

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 = dfsnfs)
```

```
                          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.

***Registering your authentication mappings on** ev4:*

As we previously explained: before a directory can be mounted by an NFS client, the owner of the fileset has to 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 -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
```

Mounting 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 /:/brice /u/brice
   ```

2. On an OS/2 system (*ev5*):

   ```
   c>mount E:  ev4:/:/brice
   UID: 107
   GID: 100
   ```

3. On a DOS/Windows system (*ev6*):

   ```
   c>mount E: ev4:/:/brice
   ```

   We assume the disk unit *E:* is defined on the DOS system.

4.8 Configuring DCE on HACMP

The following DCE core services are installed in our test of DCE on HACMP/6000:

```
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
```

According to the DCE memo-to-users that announced IBM AIX High Availability Cluster Multi-Processing/6000 support for DCE on the AIX platform, the following HACMP/6000 Release 2.1 configurations are supported:

- One-for-One Standby Configuration with Owned/Takeover Resources

- One-Sided Takeover Using Owned/Takeover Resources
- One-for-One Standby Configuration with Rotating Resources

In line with this definition of supported HACMP/6000 configurations, we did not operate any highly available DCE service on the standby node prior to fallover.

If you decide to run DCE services like a DCE client and/or DCE application server on the standby node, you must provide a procedure to terminate these prior to takeover.

Configuration requirements for the minimum impact fallover are:

1. IP addresses for DCE services must be configured as Owned/Takeover or Rotating resources
2. Assign the hardware address for the adapter to eliminate the need for ARP cache updates on the DCE client nodes
3. File systems /krb5, /var/dce and /etc/dce must reside on the shared DASD and are configured as Owned/Takeover or Rotating resources

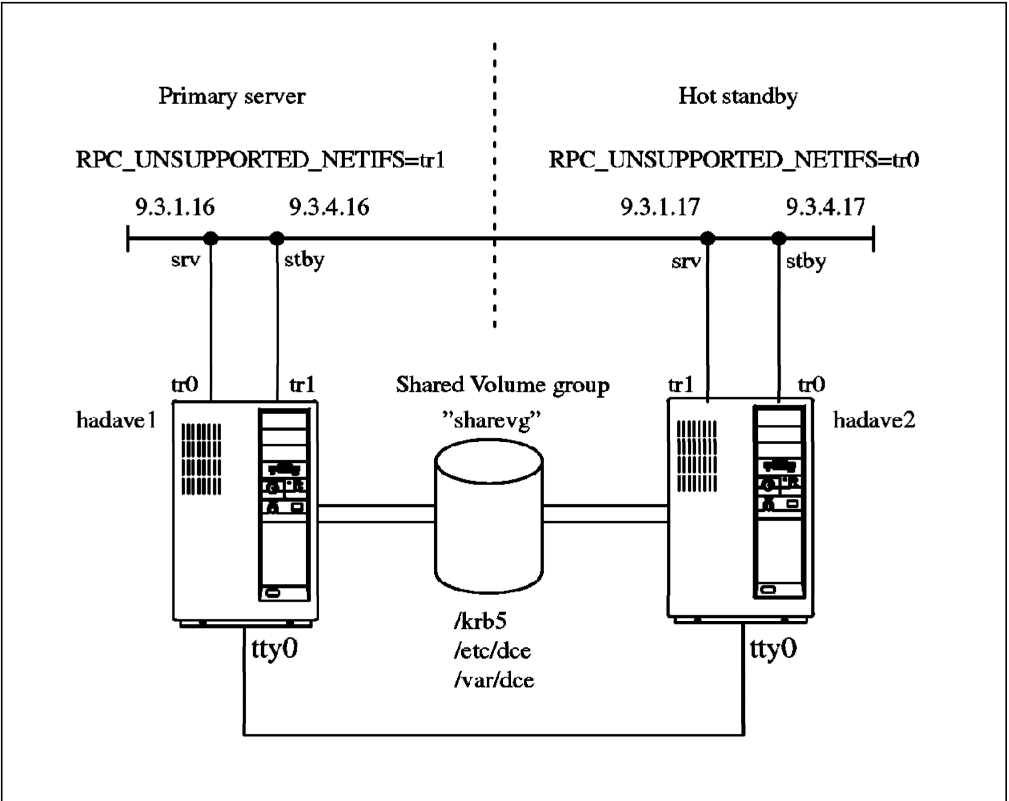


Figure 34. DCE on HACMP/6000. This cluster configuration was used with nodes ev1, ev2, ev3, and ev4 (see scenarios in Chapter 3, "Implementing DCE Cells" on page 43), which were all defined as DCE clients.

The following description shows the steps how to configure and install DCE in an HACMP/6000 cluster:

1. Prior to installing/configuring DCE, HACMP/6000 has to be installed, configured, and tested on your cluster.
2. Create separate logical volumes and file systems for /krb5, /var/dce and /etc/dce on a shared disk.

3. DCE must ignore certain network interfaces used by HACMP/6000 as it initializes. The network interfaces to ignore are the standby interfaces of the primary machine and those used strictly for cluster node keep alive communication. The environment variable RPC_UNSUPPORTED_NETIFS is used for this purpose. You must add it to the file /etc/environment. To take effect, it is necessary to reboot the system. For example, if on the primary node network interfaces tr1 and en1 were used as standby and interface sl0 for cluster node keep alive packets, then the environment variable would have to be set as follows:

   ```
   RPC_UNSUPPORTED_NETIFS=tr1:en1:sl0
   ```

   In our scenario we had the following setting for node hadave1:

   ```
   RPC_UNSUPPORTED_NETIFS=tr1
   ```

   For node hadave2, the standby machine:

   ```
   RPC_UNSUPPORTED_NETIFS=tr0
   ```

4. Use the option to configure the adapters with a user defined hardware address. This enables the adapter that is taking over to use the same IP and hardware address combination. Clients having worked with this server node before have this IP/hardware address combination in their ARP (Address Resolution Protocol) cache. So they will be able to work with the new node after takeover without having to refresh their ARP cache, which would otherwise require manual intervention on all client nodes. Following is our adapter configuration on hadave1:

   ```
                     Change Attributes of Cluster Node or Adapter

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

                                                          [Entry Fields]
     **NOTE: Cluster Manager MUST BE RESTARTED
             in order for changes to be acknowledged.**

     Node ID                                               1
     New Node ID                                          []                  +#
     Adapter IP label                                      hadave1_srv
     New Adapter IP label                                 []
     Adapter function                                      service             +
     Network name                                         [token1]             +
     Network attribute                                     public              +
     Adapter IP address                                   [9.3.1.16]
     Adapter Hardware address                             [0x10005a4f4110]




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

5. Install and configure DCE on the primary node. Be sure the shared volume group with the shared DCE logical volumes is varied on and that you run with the service IP address not with the boot address.

6. After successful installation/configuration, stop the DCE daemons using /etc/dce.clean.

7. On the standby node only DCE installation is required. All necessary configuration information for the takeover case is already on the shared disk.

8. On the standby node add the following entry to the /etc/services file:

```
kerberos5       88/udp          kdc
```

9. HACMP/6000 requires start and stop scripts for every application server to be handled by takeover procedures. It is sufficient to use `rc.dce` and `dce.clean` as DCE start/stop scripts. They perform all the necessary steps. Make sure these scripts are correctly configured in the HACMP/6000.

10. Start the HACMP/6000 cluster manager on the primary node. Once up, start the HACMP/6000 cluster manager on the secondary node. Since the restart of DCE is one of the application server scripts, there is no need to explicitly start DCE, it should come up when the cluster manager on the primary node is started.

Note

This procedure is only valid for AIX DCE Version 1.3 or later, because the DCE startup files were moved from /etc to /etc/dce which is on a shared disk. The filenames in /etc are now symbolic links. In AIX DCE 1.2 you need to copy these files as part of application startup script. Furthermore in AIX DCE 1.2 you need PTF U423300 for support of the environment variable `RPC_UNSUPPORTED_NETIFS`.

Chapter 5. New Tools and Technologies

One purpose of this chapter is to describe what is new in IBM AIX Distributed Computing Environment 1.3. Another purpose is to introduce useful technologies or tools for DCE administrators in more detail than would have been suitable in the previous task-oriented or how-to Chapter 4, “Administering DCE Cells” on page 83.

We will cover the following topics in this chapter:

1. What is new in AIX DCE 1.3
2. DFS file server replication
3. NFS/DFS authentication translator
4. Login integration
5. A mass user management tool
6. HACMP/6000 and DCE

In the previous chapters we showed how to use all these features without explaining the details.

5.1 AIX DCE 1.3 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 extend 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 5.2, “DFS Replication” on page 224 and 5.3, “NFS to DFS Authenticating Gateway” on page 227.

These are the new features which will be explained in the rest of this overview section:

- Security server replication
- DFS fileset replication - see 5.2, “DFS Replication” on page 224
- NFS to DFS Authenticating Gateway - see 5.3, “NFS to DFS Authenticating Gateway” on page 227
- Support for HACMP/6000 - see 5.6.1, “HACMP/6000 Support for DCE” on page 273
- Split configuration
- Local RPCs
- Environment variable RPC_UNSUPPORTED_NETIFS
- Monitoring function in IBM NetView for AIX
- Exportable data encryption facility CDMF
- Stub size reduction
- Preferred file server for DFS clients

5.1.1 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 4.1.6.2, "Replicating the Security Server" on page 99 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 at the first place:

```
# 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, usually you 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.

5.1.2 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 pretty 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.	

5.1.3 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 otherwise 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 which runs a security server. It could as well 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 pathnames 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 /var/dce/rpc/socket. Also by default, the name for each socket file is an object UUID, which ensures the uniqueness of each filename. 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 filenames 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.

5.1.4 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 DCE cell layout. This is discussed in more detail in Chapter 3, “Implementing DCE Cells” on page 43 and 4.1.1, “Preparing for DCE Configuration” on page 84.

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.

5.1.5 Monitoring Function in IBM NetView for AIX

IBM NetView for AIX is the platform for network management based on 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 which 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 thresholds are exceeded.

IBM NetView for AIX provides a GUI to display network topologies, be alerted, and respond to abnormal conditions present 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, which 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 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 indicate fault conditions to the network administrator

5.1.6 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.

5.1.7 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.

5.1.8 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 which could have occurred with random server selection.

5.2 DFS Replication

This subject is already fully covered by the *The Distributed File System (DFS) for AIX/6000*. However, since the availability of this feature is new, we will 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 4.2.3, "Replicating DFS Server" on page 107.

5.2.1 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. 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. Nevertheless, both types of replication produce the same result: 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 concept and the technology are well known but the availability of the product is only in AIX DCE 1.3. For more information about the concept and implementation of this feature, we suggest you to consult the *DCE V1.3 for AIX Administration Guide -- Extended Services* and for more advanced configuration see the ITSO publication *The Distributed File System (DFS) for AIX/6000*. In 4.2.3, "Replicating DFS Server" on page 107 we describe how we set up DFS with replication step-by-step. The following section gives a summary of the DFS replication concepts.

5.2.2 Why Fileset Replication?

One of the advantages of 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 machines outages. If one machine housing a DCE LFS fileset is unavailable, replicated versions of the filesets are still available from other machines.

- 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.

5.2.3 Which Files to Replicate?

Even though we can replicate any fileset, we have to be careful what we replicate. 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.

5.2.4 How Does Replication Work?

In order to become accessible by DFS clients, filesets need to be mounted. Mount points have to be created in the DFS filespace. Figure 35 on page 226 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 which are fileset mount points.

There are two types of mount points which 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 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 its parent directories.

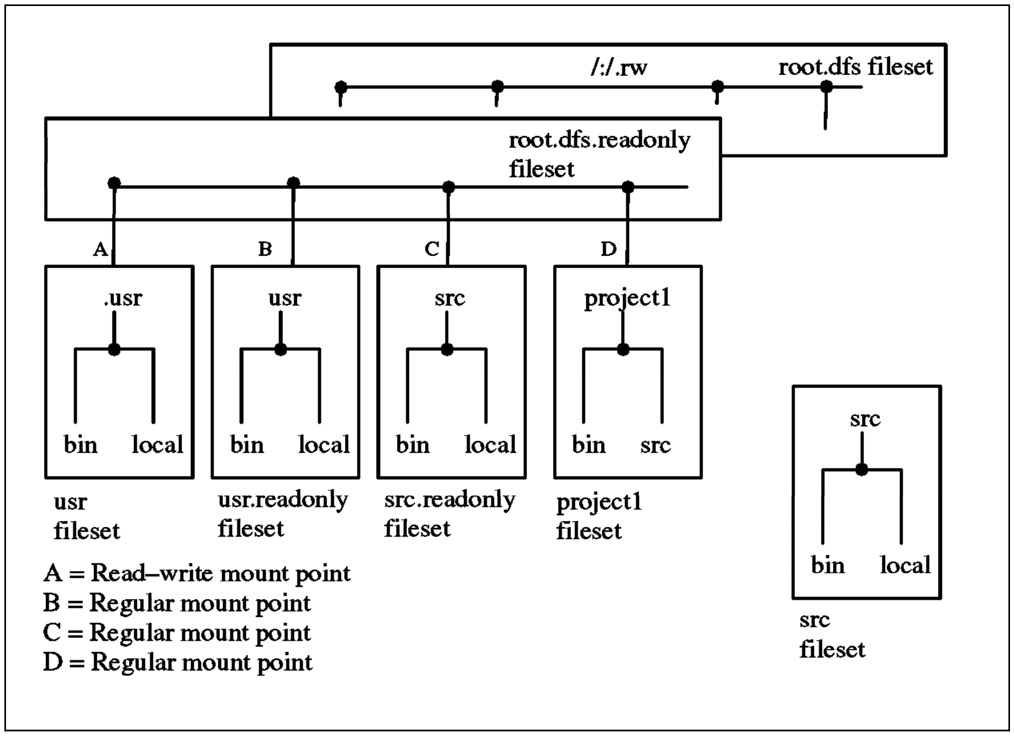


Figure 35. DFS Hierarchy File System

We can see in Figure 35 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 35:

- /:/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 pathname /:/.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.

DFS clients are caching fileset data, but also fileset information, which means path name 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 4.2.3, “Replicating DFS Server” on page 107.

5.3 NFS to DFS Authenticating Gateway

This section provides the description of the NFS to DFS Authenticating Gateway, sometimes also called 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 4.7.4, “Configuring DFS Access from NFS Clients” on page 211.

5.3.1 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 new 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 36 on page 228.

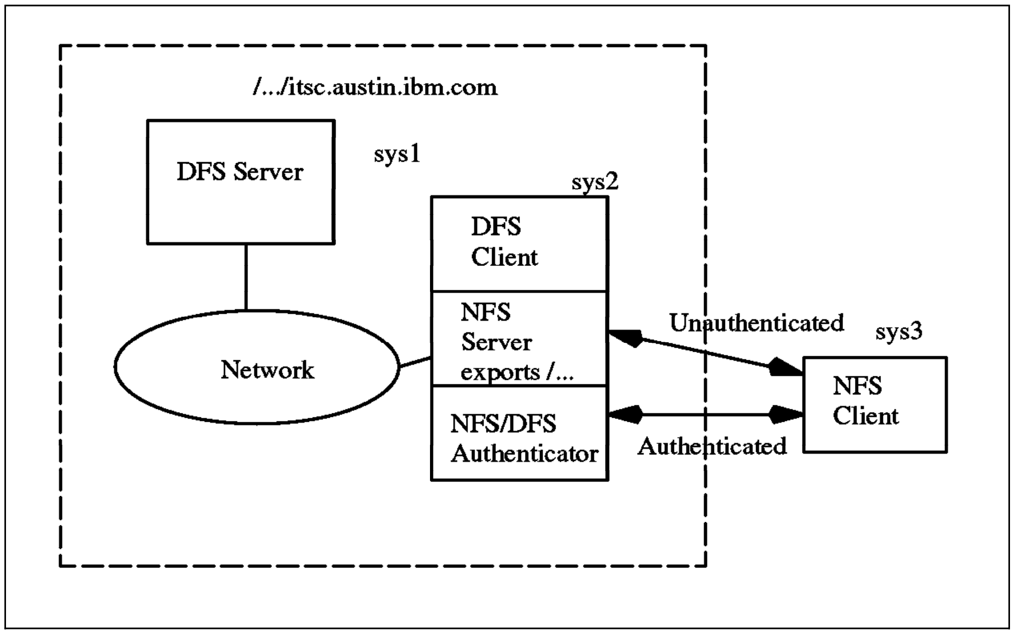


Figure 36. DFS/NFS Translator Architecture

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.

5.3.2 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, 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 not available either to NFS clients.

The real goal of the NFS/DFS Translator is to provide authenticated access to the DFS filespace from NFS client machines which do not need any extra software.

5.3.3 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.

5.3.3.1 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 which are not exported to user processes, the translator must also reside inside the kernel. On AIX Version 3, this is achieved by adding a kernel extension.

DFS like NFS is a layer underneath 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.

5.3.3.2 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:

- 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 login 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 would be started 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 his DCE credentials expire, he will start to experience access permission errors because he becomes an authenticated user. To re-authenticate, the user must use the translator registration command once again to register the mapping. Then he can continue to access the DFS filespace.

On the NFS client machine, you don′t need to unmount and remount the file system.

5.3.4 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 5.3.5, "Administration Tasks for the DFS User" on page 232.

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 5.3.6.1, "Using the Translator from a UNIX NFS Client Machine" on page 234.

The following are the tasks of a system administrator to perform 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

5.3.4.1 Installing and Starting NFS/DFS Translator

The first step is to install the package ″dcedfsnfs.obj″ which includes:

- AIX kernel extension for the NFS/DFS Translator
- dfsiauth command to register(add), delete, and list authentication mappings
- libdceiauth.a user library
- dcedfs/dceiauthapps.h include file for application development
- System Management Interface Tool (SMIT) screens for NFS/DFS Translator management

Then provide (start) NFS/DFS Translator service:

1. Verify DCE/DFS and NFS are configured and running on your system.
2. Load 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 startups. It should be added after the rc.dce and rc.nfs lines in the inittab file, because it needs them to be active.

5.3.4.2 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 in 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.

5.3.4.3 Removing Expired Authentication Mappings

Credentials can expire. See also 5.3.5.6, “Managing Expired Authentication Mappings” on page 234 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
```

5.3.4.4 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.

5.3.4.5 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 4.7, “Integrating an NFS/NIS Environment” on page 202 which demonstrates ways of making UIDs unique.

5.3.5 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 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 of 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

5.3.5.1 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 login into that system, either locally or remotely. Before using this command, you must have an account in the DCE security registry and be logged in to DCE.

```
$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

For more information, we suggest reviewing the *DCE NFS to DFS Authenticating Gateway V1.3 for AIX*.

5.3.5.2 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 which maps the NFS client machine′s IP address and remote UID to a DCE principal which 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.

5.3.5.3 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
```

5.3.5.4 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
```

5.3.5.5 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 pathname 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 NFS to DFS Authenticating Gateway V1.3 for AIX*.

5.3.5.6 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.

5.3.6 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
- Login 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.

5.3.6.1 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
```

5.3.6.2 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
```

Issue the command:

```
c>mount E: nfs_server_machine:/:/remote_dir
```

5.3.6.3 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
```

5.4 Single Login/6000

Single Login/6000 is a complementary solution offering for DCE and AIX user integration. It was developed and enhanced for several customer projects in the banking area. DCE offers an excellent security service. However, it is not integrated into AIX and the standard DCE login command `dce_login` does not enforce any password policy.

Single Login/6000 was developed on the user API level and is not integrated into the AIX kernel nor in any AIX authentication method. That is why accounting and especially auditing is not 100 percent proper. Most auditing events are reported to the correct user ID, though. See 5.4.3, “AIX and Single Login/6000” on page 238 for more details about this.

However, there is a long list of features contained in Single Login/6000, which make it a superior solution to the standard `dce_login` command. It contains everything needed to make DCE user administration and control easier in a large DCE environment such as:

- Only one (integrated) login procedure for AIX and DCE
- Users are centrally managed (defined only once in DCE)
- Cellwide control over the number of logins for a user
- Cellwide control who is logged in where
- User gets an AIX account on any machine they log in to
- Home directory in DFS
- Supports a structured user namespace with a department concept
- XDM support
- Support for xlock and xautolock
- Configurable password restrictions
- Support for local accounting and auditing
- Account locked after configurable amount of failed logins
- Support for inter-cell login
- Full NLS support
- Full installp support
- Full SMIT support

For more information (more details, latest versions, ordering, prices) contact the following address:

```
IBM Deutschland Entwicklung GmbH
Customized Banking Technology
att. Hans-Juergen Dittgen
     Schoenaicher Str. 220
     71032 Boeblingen
         Germany
```

```
Tel.  xx49-7031-16-4867
Fax   xx49-7031-16-4572
IBMMAIL: DEIBMQVR at IBMMAIL
Vnet: DITTGEN at BOEVM4
```

The following sections give more information about Single Login/6000:

- 5.4.1, "Single Login/6000 Features Overview"
- 5.4.2, "Possible Enhancements for Single Login/6000" on page 238
- 5.4.3, "AIX and Single Login/6000" on page 238

In 4.5.6, "Configuring Single Login/6000" on page 180 we give instructions on how to configure the Single Login/6000 environment and users.

5.4.1 Single Login/6000 Features Overview

The following are the product highlights as listed in the above introductory section:

- Only one (integrated) login procedure for AIX and DCE

  The user has to login only once. He always has the same UID in DCE and AIX.

- Users are centrally managed (defined only once in DCE)

  Users are defined only once. You do not have to define each user as a principal in the DCE registry *and* in every AIX system. Each user is defined in the DCE registry only.

- Cell wide control over the number of logins for a user

  The number of network wide logins for a user can be limited and controlled. For example if the maximum number of logins for a user is defined as two, then Single Login/6000 enforces this limit cell wide and the user can login no more than twice or from two machines in the cell.

- Cell wide control who is logged in where

  Each defined Single Login/6000 user has a CDS object to store information when and where they logged in. So, an administrator can check where in the cell users are logged by listing CDS contents.

- The user gets an AIX account on any machine they log in to

  1. The user is authenticated against the DCE registry
  2. Password expiration date is checked; a change is enforced if necessary
  3. Password restrictions are applied and checked for the new password
  4. Login time and location are centrally registered
  5. AIX account information is generated
  6. Login is granted

- Home directory in DFS

  It is not impossible to have a DFS home directory without Single Login/6000, but it is much easier than with the standard `dce_login` command, where DFS is not available for the user at the time AIX makes their home directory the current directory.

- Supports a structured user namespace with a department concept

  Single Login/6000 supports a structured user namespace. Departments or regional offices can be defined. If you decide to implement departments, then each DCE node needs to be defined in a department. When users are added, you need to specify to which department they belong. They can then login to any machine in their department. If they are defined as global users, they can even login on any machine in the cell.

Single Login/6000 users have a CDS entry to control their current location and number of logins. The Single Login/6000 CDS namespace is also structured according to the departments. Each department has a CDS directory and its users have a CDS object in that directory. The DCE principal and account name reflects this structure, it is a qualified name, which would allow for multiple occurrences of the same name in different departments.

- XDM support

  The Single Login/6000 login dialog is started when a user logs in with a global AIX user ID. The Single Login/6000 client program is defined as the initial program of the global AIX user. This program is started when the user logs in from an ASCII login screen.

  If the global AIX user account is being used for login via XDM, the .Xsession file is used to start the Motif version of the Single Login/6000 login dialog, which does the same job as the ASCII version. It includes a dialog for password renewal.

  Current support is for XDM release 11.5.

- Support for xlock and xautolock

  Support for the xlock function is separately available. It checks the password with the DCE registry rather than with the local /etc/passwd files. In addition, an xautolock function is available which automatically invokes xlock after a certain time of inactivity.

- Configurable password restrictions

  Password rules such as minimum length, minimum number of alphabetic characters, minimum number of characters different from last password, and maximum number of repetitions of a character can be configured and are valid cellwide. When the user is requested to change his password, these rules are enforced.

- Support for local accounting and auditing

  Since the AIX login is started with a global user ID, some of the actions are reported to that ID. As soon as the DCE authentication is over and the AIX effective UID is set, most actions are correctly reported.

  See also 5.4.3, "AIX and Single Login/6000" on page 238 for information on AIX auditing with the use of Single Login/6000.

- Account locked after configurable amount of failed logins

  The maximum number of consecutive failed logins can be configured cellwide. When a user login fails because an invalid password was presented, the counter is incremented for that user. Since it is administered in a CDS object, it is enforced cellwide. Once the user succeeds to login before his account is locked, his counter is reset.

- Support for inter-cell login

  If users are defined as global users, they are allowed to login from any machine, even from machines in a foreign cell. On the Single Login/6000 login dialog they must specify their home cell, which is the cell in which they have an account in the security registry. The request goes to the home cell, and when they are authenticated there and are global users, they get access to the AIX machine at which they started the login procedure. They get an /etc/passwd entry just as they would on a machine in their home cell. However, since they are not working in their home cell, their network credentials are those of foreign users for this cell.

- Full NLS support
- Full installp support

  Single Login/6000 is delivered as an installp object with product history information.
- Full SMIT support

  All Single Login/6000 administrative commands are supported with SMIT panels.

5.4.2 Possible Enhancements for Single Login/6000

When we tested Single Login/6000 we came across some important items that need to be enhanced to make it more highly available and not just usable on the AIX platform.

According to the developers of the product these items can be done without too much effort, but it would only be done on a request basis.

- OS/2 support

  OS/2 does not need an integrated login because it is a single user system. However, for all the other nice features such as the overview on who is logged into the cell, for the password rules and restrictions, and the department concepts of Single Login/6000, the client needs to be ported to OS/2.
- Support for other DCE platforms

  Single Login/6000 is written with standard DCE calls, so porting to other platforms should be no big deal.
- Single Login/6000 server replication

  The Single Login/6000 server is exported to the CDS namespace in an RPC group. From that perspective it would be easy to implement replication. A second server could be started which exports its interface into CDS and the client would randomly get one or the other server addresses.

  However, since the Single Login/6000 server writes to CDS objects and holds a context handle with each Single Login/6000 client, some changes would be necessary in the way CDS is updated.

  Today, if the Single Login/6000 server is not available, the users can still login, but the central user information is not updated. If the Single Login/6000 server fails and comes back to operation, the login counters are reset.

5.4.3 AIX and Single Login/6000

Single Login/6000 is integrated in AIX at the user level. There are no hooks into the kernel. As described in 4.5.6, “Configuring Single Login/6000” on page 180 a global user in the /etc/passwd file is created, which is shared by all users:

```
dce:!:203:200:SingleLogin/6000 global userid:/home/dce:/opt/si_login/single_login
```

The initial program of this global user activates the client portion of Single Login/6000. Generally this user has no password, so after entering the user name dce, the user directly gets to the Single Login/6000 login dialog. We experienced several small problems, but overall the application was working quite well. We configured four users a1, a2, a3 and a4.

If you run `last`, you will only see the user dce logged in, no evidence of the real user working in the system.

```
root    pts/2      9.3.1.74       Wed Jun 01 19:06   still logged in.
dce     pts/15     ev1            Wed Jun 01 19:03 - 19:06  (00:02)
dce     pts/13     ev1            Wed Jun 01 19:00 - 19:06  (00:05)
dce     pts/12     ev1            Wed Jun 01 18:58 - 19:06  (00:07)
dce     pts/11     ev1            Wed Jun 01 18:56 - 19:06  (00:09)
dce     pts/11     ev1            Wed Jun 01 18:53 - 18:54  (00:00)
dce     pts/10     ev1            Wed Jun 01 18:51 - 19:06  (00:14)
dce     pts/8      ev1            Wed Jun 01 18:50 - 19:06  (00:16)
```

So, possibly all the account subsystem commands might have the same problem, we did not test accounting any further.

When the user a4 enters into the system, a line in the /etc/passwd is generated

```
a4:*:204:12:Single Login / 6000 defined user::
```

No entry or stanza has been added to the /etc/security/passwd for user a4. So, a user who wants to login into AIX using a4 will be rejected:

```
AIX Version 3
(C) Copyrights by IBM and by others 1982, 1993.
login: a4
a4's Password:
3004-007 You entered an invalid login name or password.
login:
```

The audit subsystem generated some messages that might mislead system administrators who are used to running audit in AIX. We configured audit in the /etc/security/audit/config file for the users dce, a1, a2, a3 and a4 and verified, if some of the kernel, authentication, file system, and command events in the /etc/security/audit/events were accounted to the right user. We have always used the same command:

```
# auditstream | auditselect -e "(event == EVENT_TO_TRACE  || event == USER_Login) \
  && (login == dce || login == a4)" | auditpr  -t0 -heRl
```

where EVENT_TO_TRACE can be one of the following:

- PROC_AuditID
- PROC_RealUID
- PROC_RealGID
- PROC_Environ
- PROC_Create
- PROC_Delete
- PROC_Execute
- PROC_SetSignal
- PROC_Limits
- PROC_SetPri

- PROC_Privilege
- USER_Logout
- USER_SU
- CHECK_INITTAB

We did this for the user names a4 and dce. Here are the results:

```
# auditstream | auditselect -e ″(event==PROC_AuditID  || event==USER_Login) \
  && (login == dce || login == a4)″ | auditpr  -t0 -heRl
USER_Login      OK          dce
PROC_AuditID    OK          dce
PROC_AuditID    OK          a4
# auditstream | auditselect -e ″(event==PROC_RealUID || event==USER_Login) \
  && (login == dce || login == a4)″ | auditpr>
USER_Login      OK          dce
PROC_RealUID    OK          a4
PROC_RealUID    OK          a4
# auditstream | auditselect -e ″(event==PROC_RealGID || event==USER_Login) \
&& (login == dce || login == a4)″ | auditpr>
USER_Login      OK          dce
PROC_RealGID    OK          dce
# auditstream | auditselect -e ″(event==PROC_Environ || event==USER_Login) \
  && (login == dce || login == a4)″ | auditpr>
USER_Login      OK          dce
PROC_Environ    OK          dce
PROC_Environ    OK          dce
PROC_Environ    OK          dce
# auditstream | auditselect -e ″(event==PROC_Create || event==USER_Login) \
  && (login == dce || login == a4)″ | auditpr >
PROC_Create     OK          a4
USER_Login      OK          dce
PROC_Create     OK          dce
PROC_Create     OK          a4
PROC_Create     OK          a4
# auditstream | auditselect -e ″(event==PROC_Delete || event==USER_Login) \
  && (login == dce || login == a4)″ | auditpr >
USER_Login      OK          dce
PROC_Delete     OK          a4
PROC_Delete     OK          a4
        <logout>
PROC_Delete     OK          a4
PROC_Delete     OK          dce
PROC_Delete     OK          a4
# auditstream | auditselect -e ″(event==PROC_Execute || event==USER_Login) \
  && (login == dce || login == a4)″ | auditpr>
PROC_Execute    OK          a4
USER_Login      OK          dce
PROC_Execute    OK          dce
PROC_Execute    OK          dce
PROC_Execute    OK          a4
PROC_Execute    OK          a4
PROC_Execute    OK          a4
# auditstream | auditselect -e ″(event==PROC_SetSignal || event==USER_Login) \
   && (login == dce || login == a4)″ | audit
PROC_SetSignal  OK          dce
PROC_SetSignal  OK          dce
       .....
   < a lot of messages of this only type  for both dce and a4 >
PROC_SetSignal  OK         dce
PROC_SetSignal  OK         a4
PROC_SetSignal  OK         a4
# auditstream | auditselect -e ″(event==PROC_Limits || event==USER_Login) \
  && (login == dce || login == a4)″ | auditpr >
USER_Login      OK         dce
# auditstream | auditselect -e ″(event==PROC_SetPri || event==USER_Login) \
  && (login == dce || login == a4)″ | auditpr >
USER_Login      OK         dce
PROC_SetPri     OK         dce
PROC_SetPri     OK         dce
PROC_SetPri     OK         dce
PROC_SetPri     OK         dce
PROC_SetPri     OK         a4
# auditstream | auditselect -e ″(event==PROC_Privilege || event==USER_Login) \
  && (login == dce || login == a4)″ | audit>
PROC_Privilege  OK         dce
PROC_Privilege  OK         dce
PROC_Privilege  OK         dce
USER_Login      OK         dce
PROC_Privilege  OK         dce
# auditstream | auditselect -e ″(event==USER_Logout || event==USER_Login) \
  && (login == dce || login == a4)″ | auditpr >
USER_Login      OK         dce
# auditstream | auditselect -e ″(event==USER_SU || event==USER_Login) \
  && (login == dce || login == a4)″ | auditpr -t0 >
USER_Login      OK         dce
USER_SU         OK         a4
# auditstream | auditselect -e ″event == CHECK_INITTAB && login == a4″ | \
auditpr -t0 -heRl
CHECK_INITTAB    OK    a4
```

As you can see from the results, some of the events give messages that might be misleading. Sometimes only user dce is accounted for some actions and not a4. For instance the USER_Login event is always associated with user dce and never with a4. The USER_Logout event never shows up, when a4 logs out, not even under user dce.

When the user a4 tries to call the su command, it gets correctly accounted for it. When a4 tries to read the /etc/inittab file, the event CHECK_INITTAB is associated to him and not to dce.

The last event CHECK_INITTAB is not part of the preconfigured events of AIX. We added it. It records an event when somebody tries to look into the /etc/inittab file.

Overall the audit works fine, but the consistency of the event messages is not among the best features of this application. Single Login/6000 sets the real UID, but not the login UID. A system administrator who uses audit in AIX has to set up new events. This would enable him to track down when a user logs in and out or might help him to find a new way to interpret some of the kernel events, which are now associated partially to dce and partially to a4.

The file system and command events such as CHECK_INITTAB and USER_SU are associated correctly with user a4.

An additional suggestion to the author of this application or the system administrator of a DCE cell is to add all the si_login command files and programs to the AIX Trusted Computing Base (TCB). This would have to be done in the /etc/security/sysck.cfg database. After that the TCB bit has to be set to *on* for si_login in the following way:

```
# chtcb query single_login
single_login is not in the TCB
# chtcb on single_login
# chtcb query single_login
single_login is in the TCB
#
```

5.5 User (and ACL) Management

User management encompasses tasks such as adding, modifying, 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 33 for planning information.

Group Management

What is mentioned about users and UIDs throughout this chapter is also true for groups and group identifiers (GIDs). They have to correspond between DCE and all involved UNIX systems. If you migrate from other environments, the GIDs have to be unified and defined in DCE with the same number they had in the old environment.

Our tools were first made and described for user management only. They were only supposed to show a way for solving the problem of mass user management. Since it was a last minute decision to also create the group management commands for your convenience, they could not be documented in the same detail as the user management commands. A separate section 5.5.3, "Group Management" on page 254 will briefly describe the concepts and commands for group management.

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 not practical anymore. The following situations require tools that are able to handle a lot of users:

- First DCE installation
- Migration from 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 other tasks such as managing groups and Single Login/6000 users.

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 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 4.5, "Administering Users and Groups" on page 161 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.

5.5.1 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 /var/dce/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 universal unique identification (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 they look the name up in the registry database and the user *name* is displayed. 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.

Also stored with each user and group are their UNIX or their operating system user (UID) and group (GID) identifiers. UNIX commands consider the UID/GID to determine who the user is and what rights he has. UNIX commands internally use the UID but for display they enter the /etc/passwd file with that UID to determine the user name. They *do not* look the name up in the DCE registry database. This is what sometimes causes so much confusion in the current DCE implementation which is not integrated into AIX. If UUID and UID do not match, for instance DFS files do not seem to belong to the correct owner when looked at with UNIX 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 match the UID/GID defined for them on any possible UNIX system in the cell.

A package such as Single Login/6000 can ease that burden (see 5.4, “Single Login/6000” on page 235).

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.

5.5.2 Management Tool Structure and Overview

DCE provides a central point of administration through the use of different tools such as:

- `rgy_edit` to centrally administer the registry database
- `acl_edit` to administer the ACLs

These tools are line commands and quite primitive. They are the standard tools supported by all the vendors. However, AIX provides a better interface through the use of SMIT. The next OSF DCE release (1.1) will provide a DCE shell which will incorporate all the different standard tools. We have built a number of commands using the standard DCE commands to manage accounts.

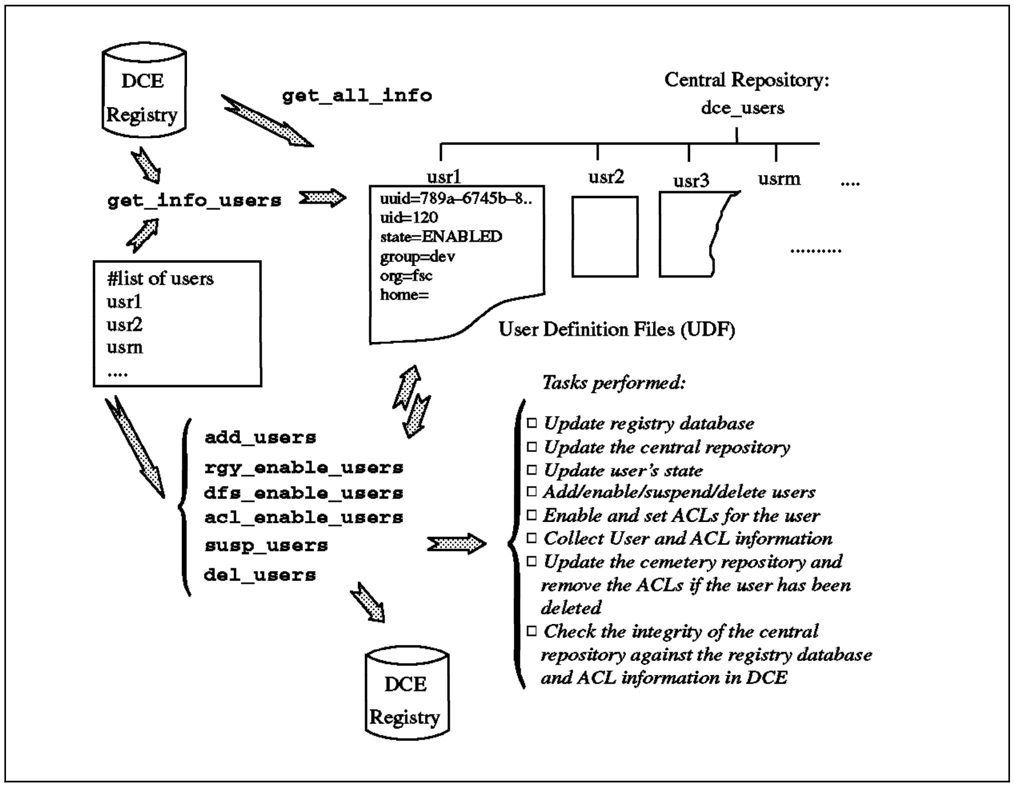

DCE Registry
get_all_info
Central Repository: dce_users
usr1
usr2
usr3
usrm
....
get_info_users
uuid=789a-6745b-8..
uid=120
state=ENABLED
group=dev
org=fsc
home=
....
#list of users
usr1
usr2
usrn
....
User Definition Files (UDF)
Tasks performed:
add_users
rgy_enable_users
dfs_enable_users
acl_enable_users
susp_users
del_users
Update registry database
Update the central repository
Update user's state
Add/enable/suspend/delete users
Enable and set ACLs for the user
Collect User and ACL information
Update the cemetery repository and remove the ACLs if the user has been deleted
Check the integrity of the central repository against the registry database and ACL information in DCE
DCE Registry

Figure 37. User Management Workflow. The commands work on a list of users and perform certain tasks for each of these users.

As you can see from Figure 37 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 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 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. 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 which are not enabled for login yet. 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 there yet, 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 reenabled.
`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 which 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, munich/sal and so on supporting the concept of branches as provided by Single Login/6000. These types of names will be stored in the central repository with file names 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. What we have used for principal and account can be adapted for group and organization as well.

5.5.2.1 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:

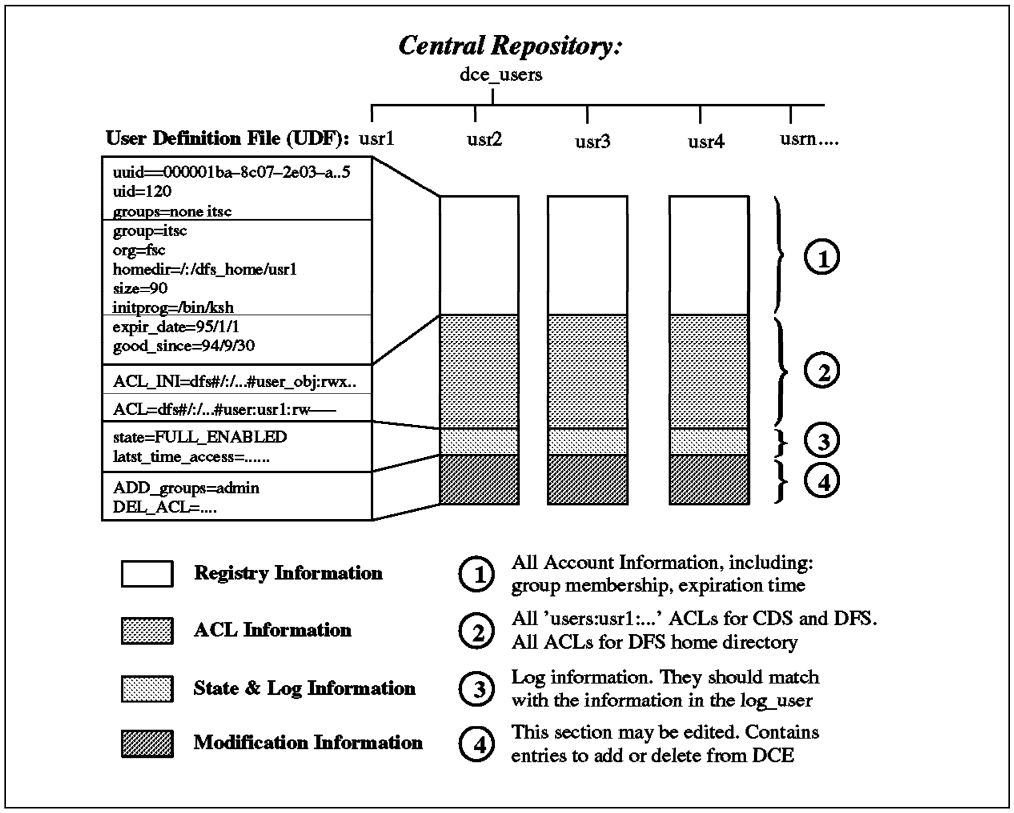

Central Repository:
dce_users
User Definition File (UDF):
usr1
usr2
usr3
usr4
usrn....
uuid==000001ba–8c07–2e03–a..5
uid=120
groups=none itsc
group=itsc
org=fsc
homedir=/:/dfs_home/usr1
size=90
initprog=/bin/ksh
expir_date=95/1/1
good_since=94/9/30
ACL_INI=dfs#/:/...#user_obj:rwx..
ACL=dfs#/:/...#user:usr1:rw——
state=FULL_ENABLED
latst_time_access=......
ADD_groups=admin
DEL_ACL=....
1
2
3
4
Registry Information
ACL Information
State & Log Information
Modification Information
1 All Account Information, including: group membership, expiration time
2 All 'users:usr1:...' ACLs for CDS and DFS. All ACLs for DFS home directory
3 Log information. They should match with the information in the log_user
4 This section may be edited. Contains entries to add or delete from DCE

Figure 38. The Central Repository. The central repository (dce_users directory) contains all the registry account and ACL information for a user.

Organized in this way the central repository contains ACL information related to a specific account or user. This is information which 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 object 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 which 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 todays DCE. Maintenance of a global ACL repository should be built into the `acl_edit` command, such 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 gives details about the attributes in that file.

5.5.2.2 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 login 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 which should have been applied in the last enabling process, but 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. Upon next reenabling steps all the values defined here are applied. When this is successful, the value is deleted from the modification section and the according value 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 diretory (or for which the user is the owner)
ADD_ACL_INI_OC	Define a new initial object creation ACL entry for the home diretory (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 which allows to remove 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 5.5.2.4, “General Structure and Customization” on page 252 for global configuration options.

5.5.2.3 Account State Information

The state in the central repository can provide cell administrators with information such as:

- How many users are enabled
- How many users are suspended
- Which users 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.

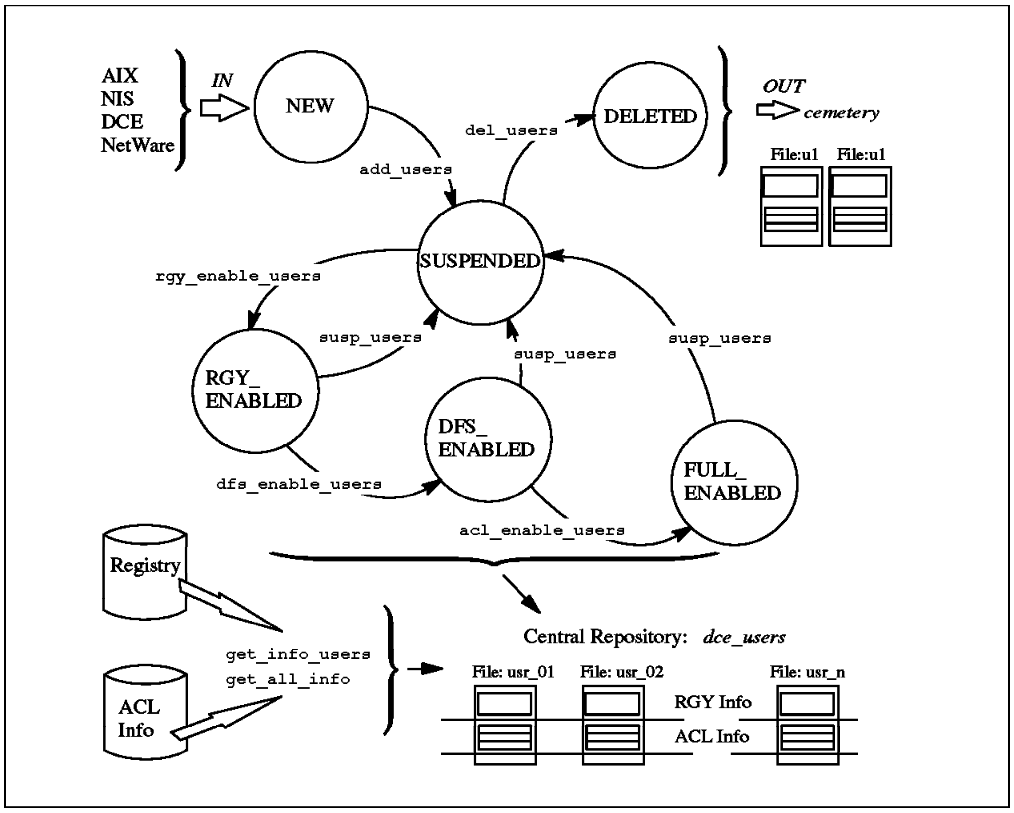


Figure 39. DCE User State Diagram. Each user defined with these set of tools is always in a certain state. The defined commands are used for state transition.

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 working in DCE. No ACL information is applied yet.
`DFS_ENABLED`	The account for the user is enabled in the DCE registry and their 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 them from logging into DCE, but it keeps all information about group membership, 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 but basically just changes the state.

5.5.2.4 General Structure and Customization

Our user management tools are contained in the tar-file umgt.tar. They expand in the current directory, whatever 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 which 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 an -h flag, which displays what the purpose of each command is. 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 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.

5.5.2.5 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. Call `. READ_UDF`, which assigns the default values to the new user file
2. Call `. WRITE_UDF` to write the upper part of the file with the default values
3. Interpret the values in the old environment and create ADD instructions which you append to the UDF

The `pwd2dce` procedure is provided on the diskette that comes with this publication. It is essentially the same procedure as `nis2dce_users` as listed in 4.7.2.3, "The nis2dce_users Procedure" on page 207.

When UDFs are created from an old environment, they should be carefully inspected before the users are added to DCE.

5.5.2.6 Integrity of the UDF

There two ways of maintaining integrity in the user definition files:

1. You do not care what might be inconsistent between the UDF and what is defined out there in the cell, that is to say the registry and the ACL lists of CDS and DFS objects.

   Then you just run `get_info_users`. This command gathers all information for the specified users and overwrites the UDF. So, this basically just refreshes the UDF.

2. You want to know where there are inconsistencies.

   Then you run `get_all_info` which extracts information for all users into a separate directory. You then can compare the UDFs in central repository with the files in the new directory with regular UNIX commands such as `diff`.

5.5.3 Group Management

Since it was so easy to adapt the user management concepts to group management, it was done for your convenience in a last minute effort. We decided anyone who intends to use the user management tools will also want to treat groups in the same way. So, we added this extension, but were not able to document it in the same detail. 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 is 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 they are 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.

5.5.3.1 Group State Overview

Please refer to 5.5.2.3, "Account State Information" on page 251 for the general idea of the states and the commands that are possible in each state.

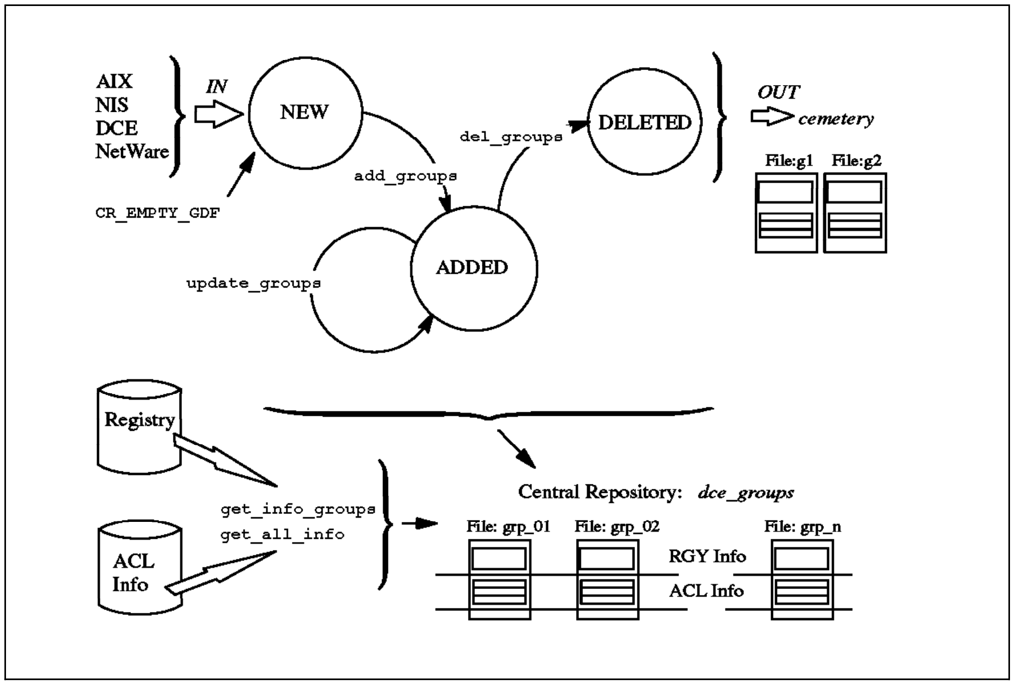


Figure 40. DCE Group State Diagram

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.

5.5.3.2 Group Commands Overview

Please refer to 5.5.2, "Management Tool Structure and Overview" on page 244 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_users	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.

5.5.3.3 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 5.5.2.1, “Central Repository” on page 246.

5.5.3.4 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 which should have been applied in the last process, but 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

5.5.3.5 General Structure and Customization

Our group management tools are contained in the tar-file umgt.tar. Together with the user management commands they expand in the current directory, whatever 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 which 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 an -h flag, which displays what the purpose of each command is. All scripts are extensively commented.

5.5.3.6 Migration from Other Environments

To create a tool which 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. Call `. READ_GDF`, which assigns the default values to the new group file
2. Call `. WRITE_GDF` to write the upper part of the file with the default values
3. Interpret the values in the old environment and create ADD instructions which you append 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.

5.5.4 Adding Users: add_users

The `add_users` procedure is a simple shell script which adds principals and accounts to DCE. After execution of this step the accounts are not enabled for login yet, 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

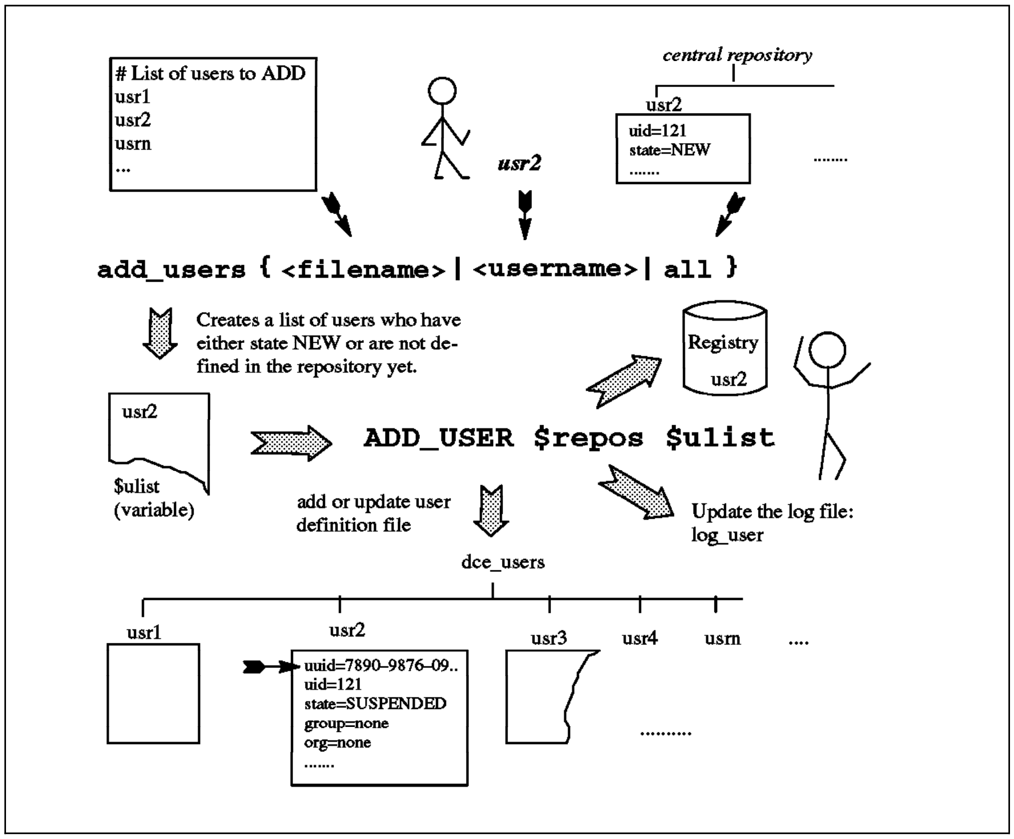


Figure 41. The add_users Procedure. After checking that the state of all candidates are NEW, the script ADD_USER is called, which creates entries in the DCE registry and in the central repository.

5.5.4.1 Syntax

The command takes one of the following forms:

```
add_users <username>
add_users <filename>
add_users all
add_users -h
```

5.5.4.2 Arguments

`username`	Single user name to be added, such as joe or austin%joe
`filename`	File containing only user names
`all`	Keyword which 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

5.5.4.3 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 which 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 5.5.2.1, "Central Repository" on page 246 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 whether 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, but not enabled for login in DCE. If defined, only the UID is applied. Group and organization are set to none. 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 could 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.

5.5.4.4 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:

```
cat << EOF | rgy_edit
domain principal
add name_principal uid
quit
EOF

cat << EOF | rgy_edit
domain account
add name_principal -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 they are in the SUSPENDED state.

5.5.4.5 How to Specify UIDs

The only way 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 5.5.2.1, "Central Repository" on page 246:

- ADD_uid=120
- ADD_newgrp=dev
- ADD_homedir=/:/dfs_home/usr2
- ADD_initprog=/bin/csh
- state=NEW
- .....etc.

The add_users command only honors the UID to create the account, because this is the only value of a user which cannot be changed later on. The rest of the attributes are used only by the different enable_users commands.

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 registry property lowest UID for principal creation to a value of your choice, for example to 2400:

```
# rgy_edit
Current site is: registry server at /.../itsc.austin.ibm.com/subsys....
rgy_edit> properties
    .....
    .....
Do you wish to make changes [y/n]? (n)  y
    .....
    .....
  Lower bound unix id for automatic UID assignment: (100)  2400
```

```
    .....
rgy_edit> quit
#
```

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.

5.5.4.6 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.
```

5.5.4.7 Initial Password

The initial password for each user will be set with an environment variable which 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 user to change the password the first time they log in. DCE the will simply display the message:

```
Password must be changed!
```

It will be good practice for the cell administrator to check with a simple script if the user has changed the password or still 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 matter of fact DCE warns you, if you give 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)
```

providing hackers with valuable information to guess users and password in DCE and worst the hacker can try many times because the number of attempts is unlimited !!!!

Single Login/6000 does not have these problems. It actually forces the users to change their password. It is possible to enforce a password policy and password rules with this program offering. It also keeps track of failed logins and excludes the user from further login after a configurable number of failed attempts.

5.5.5 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 41 on page 259. The difference is that the state must be SUSPENDED and that RGY_ENABLE_USER $repos $ulist is called.

5.5.5.1 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
```

5.5.5.2 Arguments

username	Single user name to be added, such as joe or austin%joe
filename	File containing only user names
all	Keyword which indicates that the central repository should be searched for all users in state SUSPENDED.
-h	Displays more information on the purpose of the program

5.5.5.3 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 which 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 whether it has a user definition file and whether their 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 5.5.2.4, "General Structure and Customization" on page 252. Then the DCE command rgy_edit is called to try 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 yet
- Primary group does not exist yet
- Primary organization does not exist yet

Other minor errors may be discovered, for instance 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.

5.5.5.4 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:

```
cat << EOF | rgy_edit
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
```

5.5.5.5 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 which 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 g7 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:

```
# -- !!!!!!!!!!!!!  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.

5.5.6 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.

If an administrator decides, they want to first define all ACLs manually, they need no ACL_INI definition in the UDFs. The ACL_INI definitions can also be generated later from what is actually defined on the DFS directories by running get_info_users. If they want to set user ACLs later on with acl_enable_users, they must run acl_enable_users to set the state to DFS_ENABLED, even if no ACL_INIs are defined.

5.5.6.1 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
```

5.5.6.2 Arguments

`username`	Single user name to be added, such as joe or austin%joe
`filename`	File containing only user names
`all`	Keyword which 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

5.5.6.3 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 candidiates 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 which 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.

5.5.6.4 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
```

5.5.7 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 which 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.

5.5.7.1 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
```

5.5.7.2 Arguments

username	Single user name to be added, such as joe or austin%joe
filename	File containing only user names
all	Keyword which 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

5.5.7.3 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 which 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.

5.5.7.4 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
```

5.5.8 Suspending Users: susp_users

The susp_users procedure invalidates the DCE account in the DCE registry (NOT valid), so that users cannot login anymore. Then it sets the state of the UDF to SUSPENDED. This is a safe state to either delete the user or change some attributes and reenable them again.

5.5.8.1 Syntax

The command takes one of the following forms:

```
susp_users <username>
susp_users <filename>
susp_users all
susp_users -h
```

5.5.8.2 Arguments

username	Single user name to be added, such as joe or austin%joe
filename	File containing only user names
all	Keyword which indicates that the central repository should be searched for all users in state *_ENABLED
-h	Displays more information on the purpose of the program

5.5.8.3 Description

The whole description is in the introductory remarks.

5.5.9 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 where it can be used to define the user with the same characteristics in another cell.

5.5.9.1 Syntax

The command takes one of the following forms:

```
del_users <username>
del_users <filename>
del_users all
del_users -h
```

5.5.9.2 Arguments

`username`	Single user name to be added, such as joe or austin%joe
`filename`	File containing only user names
`all`	Keyword which indicates that the central repository should be searched for all users in state SUSPENDED
`-h`	Displays more information on the purpose of the program

5.5.9.3 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 pretty 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 as defined in the UDF in 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, which 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.

5.5.9.4 Implementation Specifics

The ACL removal part is implemented in the same way as in the `acl_enable_users` command.

5.5.10 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 5.5.9, “Deleting Users: del_users” on page 269 for a description of this issue.

5.5.10.1 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
```

5.5.10.2 Arguments

`username`	Single user name to be added, such as joe or austin%joe
`filename`	File containing only user names
`all`	Keyword which 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

5.5.10.3 Description

The whole functional description is in the introductory remarks.

If a username specified as <username> or contained in the file <filename> does not have a UDF yet, 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 /:/*, which scans through the ACLs of *all* DFS files, leaving out explicit read-write mount points like for instance /:/.rw or /:/.usrbin.

5.5.10.4 Implementation Specifics

This routine is able 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.

5.5.11 Getting Information for All Users from DCE: get_all_info

The `get_all_info` procedure does basically the same as `get_info_users all` does.

The only difference is, it will query all usernames and groupnames from the DCE registry and collect information for all of them.

5.5.11.1 Syntax

The command takes one of the following forms:

```
get_all_info <userdir> <groupdir>
get_all_info -h
```

5.5.11.2 Arguments

`userdir`	Repository (directory) for user files
`groupdir`	Repository (directory) for group files
`-h`	Displays more information on the purpose of the program

5.5.11.3 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 which do not have a file yet, 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 /:/*, which scans through the ACLs of *all* DFS files, leaving out explicit read-write mount points like for instance /:/.rw or /:/.usrbin.

5.6 DCE on IBM AIX High Availability Cluster Multi-Processing/6000

IBM AIX High Availability Cluster Multi-Processing/6000 is an environment of loosely coupled, clustered RS/6000* machines executing the HACMP/6000 software on top of AIX to provide for high availability through redundancy and shared resource access.

DCE on the other hand can provide a large scale distributed client/server environment with application server processes running on many different machines. To coordinate load and ensure security the DCE core services need to be contacted before getting access to any application server. It is obvious that these core services must be available all the time to keep the whole DCE cell alive.

When using HACMP/6000 for the core services the clients in the DCE network are not aware of the system change and behave as they would when restarting DCE master services. Positive aspects of this are:

- High reliability
- Easy to maintain and control
- Independence of DCE

Negative aspects are:

- Requires additional planning and installation efforts
- Proprietary solution (AIX and RS/6000 based only)
- Expensive because of additional hardware requirements

Also customer developed DCE applications certainly need to be highly available.

This section gives a short overview on:

- HACMP/6000 support for DCE
- How DCE core services can use HACMP/6000
- How DCE applications can use HACMP/6000

5.6.1 HACMP/6000 Support for DCE

As mentioned in 1.3.5, "IBM HACMP/6000" on page 25 DCE is supported only on nonconcurrent access (mode 1) configurations of HACMP/6000. This means:

1. One-for-one standby configuration with owned/takeover resources

   This is a redundant hardware configuration where one or more takeover nodes (standby) stand idle, waiting for an owner node (server) to detach from the cluster. This is also known as *hot standby mode*. When the owner node rejoins the cluster, the takeover node releases the resource to its owner.

2. One-sided takeover using owned/takeover resources

   Also known as *simple fallover* is similar to *hot standby*, but the standby node is doing some work which it can give up when a takeover occurs.

3. One-for-one standby configuration with rotating resources

   This configuration differs from the above in that the ownership of the resources is not fixed. In this case, resources are not designated as *owned* by one node and *takeover* to the other. Rather they are tied to a shared IP address and defined as rotating resources to both nodes. This is also known as *rotating standby*.

In order to understand IBM AIX High Availability Cluster Multi-Processing/6000 and get more details on how it works, the following publications are recommended:

- *HACMP/6000 System Overview*
- *HACMP/6000 Planning and Installation Guide*
- *HACMP/6000 Administration Guide*
- *HACMP/6000 Troubleshooting Guide*
- *HACMP/6000 Application Programming Interface Guide*

5.6.2 DCE Core Services on HACMP/6000

As mentioned above availability of DCE core services such as CDS, security service, and DTS are vital to a DCE cell. DCE itself implements redundancy by replicating its core servers and their associated databases. As with other distributed or replicated databases they allow only read access. Write access is only possible to the master database.

This answers the question as to where HACMP/6000 makes sense in DCE cells. Usually a cell is operational with read access to CDS and security. Tickets can be issued and binding handles can be looked up, so users still can login and services can be found and executed. However, the following are some examples where write access is always required:

- DCE application server frequently starting and stopping

DCE application servers which behave as recommended export their interfaces to CDS when they start and remove them when they stop. This means they need write access.

- Applications might (ab)use CDS as a central data repository
- Customer wants to be always able to change cell configurations
- Customer wants to be always able to modify the registry (user accounts)

Since HACMP/6000 ensures the availability of resources during system hardware or network failures and is independent of DCE, it is the ideal platform for the DCE security and CDS core services.

In case of a failure (disk, network or system hardware) a takeover to a standby HACMP cluster system takes place. This takeover may cause an unavailability of core services for a few minutes. This unavailability can be bridged with the DCE read-only replication of the master services. As soon as the standby DCE cluster system restarts the DCE master core services, the clients have there read/write services accessible again.

DCE on HACMP/6000 is supported with AIX DCE Version 1.2 and Version 1.3 for the following services:

- rpcd
- secd
- sec_cl
- cds_srv
- cds_cl
- cdsadv

Additionally we installed and tested dts_local which worked fine. At the time we wrote this publication, no official support for DFS was announced.

DFS Support for HACMP/6000

In the meantime DFS support has now been announced. All components of DFS can be installed in the above mentioned HACMP/6000 mode 1 configurations. More information can be found in the InfoExplorer documentation for AIX DCE 1.3.

5.6.3 DCE Application Servers on HACMP/6000

Not only DCE core services can be made highly available with HACMP/6000, any other DCE application can be implemented to HACMP/6000. It is probably the easiest way to make an application highly available which was not designed this way. However, if there are requirements for highly available DCE servers at many different locations, it may become too expensive to ensure high availability through HACMP/6000. It may also be possible that a DCE application server needs to run on another platform than AIX. Such situations require the usage of DCE dependent tools for replication.

In order to run replicated or with redundant instances DCE client/server applications should be well written DCE programs which are capable to register and unregister in the CDS namespace properly. The *OSF DCE Application Development Reference (Prentice Hall)* from OSF is very helpful to understand how to write proper DCE applications. As mentioned in 5.6.2, “DCE Core Services on HACMP/6000” on page 273 you might want to consider HACMP/6000 to hold the CDS master directories. This would enable the DCE applications to register their interfaces at all times.

In order for application servers to be able to register/unregister their interfaces with CDS, they need write access to CDS. If there is a lot of application starting and stopping going on, CDS must be highly available for write access. If you wanted to avoid the need for highly available write access, you could decide to manually add binding information to CDS and to write the servers programs so they do not register/unregister themselves. However, using static CDS entries introduces the risk that some stale interface definitions are hanging around in CDS for services which are not available for any reason. This would result in DCE timeouts when DCE clients try to use them.

Appendix A. Installing the Tools

Together 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 backup_CH, 8039 bytes, 16 media blocks.
x cleanif, 9692 bytes, 19 media blocks.
x cleanup_cache, 2174 bytes, 5 media blocks.
x cleanup_cds_cache, 1178 bytes, 3 media blocks.
x cleanup_ip, 2448 bytes, 5 media blocks.
x copy_CH, 5270 bytes, 11 media blocks.
x create_cds_entry, 662 bytes, 2 media blocks.
x rmsi_stream_mapping, 3045 bytes, 6 media blocks.
x umgt.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 umgt.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 5.5, “User (and ACL) Management” on page 242.

We suggest creating a directory /umgt 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, 5230 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, 3088 bytes, 7 media blocks.
x GET_ACCOUNT, 5709 bytes, 12 media blocks.
x GET_ACL, 10390 bytes, 21 media blocks.
x GET_ACL_INI, 5959 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, 15 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, 9763 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. Description of the Systems in our Scenario

This appendix is a overview of our test environment. It shows the IP environment with connections, IP addresses, and system names. Then it lists all machines with their hardware configuration.

B.1 Our Test Network Environment

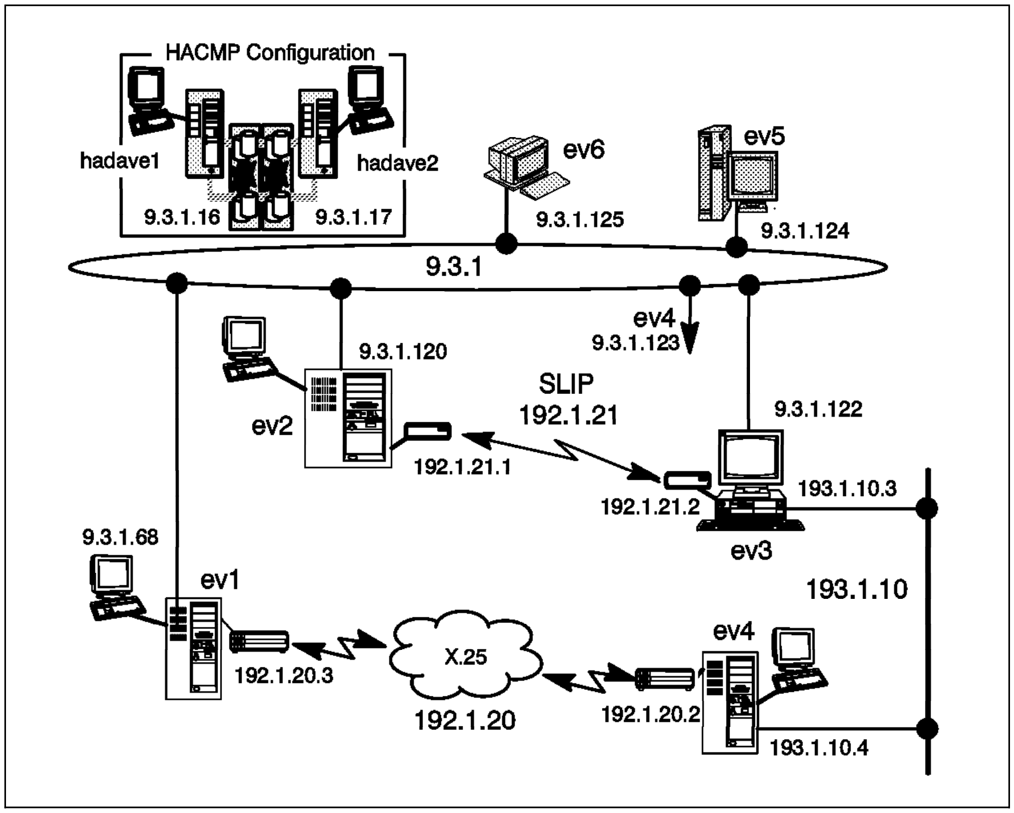


Figure 42. IP Network of our Test Environment

B.2 Hardware Configurations

The following will show you the system configurations of our test environment. The systems are named ev1 through ev6 where ev1 to ev4 are RS/6000, ev5 and ev6 are PS/2 systems. The systems of the High Availability Cluster are named hadave1 and hadave2.

Disclaimer: Although we have used the following systems within our test scenarios, they may not fit the needs of your customer expectations. The performance numbers you will see later on in this chapter are also not mentioned to show DCE performance but can help to design DCE cell structure in order where and why to place certain DCE services. This becomes very important especially while planing the implementation of large DCE environments.

B.2.1.1 System Information of ev1

System configuration:

- RS/6000 Model 520
- mem0 Available 00-0D 32 MB Memory Card
- mem1 Available 00-0H 32 MB Memory Card
- hdisk0 400 MB SCSI Disk
- hdisk1 400 MB SCSI Disk
- AIX Version 3 Release 2.5

Network configuration:

- tr0 Available Token Ring Network Interface

  ```
  flags=8063<UP,BROADCAST,NOTRAILERS,RUNNING,ALLCAST> inet 9.3.1.68
  netmask 0xffffff00 broadcast 9.3.1.255
  ```

- xt0 Available X.25 Network Interface

  ```
  flags=61<UP,NOTRAILERS,RUNNING> inet 192.1.20.3 netmask 0xffffff00
  ```

B.2.1.2 System Information of ev2

System configuration:

- RS/6000 Model 720
- mem0 Available 00-0B 16 MB Memory Card
- mem1 Available 00-0D 32 MB Memory Card
- mem2 Available 00-0F 16 MB Memory Card
- mem3 Available 00-0H 32 MB Memory Card
- hdisk0 670 MB SCSI Disk
- hdisk1 670 MB SCSI Disk
- AIX Version 3 Release 2.5

Network configuration:

- tr0 Available Token Ring Network Interface

  ```
  flags=8063<UP,BROADCAST,NOTRAILERS,RUNNING,ALLCAST> inet 9.3.1.120
  netmask 0xffffff00 broadcast 9.3.1.255
  ```

- sl1 Available Serial Line Network Interface

  ```
  flags=31<UP,POINTOPOINT,NOTRAILERS> inet 192.1.21.1 --> 192.1.21.2
  netmask 0xffffff00
  ```

B.2.1.3 System Information of ev3

System configuration:

- RS/6000 Model 320
- mem0 Available 00-0B 16 MB Memory Card
- mem1 Available 00-0C 32 MB Memory Card
- hdisk0 320 MB SCSI Disk
- hdisk1 400 MB SCSI Disk
- Operating System: AIX Version 3 Release 2.5

Network configuration:

- tr0 Available Token Ring Network Interface

  ```
  flags=8063<UP,BROADCAST,NOTRAILERS,RUNNING,ALLCAST> inet 9.3.1.122
  netmask 0xffffff00 broadcast 9.3.1.255
  ```

- en0 Available Standard Ethernet Network Interface

  ```
  flags=2000063<UP,BROADCAST,NOTRAILERS,RUNNING,NOECHO> inet 193.1.10.3
  netmask 0xffffff00 broadcast 193.1.10.255
  ```

- sl0 Available Serial Line Network Interface

  ```
  flags=71<UP,POINTOPOINT,NOTRAILERS,RUNNING> inet
  192.1.21.2 --> 192.1.21.1 netmask 0xffffff00
  ```

B.2.1.4 System Information of ev4

System configuration:

- RS/6000 Model 520
- mem0 Available 00-0H 64 MB Memory Card
- hdisk0 400 MB SCSI Disk
- hdisk1 400 MB SCSI Disk
- AIX Version 3 Release 2.5

Network information:

- tr0 Available Token Ring Network Interface

  ```
  flags=8063<UP,BROADCAST,NOTRAILERS,RUNNING,ALLCAST> inet 9.3.1.123
  netmask 0xffffff00 broadcast 9.3.1.255
  ```

- en0 Available Standard Ethernet Network Interface

  ```
  flags=2000063<UP,BROADCAST,NOTRAILERS,RUNNING,NOECHO> inet 193.1.10.4
  netmask 0xffffff00 broadcast 193.1.10.255
  ```

- xt0 Available X.25 Network Interface

  ```
  flags=61<UP,NOTRAILERS,RUNNING>  inet 192.1.20.2 netmask 0xffffff00
  ```

B.2.1.5 System Information of ev5

System configuration:

- PS/2 Model 8595-OKF
- OS/2 2.1 with TCPIP for OS/2 V2.0
- 24 MB memory available
- Math Coprocessor installed
- Disk0 400MB SCSI Disk

Network configuration

- lan0: IBM Token-Ring Network 16/4 Adapter/A (Pimary, 16MBps)

  ```
  flags=3063<UP,BROADCAST,NOTRAILERS,RUNNING,BRIDGE,SNAP>
  metric 1 inet 9.3.1.124 netmask ffffff00 broadcast 9.3.1.255
  ```

B.2.1.6 System Information of ev6

System configuration

- PS/2 Model 8595-OKF
- DOS 6.1 with Windows 3.1 and TCPIP for DOS V2.1.1
- 24 MB memory available
- Math Coprocessor installed
- Disk0 320MB SCSI Disk

Network configuration

- nd0: IBM Token-Ring Network 16/4 Adapter/A (Pimary, 16MBps)

  ```
  flags=1063<UP,BROADCAST,NOTRAILERS,RUNNING,ALLROUTES> inet
  9.3.1.125 netmask ffffff00 broadcast 9.3.1.255
  ```

B.2.1.7 System Information of hadave1

System configuration:

- RS/6000 Model 520
- mem0 Available 00-0H 32 MB Memory Card
- hdisk0 857 MB SCSI Disk
- hdisk1 857 MB SCSI Disk
- hdisk2 857 MB SCSI Disk
- hdisk3 857 MB SCSI Disk
- hdisk4 400 MB SCSI Disk
- hdisk5 670 MB SCSI Disk
- hdisk6 1.0 GB Differential SCSI Disk
- hdisk7 1.0 GB Differencial SCSI Disk
- scsi0 Available
- scsi1 Available
- scsi2 Available
- Operating System: AIX Version 3 Release 2.5

Network configuration:

- tr0 Available Token Ring Network Interface

  ```
  flags=8063<UP,BROADCAST,NOTRAILERS,RUNNING,ALLCAST> inet 9.3.1.16
  netmask 0xffffff00 broadcast 9.3.1.255
  ```

- tr1 Available Token Ring Network Interface

  ```
  flags=18063<UP,BROADCAST,NOTRAILERS,RUNNING,ALLCAST,HWLOOP> inet 9.3.4.16
  netmask 0xffffff00 broadcast 9.3.4.255
  ```

B.2.1.8 System Information of hadave2

System configuration:

- RS/6000 Model 520
- mem0 Available 00-0H 32 MB Memory Card
- hdisk0 857 MB SCSI Disk
- hdisk1 857 MB SCSI Disk
- hdisk2 670 MB SCSI Disk
- hdisk3 857 MB SCSI Disk
- hdisk4 857 MB SCSI Disk
- hdisk5 1.0 GB Differential SCSI Disk
- hdisk6 1.0 GB Differential SCSI Disk
- scsi0 Available
- scsi1 Available
- scsi2 Available
- scsi3 Available
- Operating System: AIX Version 3 Release 2.5

Network configuration:

- tr0 Available Token Ring Network Interface

  ```
  flags=8063<UP,BROADCAST,NOTRAILERS,RUNNING,ALLCAST> inet 9.3.1.17
  netmask 0xffffff00 broadcast 9.3.1.255
  ```

- tr1 Available Token Ring Network Interface

  ```
  flags=18063<UP,BROADCAST,NOTRAILERS,RUNNING,ALLCAST,HWLOOP> inet 9.3.4.17
  netmask 0xffffff00 broadcast 9.3.4.255
  ```

List of Abbreviations

ACL	access control list
ADSM	ADSTAR Data Storage Management
ATM	asynchronous transfer mode
CDMF	Common Data Masking Facility
CDS	Cell Directory Service
CICS	Customer Information Control System
CMIP	common management interface protocol
CMVC	Configuration Management and Version Control
COSE	Common Open Software Environment
DCE	Distributed Computing Environment
DAP	directory access protocol
DES	data encryption standard
DFS	Distributed File System
DME	Distributed Management Environment
DNS	domain name service
FCS	fibre channel standard
FLDB	Fileset Location Database
GDA	Global Directory Agent
GDF	group definition file
GDS	Global Directory Service
HACMP	High Availability Cluster Multi-Processing
IBM	International Business Machines Corporation
ICC	Initial Container Creation (ACL)
IDL	interface definition language
IHMP	IBM NetView Hub Management Program
IOC	Initial Object Creation (ACL)
ITSO	International Technical Support Organization
LAN	local area network
LFS	Local File System
MAN	metropolitan area network
NFS	Network File System
NIS	Network Information System
ONC	Open Network Computing
OSF	Open Software Foundation
PAC	privilege attribute certificate
PTX	Performance Toolbox
RDBMS	relational database management system
RPC	remote procedure call
SCM	System Control Machine
SNMP	simple network management protocol
SQL	structured query language
UDF	user definition file
UUID	universal unique identifier
WAN	wide area network
XMP	X/Open management protocol

Index

ITSO Technical Bulletin Evaluation RED000

International Technical Support Organization
Using and Administering AIX DCE 1.3
November 1994

Publication No. GG24-4348-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

What other topics would you like to see in this Bulletin?

What other Technical Bulletins would you like to see published?

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

Name

Address

Company or Organization

Phone No.

ITSO Technical Bulletin Evaluation **RED000**
GG24-4348-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department 948, Building 821
Internal Zip 2834
11400 BURNET ROAD
AUSTIN TX
USA 78758-3493

Fold and Tape **Please do not staple** Fold and Tape

GG24-4348-00

Cut or Fold
Along Line

Printed in U.S.A.

GG24-4348-00