SG24-4949-00

Security on the Web Using DCE Technology

June 1997

International Technical Support Organization
Austin Center

International Technical Support Organization SG24-4949-00

Security on the Web Using DCE Technology

June 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, "Special Notices" on page 185.

First Edition (June 1997)

This edition applies to IBM DFS Web Secure, Version 1.1, DE-Light Web Client and Gateway, Version 2.0, from Transarc Corp., IBM Internet Connection Secured Netwok Gateway for AIX, Version 2.2, and the DCE/DFS and DSS family of products for use with AIX and OS/2 Warp.

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

Contents

Figures

Tables

Preface

The Internet and corporate, internal intranets running browser-based applications have become one of the fastest growing segments in modern information processing technology. They allow fast access to relevant data and applications and therefore open new ways for organizations to support their business. While the base technology for such Web-based, network-centric applications is relatively simple and cheap, managing controlled access and protecting resources becomes a more complex issue. The Distributed Computing Environment (DCE), along with the Distributed File System (DFS), are established industry standards that provide excellent levels of security features and a base for platform independent, distributed applications.

This redbook describes various ways to implement and exploit the superb security features of DCE and DCE-based applications running on intranets and on the Internet.

The book is written for anyone who needs to understand how DCE and DFS can work together to enhance the capabilities of Web-based applications. The main focus is on security.

Several practical examples are presented to demonstrate the implementation of the concepts and the use of the products in real life. Sample program code is provided for an easy start.

How This Book Is Organized

This redbook is organized as follows:

- Chapter 1, “Security Overview for Distributed Environments” on page 1

  This chapter provides an introduction to security concepts and currently available solutions for specific use on the Internet and in Web-based applications.

- Chapter 2, “The Business Requirement” on page 21

  As corporate internal intranets and the Internet come together, new business requirements arise which require adequate solutions for the security exposures. This chapter explains some typical scenarios where DCE can provide the solution.

- Chapter 3, “Strengthen Server Security: IBM DFS Web Secure” on page 29

  This chapter provides an overview of Web server technology and how the IBM DFS Web Secure product can extend its functionality by adding DCE security features.

- Chapter 4, “DFS Web Secure Installation and Customizing” on page 43

  This chapter explains, in a step-by-step approach, the installation and configuration of the IBM DFS Web Secure product with its prerequisites, the Web server and DFS.

- Chapter 5, “Running the Business With DFS Web Secure” on page 63

  After the products are installed and running, new questions may arise, such as how do I manage the users and how can certain events be audited? This chapter contains useful guidelines on how to run these products in a real world environment.

- Chapter 6, “DE-Light Web Client” on page 109

  This provides an in-depth view of the DE-Light Web Client product from Transarc Corp. This chapter explains how this product allows you to run DCE applications from a Web browser.

- Chapter 7, “Alternate Methods for Using DCE on the Web” on page 133

  This chapter provides additional ways to exploit DCE and DFS security schemes in an intranet or Internet environment.

- Chapter 8, “Bridging Networks Together: DCE Over Firewalls” on page 143

  Firewalls are not only key between an intranet and the Internet, they may also exist within an organization. As they limit the access to resources on the other side, the question of whether DCE will still run arises. This chapter explains how to set up a firewall in order to allow DCE services to operate across it.

- Appendix A, “Other Sources on the Internet” on page 155

  This appendix lists additional, valuable sources of information pertinent to the implementation and use of DCE and the Web environment.

- Appendix B, “Program Listings” on page 159

  This appendix contains the listings of the example programs used in this book.

The Team Who Wrote This Book

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Austin Center.

Heinz Johner is an Advisory Systems Engineer at the International Technical Support Organization, Austin Center. Before joining the ITSO in mid-1996, he worked for IBM Switzerland, and he had the overall responsibility for AIX, DCE, and systems management projects for large customers. He was also involved as a consultant in various other customer projects in the same technical areas.

Dr. Andy Bond is a Principal Research Scientist at the Co-operative Research Centre for Distributed Systems Technology in Brisbane, Australia. He has over 10 years experience in the field of distributed systems, including load sharing, distributed system architecture design, DCE, CORBA, and experimental middleware technologies. He has written many publications in these areas. He holds a Ph.D. degree from Victoria University of Wellington, New Zealand where his topic was ″Adaptive Load Sharing in a Distributed Workstation Environment″.

Ian Wheway is a RISC System/6000 Systems Specialist in the United Kingdom. He has nine years of experience as an AIX specialist. He has worked at IBM for 11 years. His areas of expertise, in which he has been involved as a consultant, include DCE and Object Oriented technology.

Thanks to the following people for their invaluable contributions to this project:

Shepherd Shi
IBM Austin

Ted Shrader
IBM Austin

Bruce Rich
IBM Austin

Bjarne Rasmussen
IBM Denmark, Nordic Open Systems Center

Bobert Macgregor
IBM Raleigh, ITSO

Dino Chiesa
Transarc Corporation

Pat Stephenson
Transarc Corporation

Comments Welcome

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

redbook@vnet.ibm.com

Your comments are important to us!

Chapter 1. Security Overview for Distributed Environments

A distributed environment emphasizes the efficient use and availability of resources across a multi-platform computing environment. In recent years, the client/server model has operated within business intranets to provide information access to desktop clients connected to distributed servers. This environment is isolated within an organization and can rely on physical security to provide a firm foundation for an electronic security architecture.

The recent explosion of the World Wide Web and subsequent impact on the need for access to the Internet has intensified the potential security dangers in a distributed environment. Organizations now require that a gateway be available to pass from the intranet through to the Internet. Organizations may require three modes of information access.

- New customers and contacts can be made by providing information on the World Wide Web. Powerful search engines can provide an access point to your information in this *public* forum as quickly as you can make it available. This requires minimal security constraints on the machine providing the information but maximal security for access to more private data.
- *Business partners* often benefit from accessing information beyond that made available to the general public. For example, you might want companies that repair the equipment you manufacture to have access to the same product specifications that the design engineers deal with. In this case you have to provide medium-level security before allowing access to this information.
- Finally, we have *internal* information that is private and confidential to an organization. This information must not be available to the general public nor to business partners. It might even be necessary to restrict access to some data within the organization itself.

There is a trade-off between taking advantage of potential new business opportunities and the protection of company data. An effective security policy and architecture must verify the identity of users wishing to access data, check their authorization to get to the data, and possibly provide some form of encryption to ensure that the data is delivered in a private and confidential way to the intended and verified recipient.

The World Wide Web servers and browsers are continually being enhanced to support multiple modes of information access. Secure protocols such as Secure HTTP (S-HTTP) and Secure Sockets Layer (SSL) provide private information transport between authenticated browsers and servers. Each party can be assured that they are talking to the correct partner and that others cannot meaningfully intercept the information that is passed. Such schemes lay the foundations for the long-promised electronic commerce systems. Systems such as *DigiCash* and *Open Market* are pushing new technologies for providing secure payment methods over the Web.

A common security infrastructure that exists today is the Distributed Computing Environment (DCE). It provides the essential features of a security infrastructure, including authentication, authorization, and privacy.

- IBM DFS Web Secure extends Web server technology to utilize the full security infrastructure available to documents stored within DCE's Distributed File System (DFS).

- Transarc's DE-Light lets the Web gain access to new and legacy DCE servers via a lightweight DCE gateway already working successfully in the Internet and intranet environments.
- Distributed Systems Series Link (DSSL) for Lotus Domino is another integration tool for DCE that brings together the desktop information environment and DCE servers.
- Other concepts, or even products, may exist that exploit DCE security schemes for Web-based applications.

Each of these tools builds stronger links between the intranet and Internet while ensuring security is maintained. In this chapter we provide an overview of each technology and describe how DCE security can contribute to an effective security architecture in an Internet/intranet environment. Throughout the other chapters of this book, we go into more detail and explain how these products can be used in an actual Web environment.

1.1 Security Requirements in Network-Centric Computing

Organizations have embraced networking to bring together previously disconnected elements into a more cohesive whole. Through such an infrastructure, the *Client/Server Computing Model* has promoted the sharing and availability of resources. This sharing is a primary characteristic of a *distributed computing* environment. These networks are typically internal to an organization and rely on this fact when defining their security requirements.

In contrast the *Network-Centric Computing Model* opens the organization up to access by external sources. This may include arbitrary strangers browsing the network or identified external collaborators with designated access to previously internal information.

The two models of network-based computing place different demands on security policy and infrastructure. The common security requirements for any network computing environment include:

- Verifying the identity of those wishing to gain access to information and services, also called *authentication*
- Managing access to information and services based on a verified identity, also called *authorization*
- Securing information in an encrypted form to prevent unauthorized access as it is moved around the network, also called *privacy*

(See 1.2, "Security Concepts" on page 5, for a more thorough description of these terms.)

Security in an open network is often restrictive by default, but the internal network applies the reverse policy.

1.1.1 The Classic Client/Server Model

A client/server environment can be thought of as an extended family. By default information is available, and there is an element of trust among the participants. Clients and servers within the environment usually have very specialized, but shared, protocols and know their peers within the environment. For example, a database is programmed to expect a certain number of transactions each day and from which terminals these transactions will be provided. Applications use their own authentication and authorization methods, or they use a sophisticated foundation for this purpose, such as DCE.

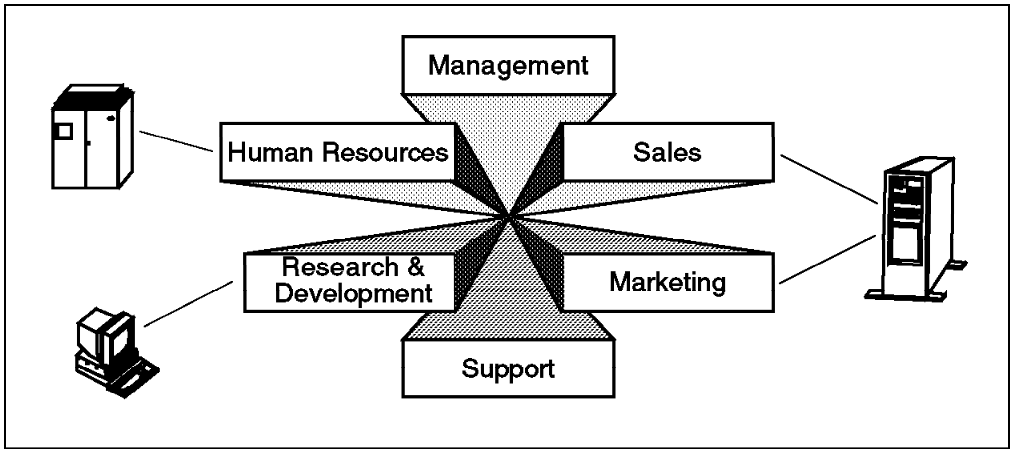


Figure 1. An Intranet Combines Internet Technologies and Organizational Information

A subset of the client/server environments that is becoming more widespread within today's organizations is the *intranet*. The term refers to a more general view of an organization's internal, network-based services that allows for a close collaboration between the organization's divisions, departments, and eventually down to the single employee (see Figure 1).

An intranet is characterized by:

- The use of standard Internet technologies and tools such as the World Wide Web, electronic mail, and other TCP/IP-based solutions
- Primary support for internal corporate use
- Integration with existing desktop and server infrastructures

Intranets take advantage of security facilities provided in the standard Internet environment as well as those existing within the desktop community. The combination of diverse security mechanisms is often referred to as the *single sign-on* problem. Solutions aim to integrate security under a common umbrella requiring only a single login by a user instead of one login for each application.

1.1.2 The Network-Centric Computing Model

The Internet has given organizations a new opportunity to exploit information delivery to customers and collaborators. By providing a gateway onto the Internet, effective marketing and collaboration plans can be realized through managed access to what would otherwise exist only on the internal intranet.

The Internet is well known for a set of protocols and services that interconnect a diverse range of organizations, including:

- Universities and research institutions
- Computer hardware and software vendors
- Retailers
- Homes

Protocols exist to support the World Wide Web, electronic mail, news distribution, remote file access, and many others. It is these protocols and their associated services that have been borrowed by the intranet community for information distribution within the organization. Important differences still exist between intranets and the Internet, however.

- The Internet covers a wide range of cultures and countries each of which can impose restrictions on acceptable activity within their own community. The Internet knows little about physical or political boundaries. The resources that form the Internet are most often owned and controlled by ″somebody else″.
- An intranet generally involves physical control of the network environment. Network snooping and machine access can be controlled or even restricted. The participants—the information providers, as well as the consumers—generally share a common, business-related interest.

It must be assumed that there will be malevolent attacks against an organization through the Internet if an access point is provided. In order to prevent the success of such attacks, an organization must adopt a set of security policies and mechanisms that adequately protect the organization from such attacks.

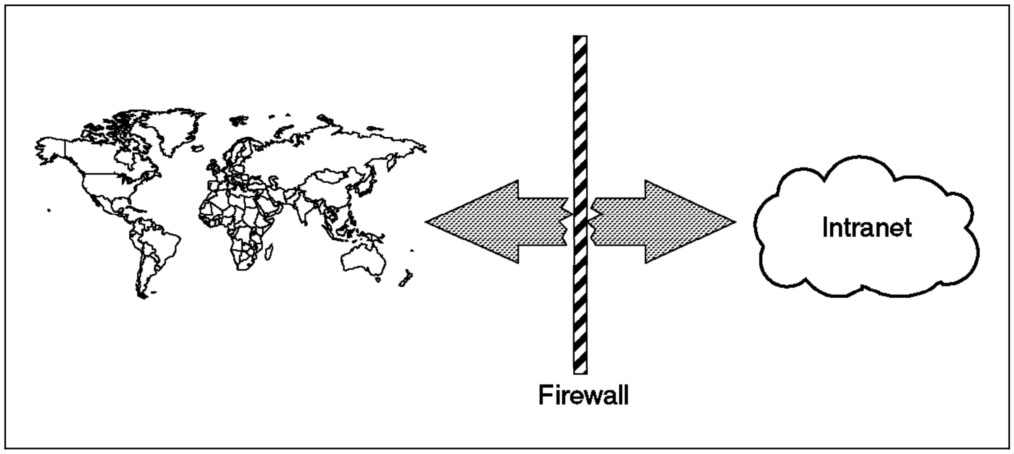


Figure 2. A Firewall Protects the Intranet from the World

One common method used to protect access to the intranet is a *firewall*. This is a computing device designed to filter unwanted traffic based on source and protocol. In order to access resources within the intranet, special permission must be given by the administrators of the firewall to allow access from a particular domain or company.

Sometimes it is useful for an organization to provide a machine outside the firewall if it is to be accessible by anyone on the Internet. This often occurs when marketing information should be available to potential new customers. In this case the firewall administrator treats the machine outside the firewall as likely to sustain attack from unknown parties on the Internet, and thus little trust is bestowed on the machine. It is treated as another external computer which may or may not be allowed access through the firewall.

When an intranet is to be opened to the Internet, security becomes even more critical. Instead of assuming access is allowed unless specifically denied, the attitude when using the Internet should be disallowing access unless specifically required. Parties interacting with the Internet must be identified and access rights managed. Sensitive Information exchanged between sources inside the intranet and the Internet must be encrypted to prevent useful interception.

Packets flowing over the Internet will pass through many networks and machines over which neither the intended recipient nor sender have control.

The counter-weight to security is the new business potential available through the Internet. The estimated number of adult Americans accessing the World Wide Web (WWW) each day is now 9 million. Combine this with the WWW search engines that can find each of those users a page within your organization in the stroke of a dozen keys, and the potential new business is immediately obvious. It should be equally as obvious that the potential for fraud and abuse from those 9 million potential customers (and we haven't begun to consider the rest of the world) is also high. It is this trade-off between security and potential business opportunity that each organization must evaluate. Through diligent security precautions, it is possible to minimize potential security risks and still gain Internet access and business.

1.2 Security Concepts

Network security is a key organizational consideration in both the intranet and Internet communities. It covers a diverse range of areas such as physical security, the integrity of systems on the network, and protection of information transmitted across the network. Traditionally, security is thought to encompass three fundamental issues.

- *Authentication* establishes the validity of a user's (or application's) identity. This prevents a user masquerading as a trusted party from gaining unauthorized access to protected information.
- *Authorization* is the process of verifying the access rights granted to an authorized user (or application) to access restricted information or services. These rights are typically described as privileges such as read, write, or execute in the UNIX file system.
- *Privacy* is a security characteristic applied to information either being transmitted or stored which conceals it from users (or applications) who don't have appropriate authorization. *Integrity* ensures that transmitted information is not modified during transmission.

Security also needs to consider issues of auditing, non-repudiation (preventing replay of network transactions), and denial of service. Some form of cryptographic mechanism is key to many of these requirements.

Cryptography is the study of encryption protocols and algorithms. It is used to prevent access to information either in transit or on a storage device. The two fundamental methods for encrypting information involve use of a shared secret key or the combination of a publicly available key together with a matching secret key.

Each method uses a *key* together with a mathematical formula to transform plaintext data into an unreadable form, called *cyphertext*. Another key is required to decrypt the unreadable text data back into the original form by using another mathematical formula. The primary difference between the two methods is the key distribution method required. The symmetric method requires a shared secret key, but the public method allows the second, or public, key to be openly distributed with no loss of future security. These methods are described in more detail in the following section.

1.2.1 Private (or Symmetric) Key Cryptography

Private key cryptography is a very simple form of cryptography. In this case the same key is used to both encrypt and decrypt data using a shared mathematical formula. The mathematical formulas are computationally efficient, but the method has the disadvantage of requiring a shared key between the person encrypting the data and the other party decrypting the data. Often this exchange requires the physical interaction of the two parties.

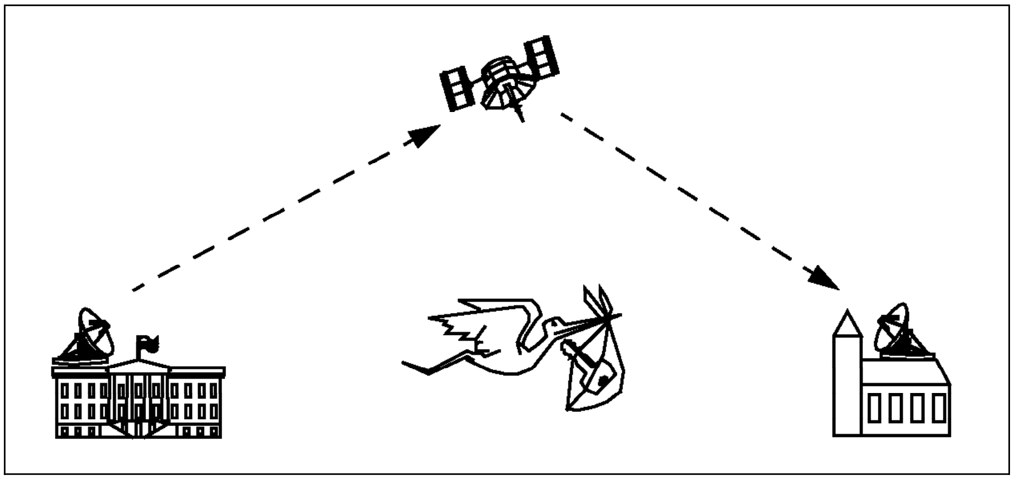


Figure 3. Private Key Cryptography Requires the Physical Distribution of a Shared Key

The best known private key technology is the Data Encryption Standard (DES), which was defined and endorsed by the U.S. Government in a 1977 standard. DES was originally developed by IBM and is the most widely used cryptography system in the world. It can be used to encrypt communication when both the sender and receiver share the secret key used to encrypt and decrypt the message. Alternatively, it can provide single-user encryption for securing information on disk. It uses a 56-bit key and was originally intended for implementation in hardware.

The next round of private key security being adopted by the U.S. Government is based around the Clipper Chip. This is an encryption chip announced in 1993 that uses a secret encryption algorithm called *Skipjack*. The chip is implemented in hardware by selected manufacturers, but adoption of the technology has been lukewarm as the security community is suspicious of an encryption technology without first seeing the algorithms used. The technology uses an 80-bit key and 32 rounds of scrambling in contrast to DES, which has a 56-bit key and only 16 rounds of scrambling. A software version of Skipjack was purportedly posted to the Internet in the summer of 1995, but it was believed to be a hoax.

The National Security Administration (NSA) is in the business of slowing the adoption of powerful encryption technologies so as ″to prevent national enemies from employing encryption methods too strong for the NSA to break″ (from the RSA (RSA Data Security, Inc.) This has led to rumors of back doors and weaknesses in NSA-endorsed encryption technology such as DES and Clipper. Such accusations are placing increasing pressure on cryptography-mandated cryptography standards.

1.2.2 Public Key Cryptography

Private key techniques do not scale well to a distributed environment. It is often difficult to securely exchange the shared private key without physically exchanging the key. Public key technology overcomes this problem through the invention of a mathematical relationship between a *private* key and a *public* key. This is based upon the mathematical complexity of factoring very large numbers into two prime factors. Unfortunately, encryption using a public key is computationally more expensive than private key technology, but key distribution makes it more applicable to the expanses of the Internet.

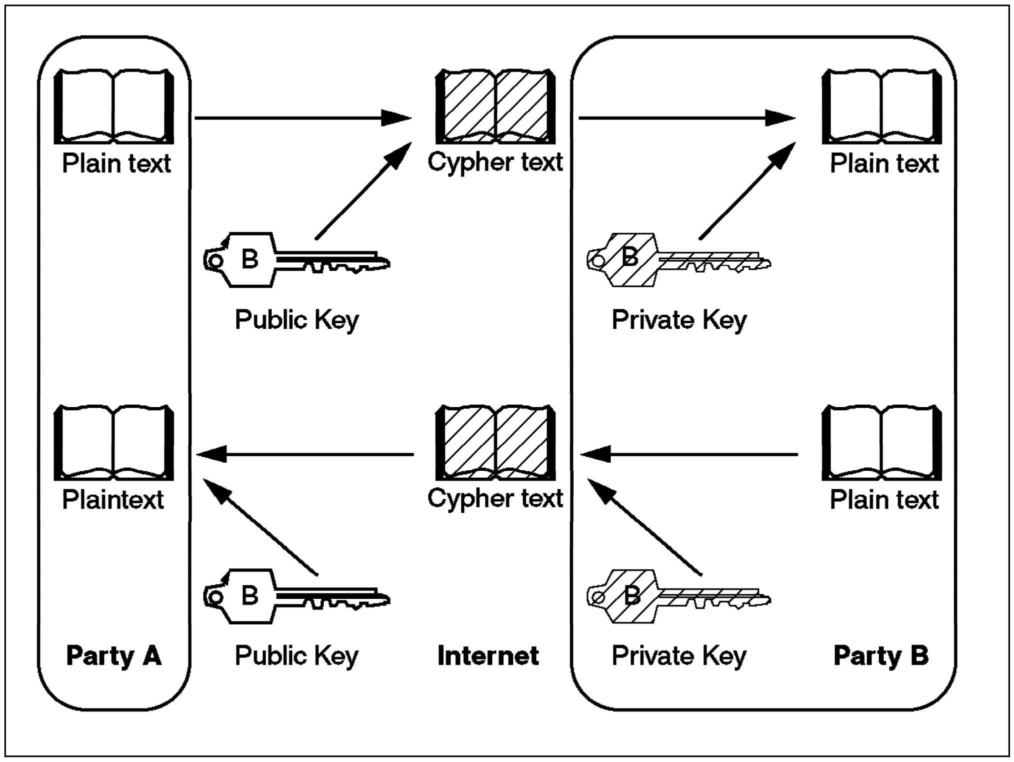


Figure 4. The Relationship between Public and Private Keys

Figure 4 illustrates the relationship between unencrypted *plain text* and encrypted *cypher text* when using public key technology across the Internet. Plain text at party A is encrypted using the public key of party B. This produces cypher text that can securely traverse the Internet. Party B can decrypt the cypher text using the private key. Party A can be sure that only party B can read the encrypted plain text since only the private key of party B can decrypt the message into plain text.

The reverse path through the encryption process is also possible. Party B can encrypt some plain text using its private key and send it across the Internet. Anyone knowing the associated public key can then decrypt the cypher text and be assured that it was published by party B. A version of this technique is known as a *digital signature*. For example, an encrypted name and address is appended to some text which when decrypted will verify the identity of the text's author. The reader should note that this doesn't have the same security semantics as the technique described in the previous paragraph. The cyphertext is readable by anyone with access to the public key of party B (that is, almost anyone). It only ensures authentication of the sender, not privacy.

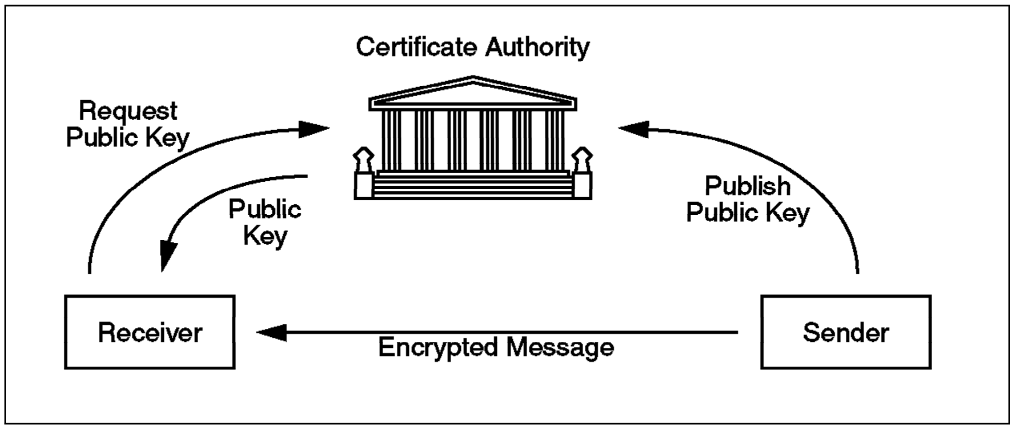


Figure 5. A Public Key Infrastructure

The public key infrastructure shown in Figure 5 requires some mechanism for the distribution of publicly available public keys. This is commonly known as a *Certificate Authority* (CA). Certificates have been standardized to include:

- The certificate issuer′s name
- Who the certificate is being issued for
- The public key of the subject
- Some time stamps

The certificate is signed using the certificate issuer's private key. Trust is now based on the validity of a CA's public key instead of the public key of everyone else. Please refer to 1.3.3.1, "Secure Socket Layer - SSL" on page 10, for a more detailed description of this process.

The most widely distributed public key technology is probably *Pretty Good Privacy* (PGP) written by Phil Zimmerman. This package is used throughout the Internet to digitally sign publicly available information as well as encrypt data being sent across the Internet. It uses a combination of public key technology (based on the RSA algorithm) to simplify key access and private keys (based on the IDEA algorithm) to actually do the encoding of data. This is a common security approach as the public key technology ensures easier key distribution, but the symmetric/private key technology is computationally more efficient. This duality of encryption methodologies is seen again in the next section on the Secure Sockets Layer (SSL).

1.3 Security Solutions for the Web

The World Wide Web is a relatively young child of the Internet, but in its short life it has introduced many to the mysteries of interconnection and perils of Internet security. Previously, Internet security was either non-existent, or users adopted a strategy of denial. There were very few middle-of-the-road solutions. The Web has introduced a more cosmopolitan attitude to information access and has forced players to adopt more complex security strategies that encourage access and connectivity while restricting undesirable behavior.

1.3.1 Firewalls

A firewall is a machine that enforces an access-control policy between two networks, usually an intranet and the Internet. It usually allows the specification of what is permitted through the firewall as well as what is explicitly excluded. Firewalls are designed to act as a ″sacrificial lamb″ in the event of a security breach. The firewall machine may be corrupted, but the machines on the intranet are still protected. Figure 6 shows a firewall protecting an intranet while the Web server sits exposed to the Internet. In this case both the firewall and Web server machines may be compromised.

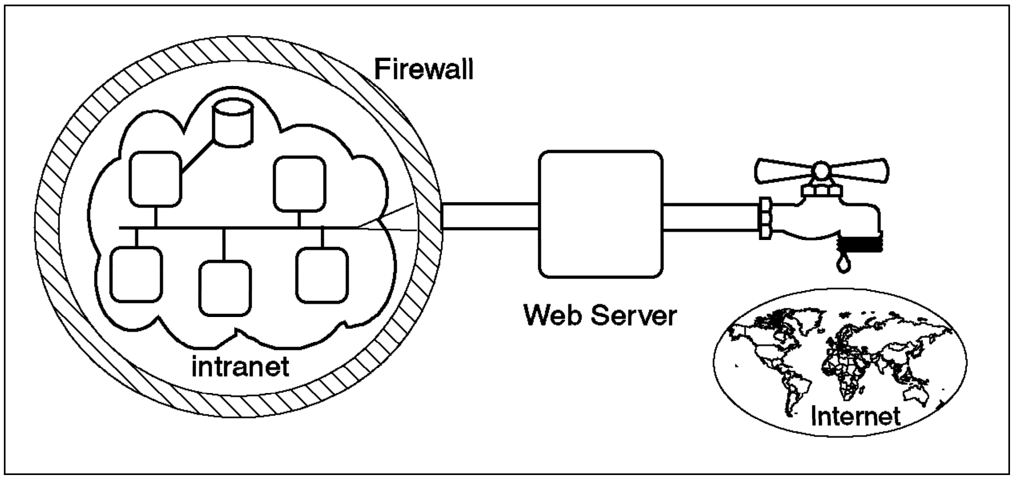


Figure 6. Firewall Protecting an Intranet

At one extreme, firewalls can be configured to only allow electronic mail through, but may be less restrictive by allowing all but known problem services to access. They prevent unauthorized logins from machines outside the intranet domain and act as a significant line of defense against denial of service attacks. If a problem is detected, it is a simple task to unplug one machine from a network rather than tens or hundreds of machines. Firewall logs audit traffic passing through the gateway, thus simplifying the problem of security-violation detection.

A firewall can only protect against information that traverses the firewall. Even in that case, it can't look inside electronic mail to ensure intellectual property isn't being sent. There are also other physical security risks such as information loss through Fax, telephone, or disk media. To be effective, a firewall must be part of an overall, effective corporate security architecture.

Section 8.1, "Principles of a Firewall" on page 143, describes firewalls in more detail.

1.3.2 Server-Based Access Restrictions and File Protection

The next layer of protection for a Web server is at the server itself. The Web administrator can use access restrictions to limit access to the web server. Three types of access restrictions are supported.

- Document and directory access can be restricted so that browsers connecting from certain IP addresses, subnets, or domains are permitted or denied.

- User names and passwords protect documents and directories on an individual basis. This is covered further in section 5.4, “DFS Access Control” on page 78.
- Encryption of document requests and of documents themselves restrict access to all but the intended recipient.

Restrictions based on IP address can fall victim to IP spoofing, a case where the browser pretends to be from another host. This can be made much more difficult to do if the Web server exists within a firewall; however, it also makes access to the Web server more restrictive. The firewall restricts external parties from pretending to provide legitimate services such as the Domain Name Server (DNS). However, if such restrictions are introduced, the firewall effectively cuts access to Web documents external to the firewall since the Web browser can’t map machine names to network addresses and so can’t get access to Web servers providing the documents. This illustrates the classic trade-off between access and security.

Web servers must run as a privileged user on UNIX systems so that they can offer their service at a well-known connection point at the host. It is recommended that they change identities to another less-privileged user to access documents and execute programs on behalf of foreign browsers. Otherwise, bugs or mischievous programs can gain access to all of the local file system and thus compromise security. Most Web servers, such as Netscape Corporation’s FastTrack Server or Enterprise Server, provide this optional identity change. The non-privileged identity can then rely on standard file system protection to limit access to parts of the file system or to the operating system.

Most Web servers add an additional layer of security based on user name and password. Individual users and groups of users can be allowed or denied access to directories and documents based on their identities. These are authenticated through a user name and password requested on access to a privileged part of a document space. The identity lasts for all privileged accesses on that server. IBM’s DFS Web Secure product provides an extension of this same concept by utilizing DCE/DFS authentication and security to access documents stored under a DFS file space attached to a Netscape Web server (see section 5.4, “DFS Access Control” on page 78).

1.3.3 Secure Transfer Protocols: SSL and S-HTTP

SSL and S-HTTP are authentication and encryption protocols for the Web. Each provides secure information transfer over a network communication channel accessed through a local *socket*, the communication endpoint. The protocols are not compatible, and in order to use one, the Web server and connected browser must both provide support for the chosen protocol. Netscape’s line of products, for example, support both of these mechanisms.

1.3.3.1 Secure Socket Layer - SSL

The Secure Socket Layer (SSL) supplies privacy and reliability between two communicating applications. It provides connection security with three properties.

- Privacy through an encrypted connection based on symmetric keys
- Authentication using public key identification and verification
- Connection reliability through integrity checking

These characteristics are attained while ensuring cryptographic security, interoperability across implementations, extensibility to new encryption mechanisms, and relative efficiency. SSL was proposed by Netscape and is commercially implemented in many different browsers including Netscape Navigator, Secure Mosaic, and Microsoft Corporation's Internet Explorer.

The operation of SSL is a typical example of cooperation between a client, server, and certifying authority. We will briefly run through the steps in this process; more detailed technical information can be found at Netscape's Web site at `http://home.netscape.com/newsref/std/SSL.html`.

The person managing the Web server must generate or obtain a public/private key pair and register the public key with a Certificate Authority (CA). Trusted CA's will require some physical verification of identity as well as company/individual information for describing the entity.

The CA will provide a digital certificate to the Web Server manager which is an encrypted copy of the Web server's new public key using the private key of the CA. The CA public key is assumed to be globally available and is one of the few trusted organizations able to operate as a CA. This digital certificate can be opened by anyone with the CA's public key, and it verifies that the CA has verified the Web Server's public key.

A user starts an SSL-aware browser such as Netscape's Navigator with a list of trusted CA's. For example, Netscape Navigator comes knowing about VeriSign and the United States Postal Service.

Once the server has established its identity and the SSL aware browser is started, the following sequence of actions take place for each new SSL channel, which is established for each new document:

1. The browser sends a ″hello″ message to the Web server, and the server responds with a copy of its digital certificate.
2. The browser decrypts the server's public key using the well-known public key of the CA (from its trusted CA list). The browser queries the user about talking to this site.
3. The browser generates two random numbers that will be used for symmetric key encryption, one number for the receiving channel and one for the sending channel. These keys are encrypted using the server's public key and then transmitted to the server.
4. The browser issues a challenge (some text encrypted with the send key) to the server using the send symmetric key and waits for a response from the server that is using the receive symmetric key.
5. Data is exchanged across the secure channel.

SSL takes advantage of key availability in the public key algorithm to authenticate identity, but uses the more efficient symmetric key technique for actual data transmission. Web servers use SSL as a secure alternative to the standard socket layer and will in future provide SSL versions of FTP, Telnet, and other Internet protocols.

1.3.3.2 Secure HTTP - S-HTTP

Secure HTTP is an extension to the HyperText Transfer Protocol used over the World Wide Web. It was developed by E. Rescorla and A. Schiffman of EIT (Enterprise Integration Technologies) to secure HTTP connections. S-HTTP provides:

- Confidentiality
- Authentication
- Message integrity
- Non-repudiation of origin

Whereas SSL is a layer over which HTTP (or even S-HTTP) can be sent, S-HTTP provides additional protocol elements for the negotiation and encryption of secure communication channels at the same level as HTTP, that is, the application level. The protocol is designed to be extensible and provides a choice of key management mechanisms, security policies, and cryptographic algorithms. These security options are negotiated for each secure transaction.

In similar fashion to SSL, the initial handshaking between client and server involves the negotiation of cryptographic information. S-HTTP partners exchange protocol messages detailing which cryptographic features they will accept. These options can be optional, required, or refused depending on the abilities of the parties. The negotiated cryptographic features cover various algorithms and mechanisms.

- *Privacy enhancements* cover the encryption scheme and may include combinations of signed, encrypted, and authenticated.
- The method of public key encryption for the *signature algorithm.*
- The method of *symmetric key encryption* used to ensure privacy.
- The *method-digest algorithm* defining a hashing function for generating message authentication codes (for integrity checking).
- A *key-exchange algorithm* for transferring symmetric keys (similar to SSL). For example, RSA public key encryption or Kerberos key distribution support.
- The *privacy domain* or message formats that the client and server will support. This controls the syntax for items such as digital envelopes, signatures, and certificates.

These cryptography features generally apply to both directions in an S-HTTP channel. Obviously, a large range of possible cryptographic option combinations exist, and these make for both an extensible and flexible security infrastructure.

1.3.4 Electronic Transactions

Electronic commerce has been hailed as the next phase in the consumer evolution. Ideally, an electronic commerce infrastructure will include retailers, banks, and consumers all interacting and using digital cash. The road to this ideal began with brave consumers sending their credit card numbers in plain text through either electronic mail or a Web browser. The first wave of security saw public key techniques applied to electronic mail where the retailer's public key was used to encrypt the consumer's credit card information. More recently, we have seen the introduction of SSL- and S-HTTP-aware Web servers that provide privacy and authentication as described in section 1.3.3, "Secure Transfer Protocols: SSL and S-HTTP" on page 10.

The missing ingredient in these schemes has been the adoption of digital cash. Credit cards provide limited authentication based on a name, number, and expiration-date combination. This is inadequate in an electronic world. This section describes several new businesses set up to operate in the digital cash arena.

1.3.4.1 First Virtual

First Virtual Holdings, Inc. operates the First Virtual Internet Payment System, an electronic payment system for commerce over the Internet. It was founded by a combination of academic computer scientists working on messaging systems and a California lawyer.

A new client registers with First Virtual through a two-step process. An online sign-up form is completed, and then a phone conversation is used to exchange credit card information for a First Virtual PIN. This Personal Identification Number (PIN) is used to make purchases with First Virtual registered retailers.

Security is provided by a callback scheme where First Virtual use electronic mail to verify purchases before processing the payment. Credit card information is never transferred over the Internet, and retailers use specially provided software to verify First Virtual PINs.

1.3.4.2 DigiCash

DigiCash is a digital cash system developed in the Netherlands by DigiCash. The system works much like a phone card system. Users purchase ″cyberbucks″ from a participating bank or remotely by using credit cards or money transfer. Cyberbucks are much like real currency as they can be exchanged for merchandise or traded among individuals. They manifest themselves as specially generated serial numbers which are prevented from being forged or spent more than once. In contrast to credit-card-based systems, cyberbucks can be spent anonymously and leave no audit trail. Cyberbucks can be redeemed for physical cash at participating banks.

The DigiCash technology is based on public key encryption, which forms a bond of trust between the consumer, the retailer, and a bank. Cyberbucks are digitally signed by banks and use an unpredictable challenge on the notes identity to prevent spending the same buck twice. If a note is challenged more than once, enough information exists to identify the consumer.

1.3.4.3 CyberCash

A digital wallet is the metaphor used for this digital payment system. A user downloads a wallet from the CyberCash Web site and initializes it with payment information such as credit card numbers or bank account numbers. This information is stored encrypted in the wallet. When a purchase is to be made, a window pops up at the browser asking which payment method to use and the retailer links in to the CyberCash site and records the transaction. The wallet also records information for later checking with credit card and bank statements.

CyberCash also utilizes public key technology. Payment information is passed from the wallet to the retailer encrypted with the wallet's public key. The retailer encrypts the encrypted payment information with its own private key and passes this on to CyberCash. CyberCash is then able to authenticate both the identity of the retailer as well as the payment information from the consumer. This method does not support the anonymity of DigiCash, but does provide the consumer with protection against fraudulent retailers in the same way as credit cards.

1.3.4.4 Secure Electronic Transaction (SET)

The Secure Electronic Transaction (SET) protocol is an open standard for processing credit card transactions over the Internet created jointly by Netscape, Microsoft, Visa, and Mastercard. An important objective of the standard was interoperability as well as the provision of secure digital transactions. SET uses public key technology to certify each party involved in a transaction, including the customer, merchant, card-issuer, and merchant′s bank. Each party has access to incomplete information of any transaction but enough for them to perform their required task. For instance, the card-issuer doesn't know what is being purchased, just the price, merchant, and customer. In this way a veil of privacy exists between roles in a transaction. This system doesn't provide anonymous transactions as does DigiCash.

Browsers can use SET by downloading appropriate Java Applets and/or ActiveX controls. Both Netscape and Microsoft have Web servers capable of supporting SET transactions.

1.3.4.5 Open Market Web Commerce System

Open Market, Inc. is a new company providing an electronic commerce system. It operates in both consumer-to-business and business-to-business environments. Secure transactions are provided using a combination of a commercial transactions manager, authorization manager, and access solutions for existing Web browser products. Security is built around the SET protocol.

1.4 Security in the Distributed Computing Environment (DCE)

The Distributed Computing Environment (DCE) is a joint development from multiple vendors, including IBM, who formed the Open Software Foundation (OSF) some years ago. OSF is a not-for-profit organization that evaluates and develops vendor-independent software components for distributed systems such as DCE.

DCE's security model comprises three roles:

- *Authentication* of an object (including applications and users) to verify a trusted identity
- *Access control* to manage the authorization of an authenticated object to perform operations on a resource
- *Data protection* to maintain the privacy and integrity of data transmitted between objects

Together, the roles provide a comprehensive security service for a distributed environment.

1.4.1 Authentication

The DCE authentication model uses the Kerberos shared-secret protocol, a symmetric key algorithm. Authentication relies on the existence of principal identities and secret keys. Typically, the client obtains validated credentials by providing a principal and an associated key. This is often done by a login program such as `dce_login`. Application servers obtain a valid credential set by pointing the RPC runtime at a keytab file which contains the servers secret key. Credentials act as temporary proxies for a client's secret key.

Servers base their access control on the authentication status of a client and then on the client's set of authenticated privilege attributes. A client, on the other hand, can insist on an authenticated server or, more precisely, on a few known groups or principal names. An authenticated RPC includes the credentials as part of the call thus allowing the server to verify the identity of the client.

1.4.2 Access Control

Authorization for a client to access a resource at a server is managed through Access Control Lists (ACLs), but the authorization process is actually more complicated than this. The following authorization steps are typically performed by an application server.

- Check whether the client has provided authentication credentials.
- Verify that the protection level is acceptable.
- Determine an acceptable authorization service. This is either based on the principal name, credentials of the client, or no authorization checking is done at all.
- Check that the principal the client requested is acceptable to the server.
- Finally, extract the client privileges and check access control.

An Access Control List (ACL) is a list of tuples (that is, a database entry) containing a type, key, and set of permissions. Applications and DCE services are able to define their own access control types through an ACL manager. For DFS, the useful types are outlined in the following table.

Table 1. Access Control List Types for DFS

user	**A specific local principal (subclass of foreign_user)**
group	**A group of local principals (subclass of foreign_group)**
other_obj	**Other local principals not in user or group entries**
foreign_user	**User entry with a foreign cell reference**
foreign_group	**Group entry with a foreign cell reference**
foreign_other	**A list of foreign cells**
any_other	**Authenticated principal from any other cell**
unauthenticated	**An unauthenticated principal**

Access control is checked by iterating through the set of ACL entries associated with the server principal, and for each member of the client's credential permission set, it checks for the first match with a privilege attribute. Privileges are interpreted by the server and can include *read*, *write*, *execute*, *control*, *insert*, and *delete* permissions.

The combination of ACL types and privileges provide significant advantages over traditional file system access control. Types specifically deal with foreign cell access within a distributed environment and provide fall-back for both foreign and unauthenticated sets of users. Traditional UNIX access has to provide alternate semantics for privileges associated with directories and files. For example, the execute permission means execution ability for a file and access but not necessarily read for a directory. This is resolved in DFS with the more extensive set of ACL privileges.

1.4.3 Privacy and Integrity

DCE provides a variety of protection options as well as optional integrity checking for authenticated RPC. The protection level is agreed upon during the binding of a client to a server. More protective privacy schemes are a trade-off against performance.

- *None* is the default protection level, providing no security.
- *Connection-level protection* encrypts the initial handshaking protocol that is undertaken during the binding of a client to a server. No encryption of application data is performed.
- *Call-level protection* applies similar encryption to the protocol underlying each RPC call over a connectionless channel such as UDP.
- *Packet-level protection* applies to each packet required to perform an RPC and ensures that the packet came from the expected client.
- *Packet-integrity protection* ensures that data transferred between the client and server is not modified. This is accomplished using a cryptographic checksum attached to the packet.
- *Privacy protection* encrypts all RPC data and performs the integrity and protection actions listed above.

Specification of a desired protection scheme is left to the client. There is an option to use a default protection scheme provided by a server. The privacy algorithms available in IBM's DCE implementation include the export-restricted DES and an additional exportable Commercial Data Masking Facility (CDMF) cryptographic algorithm. CDMF is also an optional algorithm for use with S-HTTP.

1.5 DCE Security and DCE Applications on the Web

The network-centric model (see 1.1.2, "The Network-Centric Computing Model" on page 3) has transformed the business network into one with several faces. The private face lives behind a firewall, offering desktop integration and information access within the intranet. The public face of the network extends beyond the physical bounds of the organization and reaches far across the Internet to access information and services.

Each of the networking elements requires a comprehensive security policy and appropriate security architecture to protect access to data and services. In the past, applications provided their own security mechanisms, but this solution does not scale well. DCE and DFS are an ideal solution for an enterprise-wide security infrastructure. The authentication, authorization, and privacy services not only work within an organization but also scale for external organization access through linked cells.

Of special interest is the application of DCE security services to the Web. For many organizations, the Web bridges the corporate intranet and external Internet. It is the ideal candidate for a security infrastructure such as DCE which provides a similar bridging service for security. This book looks at several new technologies for exploiting DCE security across a Web environment.

1.5.1 IBM DFS Web Secure

Web servers such as Netscape's FastTrack and Enterprise servers provide a minimal authentication and authorization service for document access. When a browser comes across a protected page, the user is prompted for his/her name and password. If correct, document access authorization is verified. Authorization is restricted to read and write privileges based on either individuals or groups. Both the group and user databases are maintained within the Web server and thus form another isolated security infrastructure which neither allows for a scalable distributed environment nor for administration consistent with other applications.

IBM DFS Web Secure overcomes these problems by linking Web document access to DFS and to the DCE security infrastructure. Instead of being authenticated and authorized through the Web server, the identity and access tasks are passed through to DCE. This occurs when the browser attempts to access any document stored under DFS via the Web server. Authentication and authorization administration occurs just as normal in the DCE/DFS environment and is fully utilized by the Web server. This mechanism is transparent to the browser and is implemented as a plug-in for the server.

The authorization functionality is extended from the standard Netscape server model by taking advantage of the additional privileges and authorization types available under DCE. For example, DCE differentiates between directory browsing and access in a similar way to the UNIX file system. Netscape servers do not include the same distinction and instead allow access through directories even when read permission is denied. DFS Web Secure cleans up these inconsistencies and brings together security administration under a single umbrella. It also provides a scalable document-storage environment by utilizing the ubiquitous DFS file environment.

IBM DFS Web Secure is discussed throughout this book in Chapter 3, “Strengthen Server Security: IBM DFS Web Secure” on page 29, in Chapter 4, “DFS Web Secure Installation and Customizing” on page 43, and in Chapter 5, “Running the Business With DFS Web Secure” on page 63.

1.5.2 DE-Light Web Client (Transarc Corp.)

Transarc's DE-Light Web Client brings DCE server access to the browser. Client programs are written in Java as applets. When the applet is referenced from the server (see Figure 7 on page 18), the applet is first downloaded, then several DE-Light helper classes are acquired from the same server.

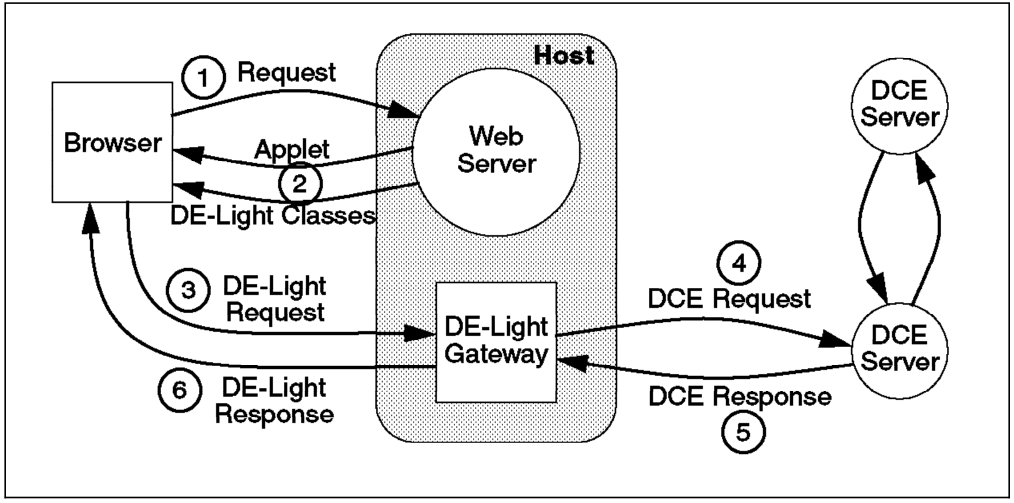


Figure 7. Transarc DE-Light Web

These Java classes talk to a *DE-Light Gateway* using a lightweight, Java-friendly protocol. Java applet security restrictions only allow the applet to contact the same host they were loaded from; so the DE-Light gateway must be on the same host as the Web Server. The gateway then transforms the request into a DCE RPC call and passes it on to the appropriate DCE application server. The response is returned from the DCE application server to the gateway then transformed back into the lightweight protocol to be returned to the Java applet.

DE-Light Web leverages the advanced programming features of Java to provide simple client development. In addition the applet architecture ensures the client always gets the latest version of the client application as well as DE-Light helper classes. Along with DCE RPC, DE-Light Web offers access to the Encina Transactional RPC (TRPC) API and to the DCE security services.

Transarc's DE-Light Web Client is explained in more detail in Chapter 6, "DE-Light Web Client" on page 109.

1.5.3 Lotus Domino and DCE Integration

Distributed Systems Series Link for Lotus Domino (DSSL for Lotus Domino) is a set of tools to integrate Lotus Domino and DCE that allows the Domino programmer to access DCE servers based on their interface definitions (IDL). A Dynamic Link Library (DLL) is generated to support access to the server based on an IDL file and a source-code template. In addition LotusScript which is required to access the DLL, is automatically generated.

The DLL and LotusScript stubs are attached to a Lotus Domino project database. The DSSL for Lotus Domino Library Distribution tool provides the DLL modules from the project database to individual workstations. The DSSL for Lotus Domino runtime automatically ensures that the latest version of the DLL is made available to the user. In addition the runtime stores the selected security level and options as well as an entry in the Cell Directory Service (CDS) from which to acquire access to the server.

DSSL for Lotus Domino extends the family of DCE-aware clients to include the Lotus Domino suite of applications. The developer benefits from a simplified programming interface while still gaining access to the range of DCE services and servers.

More details about this toolkit can be found in 7.3, “Distributed Systems Series Link for Lotus Domino” on page 137.

1.6 The DCE Distributed File System (DFS)

The Distributed File System (DFS) is a non-core element of DCE that uses the components of DCE to provide its services.

- DCE *Remote Procedure Calls* (RPC) provide transport for the file data and other management functions within DFS.
- DFS uses the *Cell Directory Service* (CDS) to provide a uniform, distributed namespace. DFS file names are cell-relative and within the cell are location independent. This allows transparent file access in spite of files moving, users moving, or new file servers being added to the environment.
- The DCE *Security Service* ensures private and secure file access. Authenticated user access files based on their privileges and the Access Control Lists (ACL’s) associated with directories and files.

DFS provides performance advantages over other distributed file systems through its use of local caching and replicated server access. It implements full POSIX single-site semantics for files, including byte-range locking. DFS security takes full advantage of that provided through DCE. All files and directories can provide Access Control Lists, allowing full use of ACL types including foreign users and groups.

Traditional UNIX file security includes three types of access: user, group, and other with permissions of read, write, and execute. DFS security extends these as described in section 1.4.2, “Access Control” on page 15. This extended access security, together with its scalability through server transparency, make DFS a better alternative for storing Web data than any traditional operating system’s native file system. Another advantage is that files do not need to be physically copied onto the Web server machine to make them accessible.

1.7 Security Requirements

Security is a state of protection against danger, especially espionage or sabotage. It protects information both within as well as that flowing in and out of the organization. We have already identified three security requirements explicitly.

- Users, applications, and other identified principals must be *authenticated* to verify their identity before further steps are taken. This ensures a trusted relationship between the security environment and the principal.
- Once a trusted identity has been established, we are able to establish the *authorization* the principal has for accessing resources within the environment. This is usually accomplished by evaluating a set of privileges possessed by the principal against access controls provided with the resource. DCE (and DFS) uses privilege certificates and ACLs to accomplish this.
- Remote access to resources accomplished through appropriate authorization for an authenticated principal does not ensure that the information is not inadvertently made available to an untrusted third party during transmission.

Privacy provides some encryption mechanism to secure information that is inadvertently seen by others.

These constitute the traditional aspects of security that are regarded by many as the essential components. However, there are several more security risks that must be countered.

- *Privacy* ensures information is not unintentionally leaked to others, but doesn't ensure that the information isn't tampered with along the way. By combining some form of encrypted checksum with the data, we can verify the *integrity* of any information that is transmitted. This is often available as part of the privacy facility.
- *Non-repudiation* is a security feature that prevents the recording and replay of messages sent between a client and server. Even though they may be encrypted and integrity-checked, replay must be prevented. This can be achieved using some identity check encrypted in the message transmission. An obvious example of needing this service is replaying deposit information between an ATM and a bank.
- One of the most difficult security risks to counter is *denial of service*. In this case a third party manages to restrict, or deny completely, access to a service. For example, this can be accomplished by repeated brute force attacks on an authentication service that denies access to legitimate clients.

Even with a well-defined security policy and a strong security infrastructure, many security problems can still occur. You can lead a user to security, but you can't make them use it (easily). A ubiquitous *audit* service is another important element of the security toolbox. It acts as both a deterrent to attack and is also one of the most valuable tools for evaluating and cleaning up after an attack.

This vast array of security-related services makes for a broad security umbrella, but can also lead to an increased *administrative* load. Even the relatively simple task of managing access control can grow exponentially if careful planning of a scalable security strategy isn't put in place early. For example, using users as the default security unit will soon become overwhelming; so a standard group structure should be set up at the outset. DFS provides a single security infrastructure that eases this administrative burden through shared authentication and authorization mechanisms spanning DCE application servers and the Web.

Chapter 2. The Business Requirement

The two major technology areas that underlay this book's subject area - the Distributed Computing Environment and the World Wide Web - have both been implemented now for several years by a wide range of organizations. This chapter covers some business scenarios where a combination of technologies from both of these areas can offer real business benefit by solving some technical challenges in today's world of distributed computing.

2.1 Sharing Information on a Company Intranet

In this scenario we consider a company where data is being shared already at a departmental level with departmental file servers. There are projects underway, however, where information needs to be shared across departments. For instance, Marketing needs to access some of the introductory documentation that is being prepared in Product Development. Currently this access can be arranged by people in the respective departments contacting each other by phone and distributing the information through e-mail on request. To make this type of information generally available to departments that may need to use it, the company has set up an internal Web server on the internal network (or intranet). This means Product Development can transmit the documentation files to a Web administrator for placement on the internal Web server. Along with the files, the file owners need to specify which departments, or even which individual users within those departments, are permitted to access the files. The Web administrator can then place the files on the internal Web server and set up the necessary access controls for those files. This scenario is shown in Figure 8 on page 22.

This company now has a general-purpose approach for sharing information between departments where users have access to standard Web browsers. Now it is possible for users in a number of departments to access the files of other departments that are of interest to them. This can be particularly useful where the files are authored in specialized tools that can produce output in a format that can be read by a Web browser. An example might be a design department using a CAD package, or most text-processing tools.

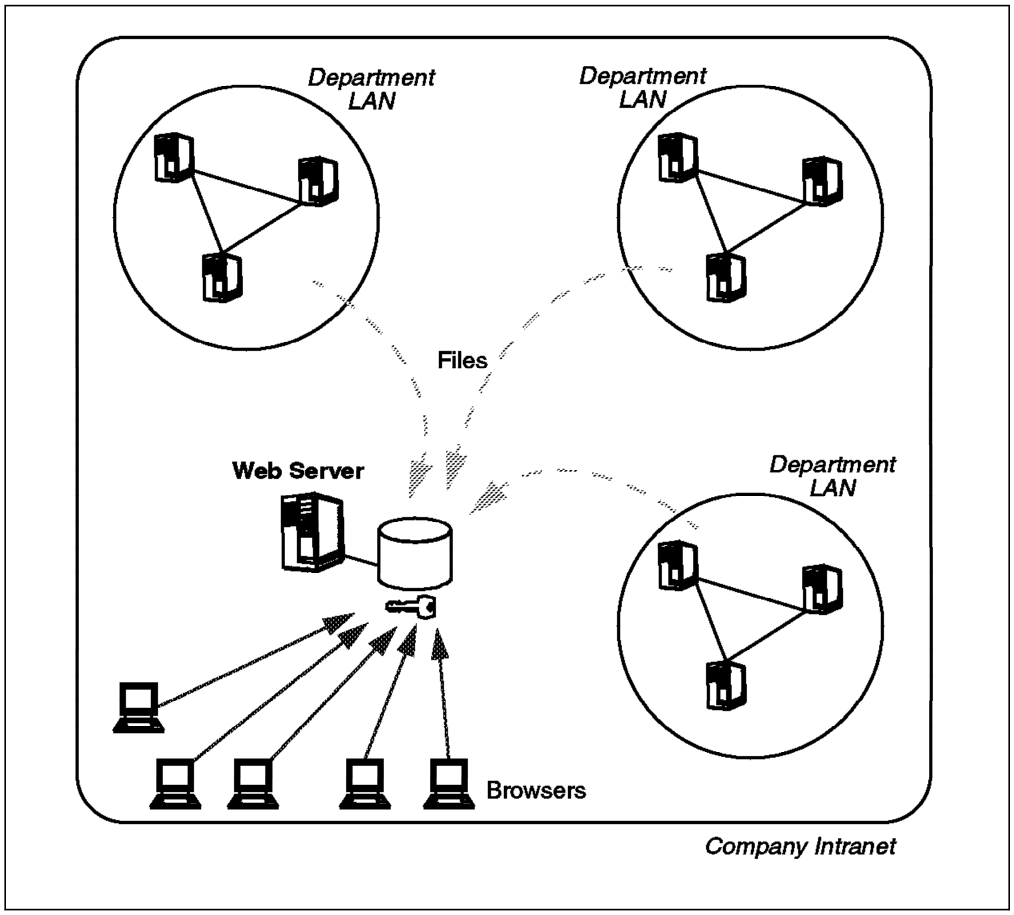


Figure 8. Sharing Information on Departmental Systems

What are the concerns in this scenario?

1. There is an administrative workload on the Web server in this scenario that includes the placement and maintenance of the files that are being served, as well as the setting up and maintenance of the access controls for those files. A database of users and groups also needs to be maintained on this Web Server system to represent the users across the company's departments who will be using the shared information. The Web server administration function (maybe one person) needs to communicate with each department that is making shared information available and has to be involved with every shared file update or change to access control information that is required. However well resourced this Web administration function is, there is certain to be some procedural delay for any changes to shared documents to reach their intended audience around the company.

2. Shared file data and updates to it are going to be sent and stored on the company Web server. These copies of the original file can soon be out of date, and they are also taking up duplicate amounts of disk space on two (or more) systems across the company. The costs of this data redundancy can soon mount in terms of disk storage, and there is a further cost associated with the processing of shared file updates through the Web administration function.

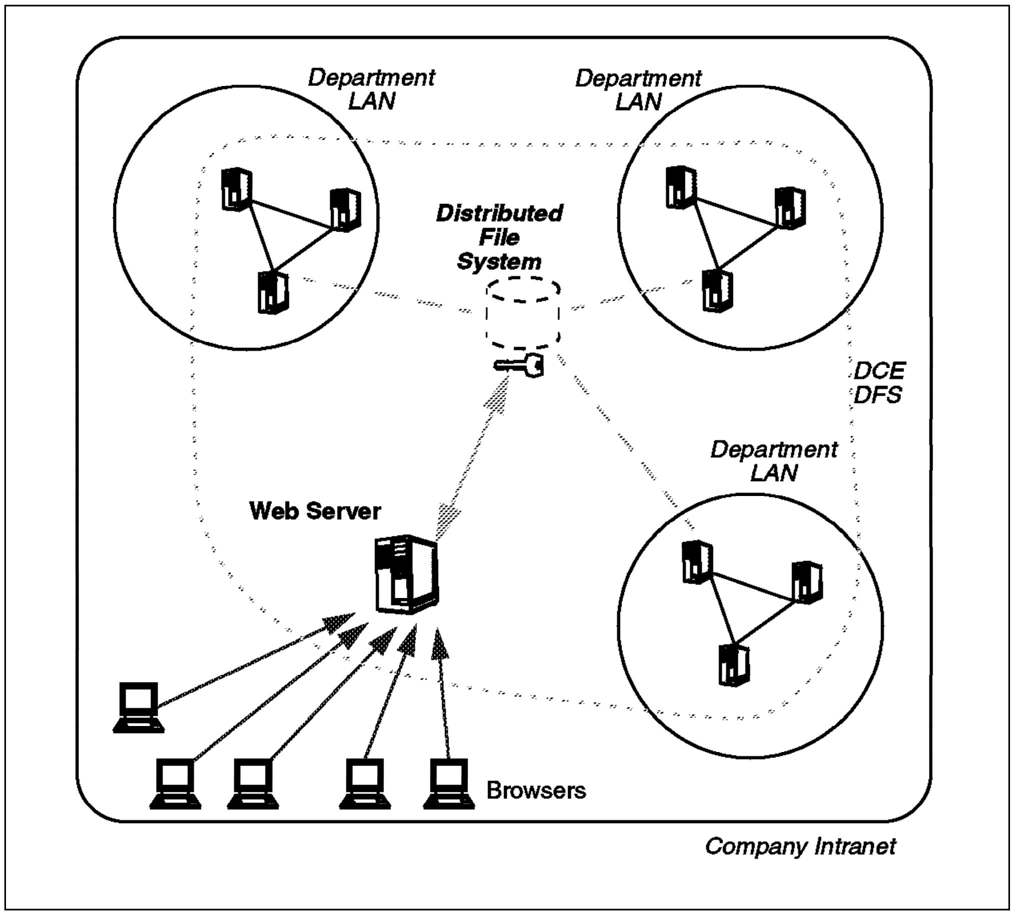


Figure 9. Sharing Information on a Company-Wide File System

Using the Distributed File System in this scenario can address these areas of concern and provide some additional benefits to the sharing of department information. Figure 9 shows the placement of the department files within a DFS file space setup. This means that there is now a single, company-wide file system in which all departmental information can be stored.

With DFS Web Secure, as introduced later in this book, it is now possible for the company Web server to provide access to the whole DFS file space, without having to bring copies of the shared files onto its own disk storage. The advantages of storing information in DFS in such a scenario are:

1. Security and access control is handled by DCE and DFS. In particular, file access control can be now be controlled on a file-by-file basis by the owner of the file. Hence, the department can look after the access control on its own information and does not have to involve a separate Web administration function with access control. Even individual users can grant permissions on their personal files so that they too become accessible to other users or departments across the intranet.

2. Departments have responsibility for their own data, including the information that will be shared around the company. File updates will be accessible straightaway to permitted users in other departments since they are able to access the actual file itself and not the Web server's last copy of it. This improves the currency of the shared information and eliminates the burden of copying it around. DFS technology takes care of consistency in case several users are looking at a file while it is being updated.

3. The requirements for Web administration are simplified. The previous points have covered how responsibility for file placement and access control remains with the owners of the files. The Web administrator can then concentrate on managing the Web server system in terms of its performance and availability.
4. DFS allows great flexibility in storage strategy while maintaining a consistent, logical view of the files stored within it. This means that files can reside on file servers distributed around the company, perhaps located in each department, or they can be located on large centralized file servers. DFS can be used with both strategies, and it can even maintain the logical view of the company wide file space while it is being changed and files are being moved between different file servers.

The use of DFS in a Web server environment, utilizing the DFS Web Secure product, is covered in Chapter 5, "Running the Business With DFS Web Secure" on page 63.

2.2 Enabling Remote Access to Company Information and Services

In this scenario we consider a manufacturing company that has a network of dealers selling its product lines. The dealers are separate companies that have a contractual arrangement with the manufacturing company, which makes them partner organizations, building a *virtual enterprise* together with the other dealers and the manufacturing company. This is an arrangement seen in many industries besides manufacturing, for instance in insurance companies with their insurance brokers and franchise operations. In this scenario, the dealer needs to have access to a number of business applications within the manufacturing company. These applications provide the following types of function:

- Allow enquiries about orders placed by the dealer, including their current status
- Present information about special offers currently available
- Show parts lists for the servicing departments at the dealers, including the facility to submit orders
- Show availability information for all product lines

This manufacturing company has moved to a client/server strategy for its business applications, and a number of these applications are implemented, or will be implemented, as DCE application servers. The Distributed File System is also implemented in this company and provides a uniform file space across all departments and functions. Marketing information in the form of brochure documents, product-specification sheets and press clippings are stored in DFS. Some of this information is being used to build a new Web site for the company to advertise and promote its products to the Web audience worldwide. The company function handling the dealer network wants to extend this information for the dealers and make it available to them over the same Internet network but not available to the whole Web audience (which includes competitors).

Some of the major dealers have business systems that are also using DCE technology, and there is a desire in the manufacturing company to be able to access some information in these systems also, for instance to see the stock levels at the dealers, which can help production planning in the manufacturing company.

Finally, some of the manufacturing company's employees go on extended road trips visiting existing and potential dealers, and they would find it useful to be able to remotely access some of the company's DCE applications and information stored within DFS. These requirements for extending access to this company's DCE infrastructure across the Internet are shown in Figure 10.

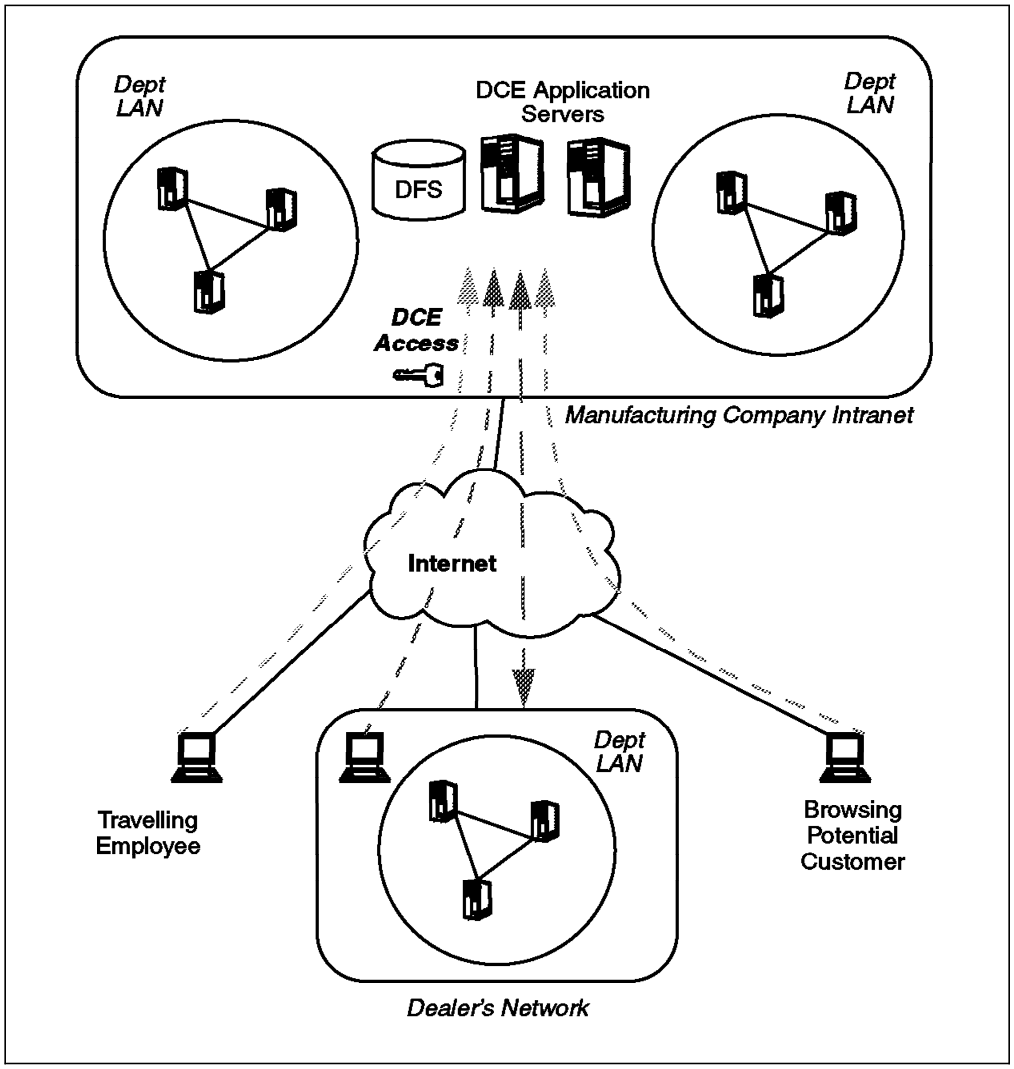


Figure 10. Enabling Remote Access to Company DCE Services

What are the key challenges that must be tackled in building a solution to meet these requirements? These and some other important factors are considered here:

1. Network security, when attaching the company network to the Internet is critical. A network security policy must be developed for the Internet connection, and this can be implemented with firewalls, which are covered in Chapter 8, "Bridging Networks Together: DCE Over Firewalls" on page 143.
2. Information- and application-level security is equally important here. This manufacturing company is looking to permit three different types of users to access information and DCE applications with correspondingly different sets of access rights. For Web access, many companies handle different requirements for access rights by providing separate Web servers for each type of external user. Here, this would mean setting up three separate Web servers; this involves the maintenance of three sets of Web server contents

as well as maintaining separate access control structures on each Web server.

With DCE and DFS installed in this company, there is already the means to establish access control definitions for many different types of users. With DFS Web Secure installed alongside a single, generally available Web server, it is possible to serve the information directly from DFS to these different types of users, all subject to the standard DCE/DFS access controls. Partners in the virtual enterprise can access and even update the same data stored in DFS, or they can run protected DCE applications over the Internet to which others do not have access. This is covered in Chapter 5, “Running the Business With DFS Web Secure” on page 63, which also looks at ways to access native DCE applications over the Internet by using a range of programming approaches and DFS Web Secure.

3. For remote access to DCE applications, it is desirable to make the requirements on the client system as light as possible. This is particularly true of any DCE application that is going to be accessed by a prospect or new customer who is connecting to the company for the first time over the Internet. Such a user is unlikely to have a DCE client runtime on his/her system and, if he/she does, it will certainly not be configured as part of the manufacturing company’s DCE cell. To open up DCE application access to lightweight clients in this way requires some gateway technology that can be used across the Internet. An example of such a technology is DE-Light Web from Transarc Corp., which allows DCE applications to be accessed from standard Java-enabled Web browsers. This approach is covered in Chapter 6, “DE-Light Web Client” on page 109, and can also be used by client systems within the manufacturing company’s intranet.

4. The manufacturing company does not want to be forced to add complexity to the applications and services that will be exposed to remote access. For DCE applications written with a security model (DCE supports several ways for applications to apply security), there will not need to be any specific changes made to allow remote access. Enabling remote access is then a matter of configuring the access controls for the application and establishing the network connection with the appropriate level of network security.

   There may be other types of applications that the company is developing that are going to be opened for remote access and that need to access information stored in DCE applications. One such example is the groupware product Lotus Domino, which enables the development of applications for groups of users which can incorporate functions to control and monitor workflow among users and departments. With the Distributed Systems Series Link (DSSL) for Lotus Domino toolkit, it is possible to create Lotus Notes/Domino applications that can access DCE application servers. These Lotus Notes/Domino applications can be made available remotely to clients running standard Web browsers. This technology is covered in 7.3, “Distributed Systems Series Link for Lotus Domino” on page 137.

5. There may be some traditional mainframe-style applications that can also be presented as DCE application services for other applications to use. The manufacturing company can use DCE on the mainframe to add these traditional mainframe applications to the set of DCE application services that can be accessed remotely in the manner described in this scenario.

   DSSL for Lotus Domino, in conjunction with OS/390 DCE Application Support (AS), can provide an easy way to access existing CICS and IMS applications from standard Web browsers through a Lotus Domino Server. A major advantage of such a solution is that programming for further application enhancements and new user interfaces can easily be done within Lotus Domino, rather than at the mainframe, where adequate skills might not widely be available. OS/390 DCE Application Support is not discussed any further in this book.

6. Setting up a peer-to-peer DCE link with the dealers' DCE cells can be tackled in a couple of ways. A dedicated network link can be set up between the two companies to carry DCE traffic over TCP/IP, and trusted DCE accounts can be created in both cells to allow intercell communication across this link. This would enable DCE applications in either cell to be accessed by clients in both cells (subject to DCE access controls). To establish such a link across the Internet, both companies will require some additional network security measures to be in force to prevent eavesdropping and unauthorized access. These can be implemented with firewalls. If both companies are using certain compatible firewall products, then it is possible to set up a secure IP tunnel across the Internet that will carry all of the TCP/IP traffic between the two companies in encrypted form between the two companies' firewalls. Firewalls are covered in Chapter 8, “Bridging Networks Together: DCE Over Firewalls” on page 143.

2.3 Using the Internet as a Backup Communications Channel

In this scenario we consider a company that has a DCE infrastructure in place and some smaller, remote locations linked to the main company network by dedicated wide area links. To ensure high availability of these links between the remote locations and the main DCE servers in the company DCE cell, there is a requirement to have an alternative means of establishing a DCE connection (and therefore a TCP/IP connection) between each remote location and the main DCE cell. A relatively inexpensive alternative for these fallback links could be to have an Internet link configured on both the central network and on the remote location network that is ready to operate at short notice. To ensure the security of DCE traffic passing across the Internet, network security measures will be required that can be implemented with firewalls and that possibly include a secure IP tunnel configured between compatible firewall products at both Internet links. Firewalls are covered in Chapter 8, “Bridging Networks Together: DCE Over Firewalls” on page 143.

The extended DCE cell and the backup Internet link in this scenario are shown in Figure 11 on page 28.

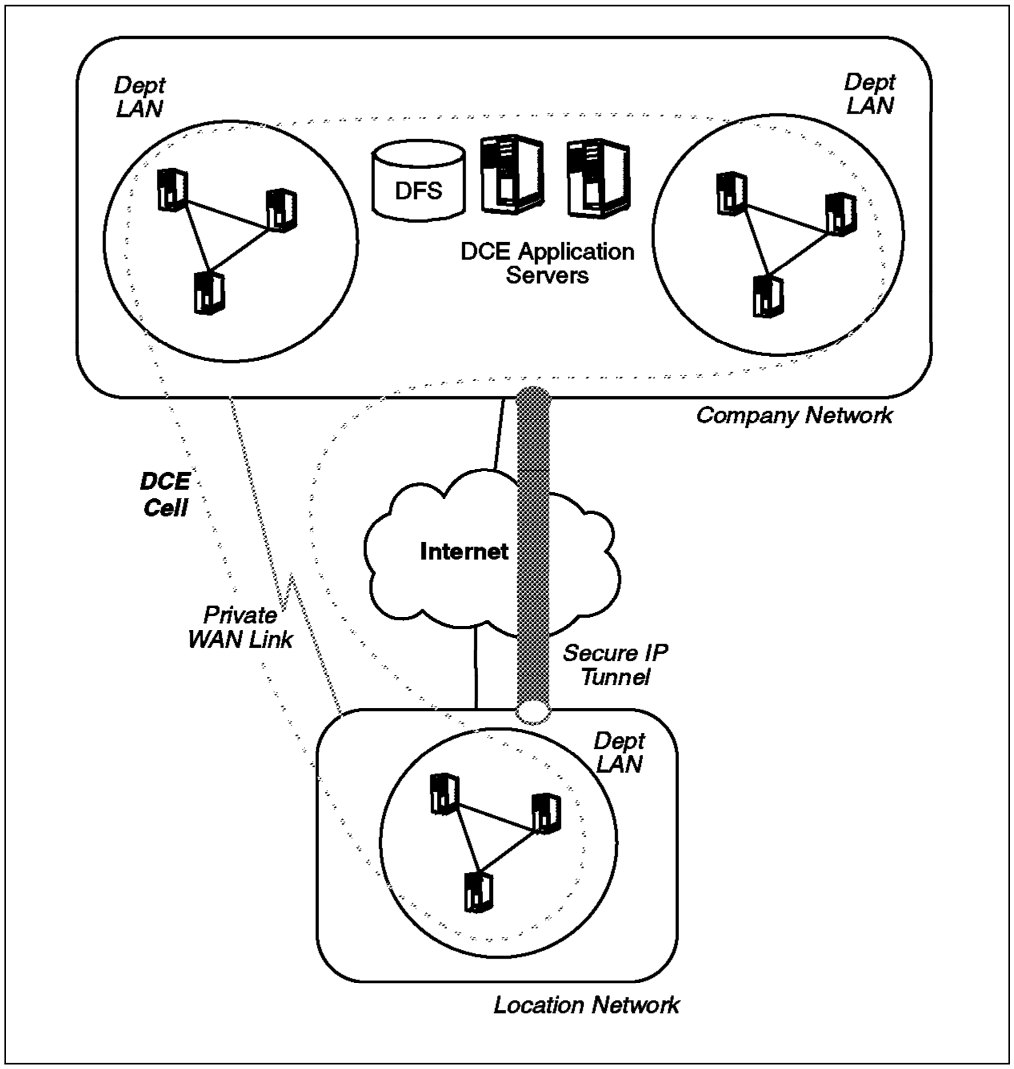


Figure 11. Scenario: Using the Internet as a Backup Communications Channel

These scenarios show a selection of business requirements for the use of DCE and DFS in the context of today's Internet/intranet networks together with their associated Web browser and light-weight client focused approach to application access. There are other business benefits associated with DCE and DFS that these scenarios have not highlighted, including:

- Availability of DCE services, applications and files within DFS through replication of key services and DFS filesets
- Scalability
- Maintainability
- Location independence of DCE applications and files within DFS
- Auditability

Chapter 3. Strengthen Server Security: IBM DFS Web Secure

The World Wide Web (WWW) is a very large-scale distributed system comprised of client browsers interacting with Web servers. It began life as simple document distribution tool in one organization and now infiltrates almost every computing desktop. The Web has redefined some definitions of scalability as it envelops new application areas. The relatively short history of the Web and a description of the current architecture provide a good perspective on future trends in this exciting industry.

Netscape Corporation provides several technologies in the Web marketplace. Their FastTrack Web Server is aimed at small to medium businesses requiring a presence on the Web in a self-contained package. It includes fundamental features of authentication and access security, Java/JavaScript support, and remote management through the Web. The Enterprise Server is a superset of the FastTrack Server, but also includes document management facilities such as a revision control system and full search capabilities. Both servers support extension through the Netscape Server Application Programming Interface (NSAPI). This defines software hooks through which new functionality can be added to a server based on document reference types.

IBM DFS Web Secure is an example of a Netscape Server extension utilizing the NSAPI. It introduces DCE security and DFS file access to the Netscape Server by providing specialized security and access mechanisms for documents residing in the DFS filespace. Security management can be coordinated through DCE with single-user and access-control databases instead of requiring management of both Netscape and DCE security environments. Management for DFS Web Secure is provided through the Web by using DFS Web Secure technology to authenticate and authorize the system manager. Access to the global file space in a scalable and secure manner helps solve part of the scalability issue that the Web architecture brings to the table.

3.1 A Review: Web Servers and Browsers

Scalability has found a new meaning under the explosive growth of the World Wide Web. From its early origins as an information distribution tool, it has blossomed into a heterogeneous, lightweight client deemed suitable for any intranet or Internet environment. In this section we examine the Web from an historical perspective and give some foundation for the technologies involved. The architecture of the Web is examined in more detail from both a browser and server perspective. As components in the most widely deployed distributed system, each offers a series of tools and features designed to extend their deployment environment as far as possible while maintaining their architectural direction.

3.1.1 Where Did the Web Come From?

The World Wide Web (Web, WWW) traces its beginning back to a tool developed for use within the European Organization for Nuclear Research (CERN). In 1989, Tim Berners-Lee proposed a system for managing information distribution and access within CERN. His proposal was based on the hypertext mechanism where hotspots in a document can redirect the reader to related information. He suggested that a system based on hypertext and managing information relationships between changing people, hardware, software, projects, and documents have the following characteristics.

- *Distributed* across multiple hosts
- Support for *heterogeneous* environments
- *Decentralized* information storage and access
- Access to *legacy* information sources such as databases
- *Extensible* by users to add new information and links to other information

By 1991, CERN had begun use of the World Wide Web internally with its own Web browser and server. In August, the first Web information was made available over the Internet, and the tools were announced to the world. The following year saw the development of several new browsers for X-Window-based systems as well as another Web server developed at the National Center for Supercomputing at the University of Illinois. The CERN and NCSA servers were to became the two dominant servers, available to anyone as public domain software.

The Web server population went from 26 in 1992 to over 200 by October 1993. The growth was exponential and was not slowing. 1993 also saw the release of NCSA Mosaic for X. This popular tool became the de facto standard for browsing technology and was the seed for the next movement in the Web world.

The growth of Web traffic and servers was still on a rapid rise through 1994. There were over 1500 servers now available around the world, and the first two international WWW conferences were roaring successes. In July of 1994 the World Wide Web Consortium (W3C) was formed to oversee the development and standardization of the Web. The following March most of the NCSA Mosaic team left to form *Mosaic Communications Corporation* which was to become *Netscape Communications Corporation*, a company built directly on the future success of the Web as a social and business infrastructure.

Beside Netscape, many other companies share the fast growing market of Web server and browser software and participate in the development of supporting tools and products around the Web. A recent survey by the Webcrawler (`http://webcrawler.com/`) shows 85 percent of Web servers to be UNIX-based with a server software distribution led by Apache, NCSA, Netscape, and Microsoft. Apache (A PATCHy) is a public domain Web server derived from the NCSA Web server and maintained by a group who originally applied patches to the NCSA Web server (and hence the name).

3.1.2 The World Wide Web Architecture

The World Wide Web is an information distribution architecture that uses the Internet infrastructure to share information. A Web browser and server combination glues many diverse Internet protocols under a common application umbrella and is fast becoming the electronic desktop for both business and home.

3.1.2.1 Two- and Three-Tier Architectures

In distributed technology terminology we refer to the Web browser as a client and to the Web server as a server, representing the classic client/server (or two-tier) architecture. The server advertises a service for access by clients. They share a common understanding of the protocols and interface supported by the server. This architecture is often specialized further into a three-tier model (see Figure 12).

- The *client interface* interacts with the user, but usually contains no or minimal application logic.
- The work of the application is conducted in the *business logic* section.
- *Data* is managed and made available in the final tier. This can include database access or other data acquisition such as DFS.

We usually distinguish two- and three-tier by the separation of business logic from data access. We require functionally separate data access servers to generate or provide the required data; otherwise it is no more than a server with associated data.

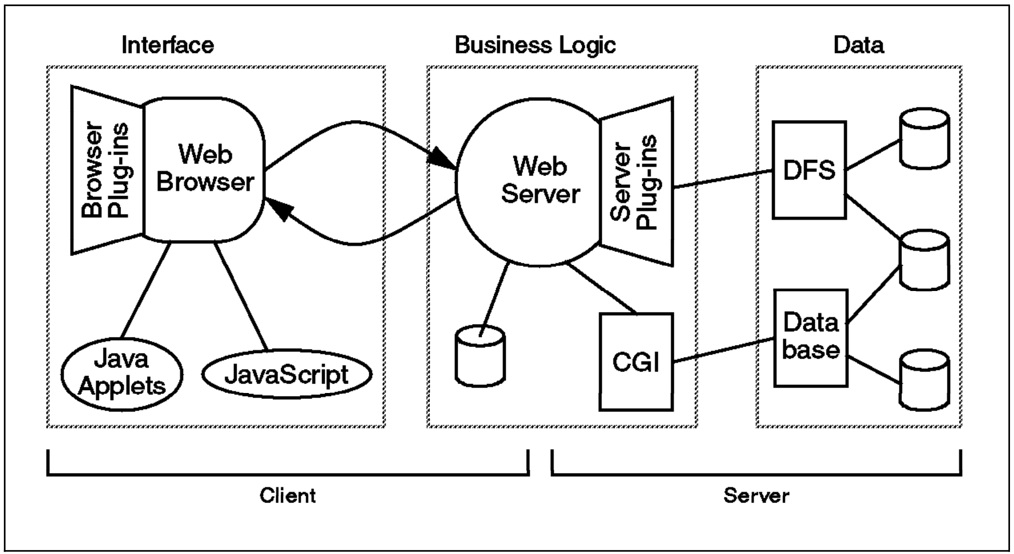


Figure 12. The Three-Tier Model

The traditional client comprises the user interface and part of the business logic, while the server forms the other part of the business logic and the data. The objective for all such models is to separate design and implementation concerns to allow for an extensible and scalable solution.

The Web browser and server architecture can be interpreted in a variety of ways. The simplest is a two-tier (client/server) model with the browser as client and the Web server. The next stage is to assume a separation of data and business logic giving us the three-tier model in Figure 12. We can also further dissect both the browser and server spaces as described in sections 3.1.2.2, “Web Browser” on page 33 and 3.1.2.3, “Web Server” on page 35. This leads to four-tier and even five-tier models. The important issue here is that tiers in a model represent separations in an application model and provide opportunity for component reuse.

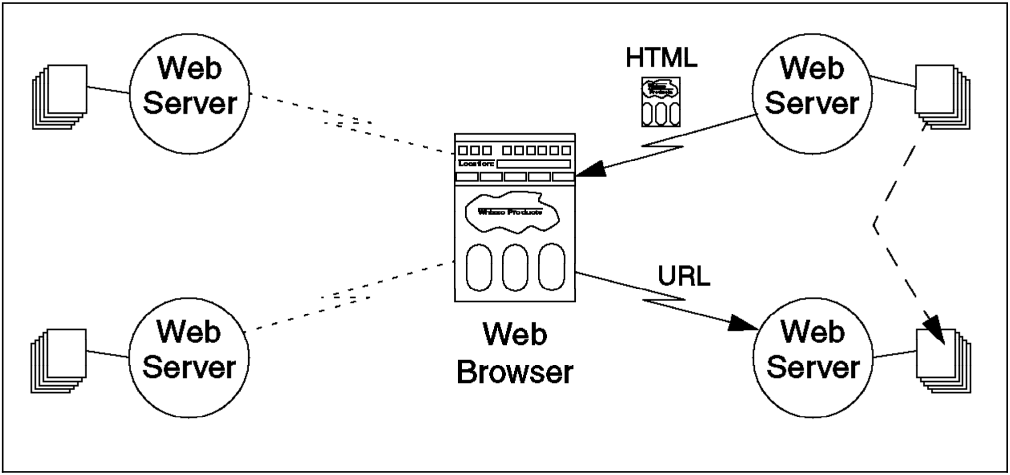


Figure 13. The World Wide Web Architecture

Web interactions are based around the *HyperText Transport Protocol* (HTTP). The protocol manages a logical interaction between a Web browser and a Web server. A Web browser can interact with several Web servers, and in turn each Web server can manage several browser connections. This forms a complex graph of peer-to-peer relationships, but the Web architecture ensures each link is independent. Figure 13 depicts a Web browser accessing documents off four separate Web servers. The *HyperText Markup Language* (HTML) binds content creators to Web browsers and in many ways is independent of a Web server. HTML documents contain Uniform Resource Locators (URLs) that link one document to another. In our example, an HTML document in the top-right corner includes a URL reference to a document available from the Web server in the lower-right corner. Web servers retrieve local documents based on a location reference and distinguish three document types.

- *Plain* HTML documents are the simplest type. They comprise a sequence of text and markup instructions, graphics, audio, or video. The markup instructions are independently interpreted by each Web browser, allowing the display of documents to be independent of their description.
- HTML can also be generated by executable programs at the time the document is referenced. These are known as *CGI* (Common Gateway Interface) scripts. They can be written in any language able to create an executable file that include interpreted scripts like Korn Shell or Perl, and compiled code from C, C++, or any other supported language.
- A new type of document that can be served from the Web server is *Java Applets and classes.* These are segments of Java language code that are downloaded to a Web browser using the HTTP protocol and executed locally on the browser to produce some graphical result. These documents do not use HTML.

The browser/server relationship was once very distinguishing, but with the introduction of downloadable Java Applets, the distinction between browser and server functionality is blurring. However, it is still useful to distinguish the browser/server relationship as the start of a trend toward very lightweight interfaces that can ubiquitously operate across a variety of applications. This was the beginning of a common, heterogeneous application desktop.

3.1.2.2 Web Browser

Unfortunately the simplistic client/server relationship between browser and server is not as clean cut as it once was. The popularity of the Web has seen an enormous growth in applications for Web technology, and there has been pressure to add new features to HTML, HTTP, and both browser and server technologies.

Browser Plug-Ins: The second wave of extensions to the original Web architecture was led by the Web browser plug-in. This is a browser-local software addition that enhances the original functionality of the browser to include new interactive and multimedia capabilities such as sound and 3-D animation. To the user, these extensions are indistinguishable from the standard browser features. Netscape was the pioneer of this technology, and the Netscape Navigator now supports over 130 plug-in modules for the Windows-95 and Windows NT platforms (see for example http://home.netscape.com/comprod/products/navigator/version_2.0/plugins).

Plug-ins live on the local browser disk and are detected by the browser when it starts. When a document is encountered that references an installed plug-in, it is dynamically loaded and given access to all or part of the browser window. The plug-in remains active until the document is closed. More information on creating plug-ins is available at http://home.netscape.com/eng/mozilla/3.0/handbook/plugins/pguide.htm

JavaScript: JavaScript provides a simple way to enhance HTML documents by using a compact, object-oriented scripting language created by Netscape. It adds interactivity at the browser as the inline code is interpreted at runtime. The JavaScript appears in an HTML document within a special script tag. The source is available along with the source for the HTML document. JavaScript has wide acceptance in the Web technology industry and was named JavaScript (originally LiveScript) following endorsement by Sun Microsystems Incorporated. The language is within the grasp of HTML programmers as opposed to Java. They are completely different languages and serve different purposes in the Web environment. JavaScript is for simple, repetitive tasks and glues together components in an HTML document. Java is for Web-aware applications as we will see below.

Java Applets and Classes: Java Applets and classes represent the third wave of the Web. Document content is created at the browser through the execution of a downloaded Java Applet application. Java is a fully featured programming language developed by Sun Microsystems Incorporated originally as a imbeddable language for appliances. Java Applets restrict the full language, ensuring well-behaved applet behavior at the browser, mostly for security reasons. Typical applications include animation and extended user interfaces involving browser-side data checking and manipulation.

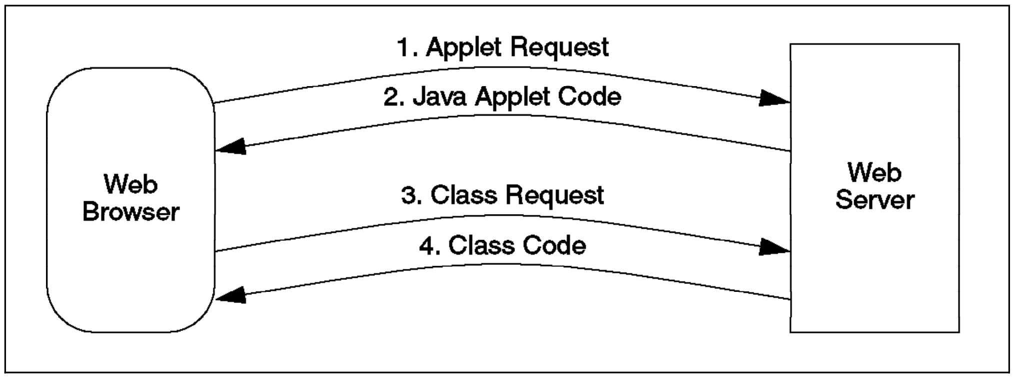


Figure 14. Using a Java Applet

Figure 14 shows the typical sequence of activity between a browser and server while downloading an applet. An initial applet request is sent off to the server when the user selects the applet, much like selecting any other hypertext reference. The Web Server downloads the applet code to the browser, and the browser proceeds to verify that the code is safe. The latest specification of Java (Java 1.1) also allows for authentication of the applet based on public key signatures. The applet may reference Java classes that are local to the browser or are external. External classes must currently be downloaded from the same server from which the applet was obtained. These classes are sent to the browser, and the same code verification procedure takes place as was used with the applet.

Applets provide dynamic document content while off-loading processing requirements from the server to the browser. Since the latest copy of an applet is always available at the browser, software distribution is no longer a problem for either the provider or the user. Java applets are platform independent because each Java-enabled browser implements a copy of the Java Virtual Machine. This is a reference software architecture much like the operating instructions for a hardware chip.

Web Server Proxies: A proxy is a gateway through which a browser accesses Web servers. A proxy can provide three useful functions.

- A *cache* for frequently accessed Web documents. Since all Web browsers can be configured to use the same proxy, this provides a more powerful caching mechanism than any provided by an isolated browser.
- A single point of high-level *logging* for Web access. This can include user access, error information and network traffic.
- A security filter controlling access based on host or domain as well as supporting encrypted communication mechanisms such as SSL.

Proxies are often used as gateways through firewalls. They either sit inside or on the firewall and control the doors that must be opened in the firewall to allow Web traffic through. In addition they can layer extra security on connections by translating insecure Web interactions into secure protocols, if supported (for example, use an SSL connection if available).

3.1.2.3 Web Server

Web server technology has not seen the same modularity approach as can be seen at the browser side. The most fundamental shared facility that all servers provide is CGI scripts. These are executable programs able to generate HTML as requested by a browser. They can be written in any language as long as an executable form is available for invocation by the Web server. Traditionally, scripting languages such as Perl have been popular for CGI integration, and many libraries have been developed to facilitate the management of HTML constructs, such as forms. There is also call for CGI scripts that can access existing business systems including databases and other distributed systems. We will see in a subsequent chapter that DFS Web Secure can be used to facilitate access to DCE application services.

Servers can also support the concept of a server plug-in. These are similar to browser plug-ins, but instead extend the functionality of Web servers. A well-known API provides hooks through which a Web server can take advantage of any extended functionality, for example, augmented authentication and authorization checking or specialized logging.

A distinguishing factor in the Web Server market is the bundling of additional services such as electronic commerce support. 3.2.1, "Netscape's Web Server Products," describes Netscape's Web server products including their Enterprise Server that supports electronic commerce.

3.2 DFS Web Secure Overview and Concept

IBM DFS Web Secure combines the strengths of DCE security, scalability, and efficiency with the ever-growing resource requirements of the World Wide Web. DFS opens the document domain to the global DCE namespace while providing robust security integrated with an existing DCE infrastructure. DFS Web Secure acts as a server plug-in to the Netscape FastTrack and Enterprise Servers. We describe the characteristics of each product and then detail their integration with DFS through DFS Web Secure.

Where you can get IBM DFS Web Secure

IBM DFS Web Secure for AIX is available for download from the URL `http://www.networking.ibm.com/dws/dwsprod.html`. It is also bundled with the IBM DFS Starter Kit for AIX, product number 5765-C35.

DFS Web Secure for Sun Solaris will be available from Transarc Corp. For product information and scheduled availability please visit the Transarc Web site at `http://www.transarc.com`.

3.2.1 Netscape's Web Server Products

Netscape Communications Corporation provide two Web server solutions to the market: Netscape FastTrack Server and Netscape Enterprise Server. Each offers a security-aware server with an emphasis on simple management and cross platform performance. The servers can be enhanced through server plug-ins developed according to the Netscape Server Application Programming Interface (NSAPI).

3.2.1.1 FastTrack Server

The FastTrack Server is the entry-level Web server offered by Netscape. It supports Java and JavaScript as well as extension through the NSAPI. The server relies on simple installation and management through the Netscape Server Manager (NSM) to enable almost anyone without in-depth knowledge to administer a Netscape server. Since NSM is itself simply a Web interface, management can be performed as easily from a remote location as it can locally. The server stores previous configuration setups to enable simple restoration of previous configurations.

NSM leads the administrator through the generation and installation of server credentials based on one of the supported Certificate Authorities. The certificate enables the server to provide SSL-based authentication, encryption and integrity.

Native access control is provided through the creation of users and groups as well as by hosts and domains. These can be used to allow or deny access based on directories, documents, or the entire server. When restricted document is attempted, the browser presents a username/password screen and will only proceed after successful user authentication. This authentication information is reused for all access beyond that level in the document hierarchy. A convenience mechanism allows a standard UNIX password file to be loaded into the Netscape server authentication database.

FastTrack provides logging facilities that provide error and activity logs as well as live activity monitoring. Reports can be generated based on this information.

3.2.1.2 Enterprise Server

The Netscape Enterprise Server is an extension to the FastTrack Server aimed at larger organizations with document management requirements. The MKS (Mortice Kern Systems Inc.) Integrity Engine provides a version control system for Web documents, allowing multiple users to share Web content development. Each individual checks out documents to be modified, then checks them back in ready for modifications to be performed by others.

The *Verity* search engine is also included with the Enterprise Server. It provides full-text search capability for both HTML and plain ASCII documents. A cataloging tool gives users access to Web documents by title, modification time or access frequency.

Logging is also extended to include CGI events and through a log customization language, which can log most transactions undertaken by the server.

3.2.1.3 Netscape Server Application Programming Interface (NSAPI)

Netscape tools embrace extension at both the browser and server sides. Browser plug-ins extend the behavior of the Netscape Navigator to provide new interactive and multimedia facilities not originally available. Similarly Netscape Web servers can be extended to provide additional functionality through the Netscape Server Application Programming Interface (NSAPI).

A Netscape Web server relies on a configuration file, obj.conf, to direct the servicing of a request. This response is broken down into a number of distinguishable steps that are evaluated according to the characteristics of the request (for example URL path name):

- Perform *authorization* checks based on information provided by the user.

- *Translate* the URL provided into a system-dependent name, a redirection request, a mirror site request, or possibly leave it untouched.
- Check the *access* path for the translated name.
- Determine the *type* of the object being accessed. This is provided as a MIME type resolved from the document.
- Select a function to *service* the request.
- *Log* information about the transaction.
- Provide *error* an *recovery* routines.

Implementations of these mechanisms for a transaction type return one of a fixed set of responses to indicate success, failure, and possible continuation. The configuration file is reevaluated after each step of the service resolution. In this way a translation can cause a new set of mechanisms to be triggered for the access check phase.

3.2.2 IBM DFS Web Secure

The Netscape Server API extends the standard server to include functionality not available with the product. This effectively breaks the server layer into two tiers: the generic Web server functionality and back-end components extending functionality into new domains.

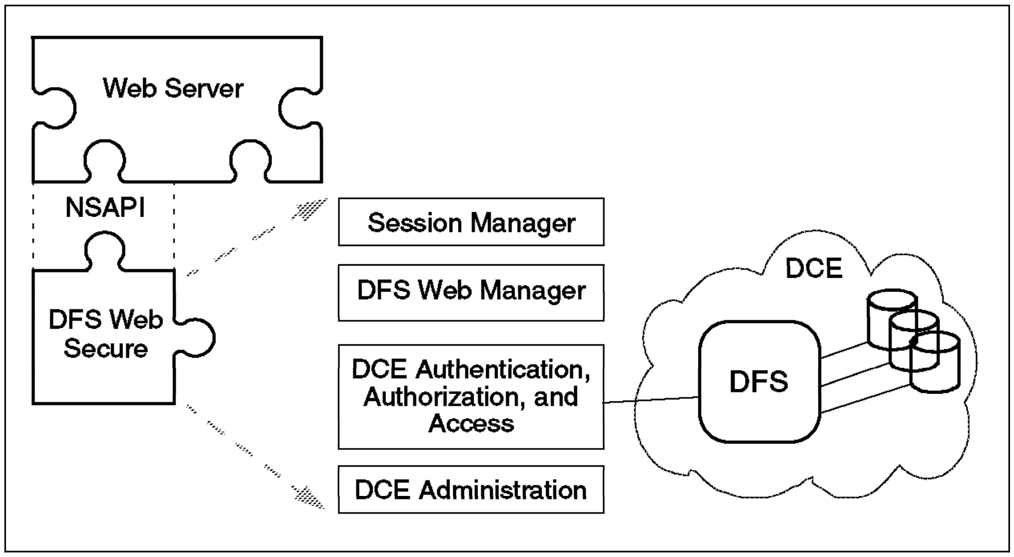


Figure 15. DFS Web Secure as a Web Server Plug-In

One such addition to the Web server is IBM DFS Web Secure. Figure 15 illustrates this integration and will serve as our reference in further descriptions of DFS Web Secure. It extends the Web server into the realm of DFS and DCE.

3.2.2.1 DFS and DCE Integration

IBM DFS Web Secure acts as a gateway between the file-level security of DFS and Netscape Web servers. It uses the NSAPI to extend the existing Netscape authentication and authorization model to include DCE/DFS access security for documents stored within DFS and made available to the Netscape server.

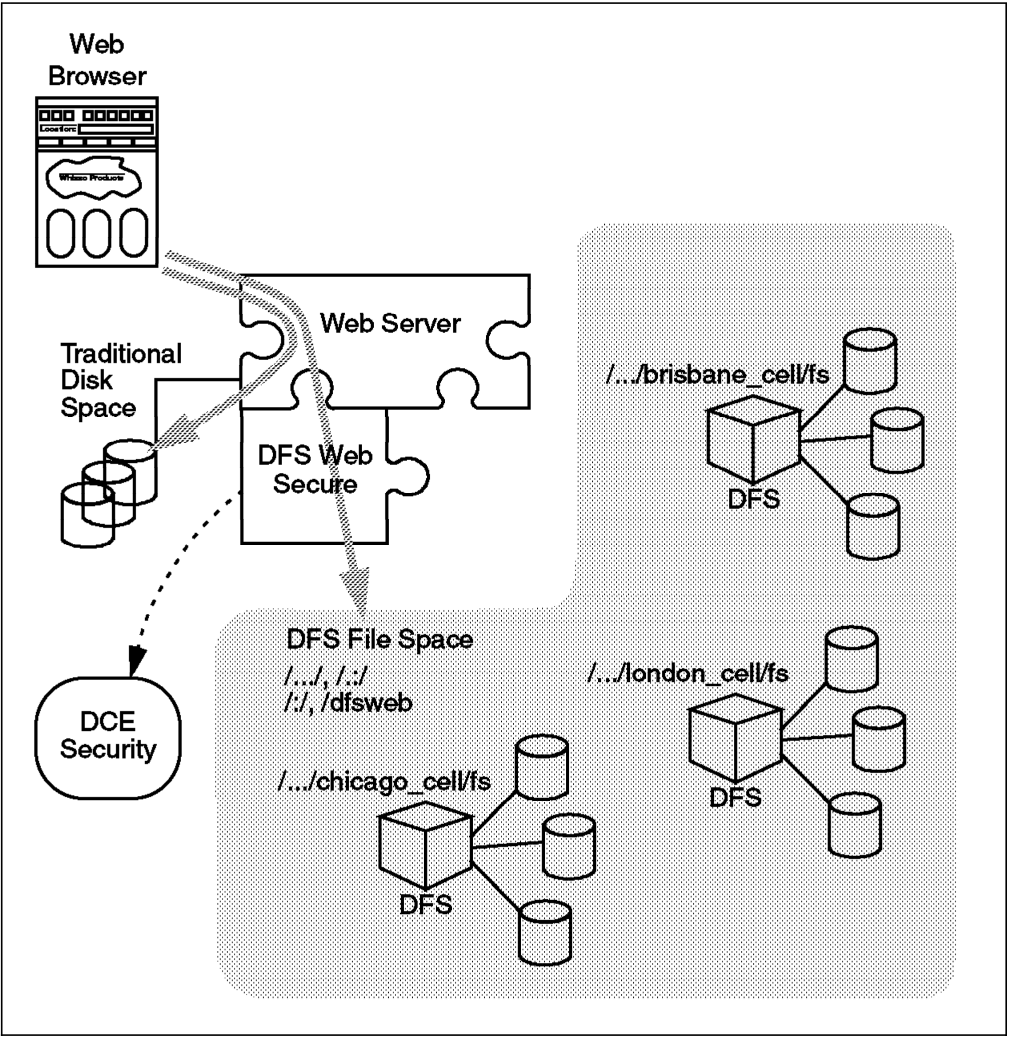


Figure 16. DFS Web Secure and the Web Architecture

The primary goal of DFS Web Secure is the integration of Web technology into an existing DFS infrastructure. Figure 16 shows the gateway in operation. Document requests from a browser are evaluated by the Netscape server using the object configuration file described in 3.2.1.3, "Netscape Server Application Programming Interface (NSAPI)" on page 36. If the translation step results in a document reference in the servers local file system, then DFS Web Secure plays no part in the ensuing document servicing.

By default, DFS Web Secure is responsible for documents with paths in four domains.

- /.../
- /.:/
- /:/
- /dfsweb/

At this point DFS Web Secure becomes responsible for access control and retrieval of these documents. It is important to note that DFS Web Secure maintains the DCE access credentials for documents in the DFS file space. If a document from this space is retrieved and contains a link to a document in the server's local file space, the subsequent document access will not involve DFS Web Secure and thus will not be conducted in the presence of DCE credentials.

The Web server hands over control only when one of the assigned paths are reached following the document name translation step in document request processing. The Web server can access DFS outside the realm of DFS Web Secure, but in that case it looks no different to a local file system and is treated as such with regard to authorization access.

DFS Web Secure is responsible for the authentication, access control, servicing, and logging of document request steps within its designated namespaces. The DCE security services are contacted to authenticate the user, and authorization access is checked subsequently by DFS based on the user's credentials and the requested document. In fact, DFS Web Secure acts as a proxy for the user, with the user's DCE credentials. This is also true for CGI programs that are to be run off the DFS file space. They inherit the user's DCE credentials from DFS Web Secure and can therefore access DCE server applications anywhere in the cell. DFS Web Secure provides extended logging facilities to those available natively under the Netscape server. These include failed DCE logins, URL access counts, user page-access counts, and user traffic volume. Given the separation of document retrieval steps, it is possible to extend this logging and subsequent threshold notification (see section 3.2.2.3, "DFS Web Manager" on page 40) to pages outside the DFS file space. The user is still required to provide a valid DCE user name and password in order to get DCE credentials, but only the authentication and logging steps are applied. Servicing is done by the default Netscape routine. This illustrates the flexibility in the NSAPI approach to customization.

As described in section 1.6, "The DCE Distributed File System (DFS)" on page 19, DFS offers many advantages over traditional file systems. Several are of particular relevance to the World Wide Web document space:

- DFS's replication and caching models provide both high availability and performance. Both are crucial issues for resource-hungry Web servers.
- DCE/DFS provides a scalable, global file space that encourages information sharing and interdomain access. Files do not need to copied to the Web server to make them available to the Web. Without DFS, Web servers are often limited to documents on their local disks.
- Web servers often straddle the boundary between intranet and Internet, making their secure operation critical. DCE and DFS provide a comprehensive security infrastructure including authorization and access control.

It is important to appreciate the separation between the Web server and DFS Web Secure responsibilities. The object configuration file plays the pivotal role in defining this boundary, and the NSAPI provides the mechanism for linking the components together. DFS Web Secure may be one of many plug-ins applied to a server.

Chapter 5, "Running the Business With DFS Web Secure" on page 63, provides further information on applying DFS Web Secure to a business using DFS.

3.2.2.2 DCE Administration

DFS Web Secure is more than a gateway to secured DFS documents. It also provides a Web-based administration tool for DCE management. It currently provides forms for administering four components of DCE.

- *User* management covers adding new users, modifying the attributes of existing users, deleting users, password modification, and listing users within the cell. For user addition and modification, both simple and advanced feature forms are provided.
- *Groups* can be constructed, modified, deleted, or listed. During creation, the administrators can initialize group membership.
- A similar set of mechanisms is available for the construction, modification, deletion, and listing of *organizations*.
- *Access Control List* (ACL) management simplifies permission specification through a point-and-click interface. Available access permissions are graphically presented, and the administrator selects those that are applicable. New entries can be added or existing ones modified.

The administrator is authenticated and authorization checked before allowing remote DCE administration through the DFS Web Secure infrastructure. This same security check can be applied to CGI scripts within the DFS Web Secure domain, and these application services can operate as DCE clients complete with credentials (this technique is further discussed in Chapter 7, “Alternate Methods for Using DCE on the Web” on page 133).

Section 5.1, “Managing User Access” on page 64, provides more information on DCE administration using DFS Web Secure.

3.2.2.3 DFS Web Manager

The DFS Web Manager extends the reporting and monitoring function of Netscape servers through the NSAPI hooks described in section 3.2.1.3, “Netscape Server Application Programming Interface (NSAPI)” on page 36. It maintains a set of four monitoring databases are managed through the Web-based administration tool. These cover:

- Failed DCE authentications
- Document access counts
- Page access counts by authenticated user
- Traffic volume (in bytes) by authenticated user

The databases can be viewed and either reset manually, or a reset interval can be specified. The more interesting use of the databases is through the automated threshold functionality that the manager supports. Simple, logical threshold rules are applied to a database requesting an activity be performed when that threshold rule evaluates to true, for example, when `cell_admin` failed logins exceed three attempts. Parameters to the trigger rule isolate it to a particular user or document. The action associated with the successful trigger rule can include:

- Reset the counter matching the monitored resource.
- Execute a program (under the user id and group id of the Web server).
- Send electronic mail to a user
- No action is taken.

Successful threshold triggers can also be logged to a file for later analysis.

A more detailed description of the DFS Web Secure thresholds is given in section 5.3, “Monitoring and Thresholds in DFS Web Manager” on page 72.

Chapter 4. DFS Web Secure Installation and Customizing

This chapter describes how to set up a Web server along with DFS Web Secure. The key installation steps are covered along with a number of things that need to be planned or considered in advance of installation. Although it explains the major installation steps, this chapter is not intended as a substitute for the full product documentation.

For availability information on DFS Web Secure, please refer to 3.2, “DFS Web Secure Overview and Concept” on page 35.

4.1 Installation

There are three main steps: 1) Netscape server installation, 2) setting up DFS access, and then 3) DFS Web Secure installation. The order in which the first two steps are carried out is not important, but both must be completed before DFS Web Secure can be set up. The solid boxes in Figure 17 represent the three key installation steps.

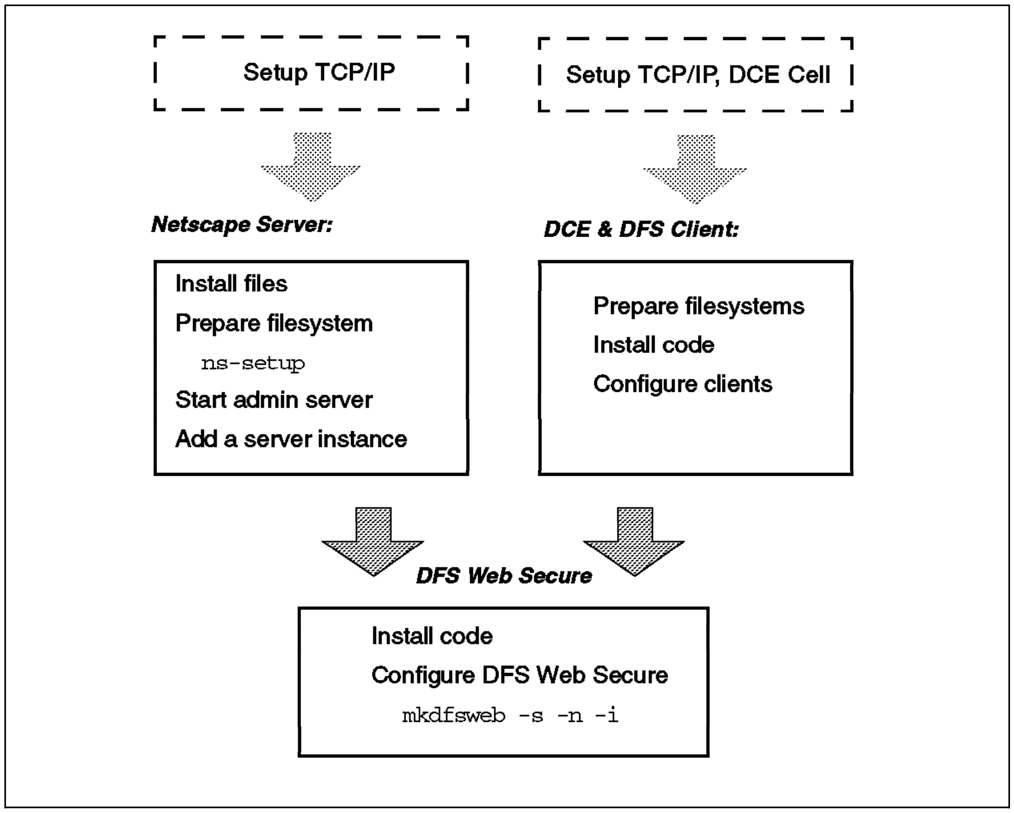


Figure 17. Key Steps in DFS Web Secure Installation

The following sections describe each of the three steps.

4.1.1 Netscape Web Server

DFS Web Secure in its first release can be used with either the Netscape FastTrack Server or the Netscape Enterprise Server. Both Netscape server programs require systems with at least 32 MB of RAM. In addition the following points should be considered before installing the Netscape server program:

1. TCP/IP must to be running on the system with Domain Name Server (DNS) or an equivalent name-resolution capability configured.

2. AIX Version 4.1 systems need the AIX Shared Library Hookable Symbols/6000 runtime (`slhs.rte.obj`) environment installed. This comes supplied with the Netscape server software when it is bought from IBM. This software does not need to be installed on AIX 4.2 systems.

   Alternatively, if you have obtained the Netscape server software from Netscape and you are running an AIX 4.1 system, then you can download this software from IBM by anonymous FTP from `ftp.software.ibm.com`.

3. A Web server user account should be set up. For security reasons, it is not recommended that the Netscape server run under the `root` user. By default, the server will opt to run as the user `nobody`, which would allow unauthorized access to the DCE credential files used by DFS Web Secure when it is running. If the Web server ran as `nobody`, any user could potentially FTP to the server and acquire and use the unprotected credential files. We recommend setting up a separate account for the Web server, for example `www` or `websvr`.

4. Consider the file space to store the Netscape server code. If you obtained the Netscape server code from IBM, then the files will be placed by the `installp` command in a directory beneath /usr/lpp/netscapeServer. The name of the directory depends on the server type. FastTrack Server requires 52 MB and Enterprise Server approximately 100 MB of disk space in the current releases at the time this book was written.

5. Consider the file space for the server root directory. This is the area where the server files are placed when the Netscape server code is installed by the ns-setup program. If you have obtained the server code from IBM, then this is like a second-stage installation.

   You may consider creating a separate filesystem for this server-root directory. For the FastTrack Server Version 2.01, which was used for the purpose of writing this book, we created a minimal filesystem of 60 MB and mounted it over the default server root directory /usr/ns-home. This is also likely to be the directory under which you will choose to store the HTML documents that will be served by the HTTP server instance that is being set up. It is possible to set up several HTTP server instances on one system.

Once the preparation steps above have been completed the Netscape server installation can begin, made up of the following steps:

1. Install files onto system—If the software is obtained from IBM, then it will be installed onto the system by the standard AIX installation command `installp` (optionally driven via `smit install`). Otherwise, if it is obtained from Netscape, you need to unpack the tar file loaded onto the system (it will have arrived either on CD-ROM or through network download).

2. Run `ns-setup` —This setup program needs to be run from the command line. It will be located under the directory where the server code was installed. If the software was obtained from IBM, this will be a directory beneath

/usr/lpp/netscapeServer, depending on the actual Netscape server you are installing. After accepting the license terms, you will be prompted to do the following:

a. Choose the server root directory (the default is `/usr/ns-home`). This is where the server files will be placed along with configuration and log files.

b. Enter the full hostname of your server, for example `ev1.itsc.austin.ibm.com`.

c. Enter the AIX user name under which the Web server is to run. In the preparation steps, we recommended setting up a separate account, in our example `www`.

d. Enter the AIX user name under which the Netscape Administration Server is to run. If you are going to have any Web servers on the standard HTTP port 80, then this has to be `root` (since only the root user can manipulate lower port numbers).

e. Choose the TCP port number for the Administration Server. This should be between 1025 and 65535, and it is best to avoid the ports 1080 and 8080, which are normally used for socks and proxy servers. The `ns-setup` tool generates a suitable random port number that you can accept or change. In any case, it is vital that you remember this number.

f. Choose an administration interface login name and password. This is used to carry out administration of the server through the Administration Server, and it does not have to correspond to an AIX user account.

g. Declare the hosts granted access to the Administration Server. Allowing remote access is useful, and you can restrict it by specifying a limited set of hostnames or IP addresses. Wildcards can be used to allow ranges of systems or addresses.

3. Start the Administration Server—This may happen automatically for the first server installation. If it is not already running, the Administration Server can be started with the command `start-admin`, which is located in the server home directory (default is /usr/ns-home). Then, start a frames- and JavaScript-enabled Web browser on any machine, and connect to the Administration Server using the server's hostname and port number as the URL (with the port number you have remembered from `ns-setup`). In our example, this had to be `http://ev1.itsc.austin.ibm.com:12629`.

   The architecture for administration of the Netscape server is shown in Figure 18 on page 46.

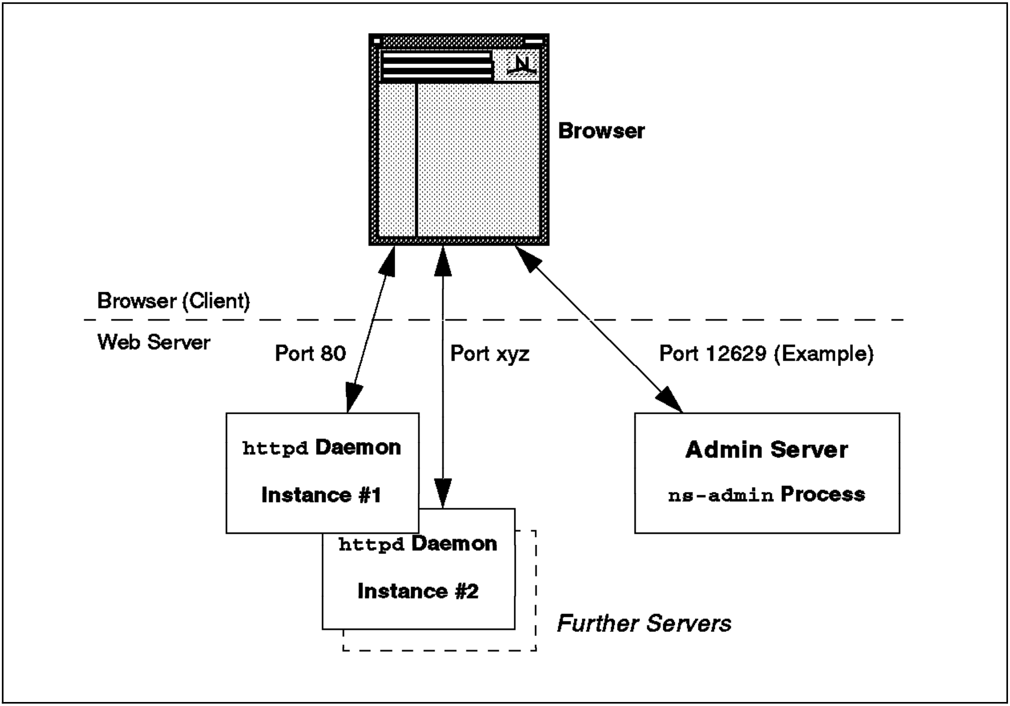


Figure 18. Netscape Server Architecture

From the Administration Server interface through the browser, it is possible to configure, start and stop the Netscape server instance(s) on your system. At least one server instance must be configured. Complete documentation covering customisation of the Netscape server is available through built-in online help (*Administrator's Guide* and *Programmer's Guide*), or over the Web from Netscape at URL
`www.netscape.com/comprod/server_central/support/fasttrack_man/index.html`

An example of the screen presented to you when you direct your browser to the Administration Server port number and supply the administration user name and password is shown in Figure 19 on page 47.

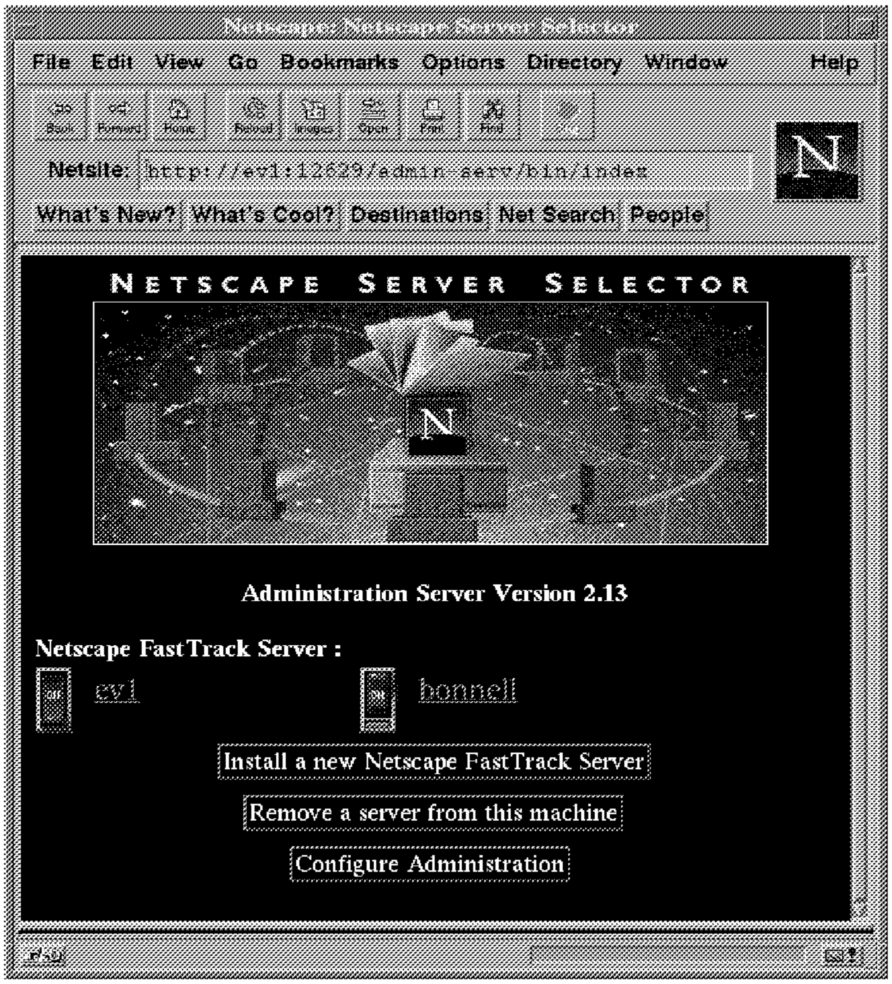

Netscape: Netscape Server Selector
File Edit View Go Bookmarks Options Directory Window Help
Netsite: http://ev1:12629/admin-serv/bin/index
What's New? What's Cool? Destinations Net Search People
NETSCAPE SERVER SELECTOR
Administration Server Version 2.13
Netscape FastTrack Server :
ev1
bonnell
Install a new Netscape FastTrack Server
Remove a server from this machine
Configure Administration

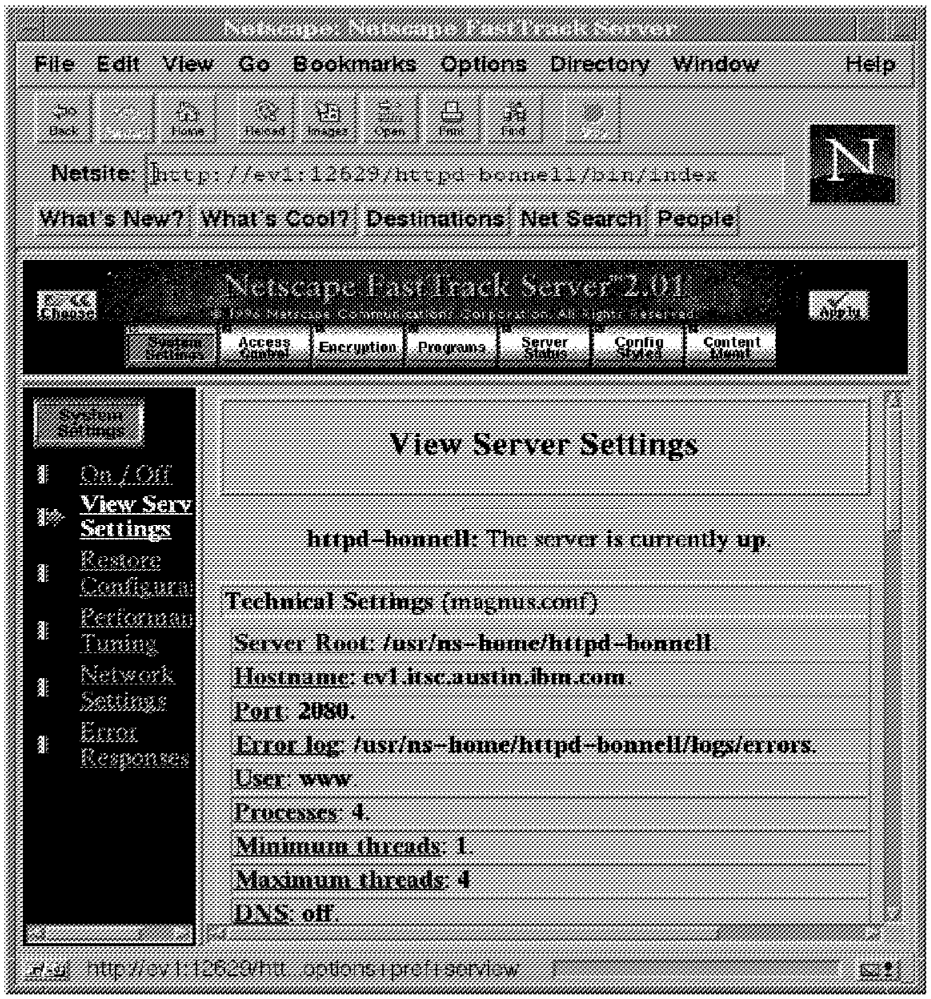
Figure 19. Netscape Server Selector Screen

To prepare for DFS Web Secure, it is important to cover a couple of parameters within your Web server configuration.

- The Web server settings for `MinThreads` and `MaxThreads` would fit the model of DFS Web Secure better if they were set to 1. This is because DFS Web Secure serializes some of its processing. For the version of the Netscape server that was used in our testing, FastTrack Version 2.01, `MaxThreads` has to be set greater than 1; so the values 1 and 4 are recommended in place of the defaults 4 and 32 for these parameters.

  To view their current values, click on the name of your server instance on the Server Selector screen and then choose **System Settings** from the Tab frame (across the top of the screen). Click on **View Server Settings** from the Category frame on the left side of the screen. These frames and the View Server Settings screen can be seen in Figure 20 on page 48.

  By clicking on either the parameter names themselves or on **Performance Tuning** from the Category frame (left of screen), the Performance Tuning screen can be reached, and the values for **Minimum threads** and **Maximum threads** can be reduced.


Netscape: Netscape FastTrack Server
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Netsite: http://ev1:12629/httpd-bonnell/bin/index
What's New? What's Cool? Destinations Net Search People
Netscape FastTrack Server 2.01
Choose
Apply
System Settings
Access Control
Encryption
Programs
Server Status
Config Styles
Content Mgmt
On / Off
View Server Settings
Restore Configuration
Performance Tuning
Network Settings
Error Responses
http://ev1:12629/htt...options+pref+serview

View Server Settings

httpd-bonnell: The server is currently **up**.

Technical Settings (magnus.conf)

Server Root: **/usr/ns-home/httpd-bonnell**
Hostname: **ev1.itsc.austin.ibm.com**.
Port: **2080**.
Error log: **/usr/ns-home/httpd-bonnell/logs/errors**.
User: **www**.
Processes: **4**.
Minimum threads: **1**.
Maximum threads: **4**
DNS: **off**.

Figure 20. View Netscape Server Settings Screen

4.1.2 DFS Client

For DFS Web Secure to access the DFS file space, it is necessary to have the DFS client installed and configured on the same machine as the Web server and DFS Web Secure. It is not necessary to have any DFS server functions installed on this machine (but there would not be a problem if you do).

Before installing the DFS client, you need to install and configure the DCE client. It is assumed that the DCE cell is already configured and running on the network that can be reached from the system that is to run DFS Web Secure. DFS server functions are also required within the cell. Instructions covering how to install the DCE and DFS server functions are covered extensively in the IBM manual *Directory and Security Server for AIX Up and Running!*, SC23-1797, or in the Redbook *Administering IBM DCE & DFS Version 2.1 for AIX (and OS/2 Clients)*, SG24-4714.

The following points will help you plan and prepare for the installation of the DCE and DFS clients that are shipped with AIX Version 4:

1. TCP/IP networking must be running with name resolution and routing working between systems in the DCE cell.

2. Create a separate filesystem /var/dce for DCE to store its working files. It aids administration and increases availability to keep this as a separate filesystem from the /var filesystem in AIX. If you chose to add such a filesystem, it must be created before the DCE client code is installed on the system from the AIX Version 4 media. The suggested size for a DCE client is between 12 and 16 MB.

   The following commands create and mount a suitable filesystem of 16 MB:

   ```
   # crfs -v jfs -g'rootvg' -a size='32000' -m'/var/dce' -A'yes' -p'rw'
   # mount /var/dce
   ```

3. File space for the DFS client cache in /var/dce/adm/dfs/cache —You may consider creating a separate filesystem for the DFS cache files since the cache size may need to be adjusted over time. If you chose this option, this filesystem must be created before the DCE/DFS client code is installed on the system from the AIX Version 4 media. The size of the cache filesystem is an important factor in the performance of DFS Web Secure.

   The following commands create and mount a suitable cache filesystem of size 12 MB:

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

   The default cache size is actually 10 MB, but it is vital that the filesystem containing the cache has enough space for the cache itself plus 15 percent.

4. Install DCE/DFS client files onto system from the AIX Version 4 media using `installp` from the command line or via `smit installp`.

5. The DCE cell name and the host names of the Security and CDS servers within the DCE cell are needed to configure the DCE client. Configuration can be carried out with the command:

   ```
   # mkdce -n <cell_name> -s <security_server_name> sec_cl cds_cl dts_local
   ```

   or via `smit mkdceclient`. The DCE Time service client, `dts_local`, included in this command is optional.

6. Finally, the DFS client can be installed with the following command:

   ```
   # mkdfs dfs_cl
   ```

   or via `smit mkdceclient`

In order to check that the DCE and DFS clients have installed successfully, it is advisable to run through a series of commands to check that your system does have access to the DCE cell. The following are some useful hints:

- Run the commands `lsdce` and `lsdfs` to show the DCE/DFS components configured on your system. Look for the entries `cds_cl`, `rpc`, `sec_cl` and `dfs_cl`.
- Run a `dce_login cell_admin` (or any other valid DCE user) to authenticate with DCE. Enter the password and verify proper function.
- Run the `klist` command to list the DCE credentials, which should be those of the principal you have chosen for login in the previous step.
- Run `rpccp show mapping` to display the RPC Endpoint map.

- Run the dcecp -c cell show command to show some details of the DCE cell configuration, including a list of hosts within the cell.
- Do a df /... to show the status of the DFS filesystem on the local system. The response should look like:

```
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
DFS             18000000  18000000    0%        0     0% /...
```

- Do a cd /.../<cell_name>/fs or cd /: which will allow you to enter the DFS filespace in your local cell and browse at least the root directory from the AIX command prompt.

After you have successfully installed and configured DCE and DFS client code on the Web server, you are ready to install DFS Web Secure, assuming the Netscape server has already been installed and configured.

4.1.3 DFS Web Secure

Apart from installing the Netscape Web server and the DFS client on your system, there are just a few requirements to check before commencing the installation of DFS Web Secure. They are:

1. An additional 5 MB of hard disk space needs to be available in /usr. The installp will automatically increase the size of the filesystem if required.
2. The AIX DCE/DFS client code needs to be updated to the PTF Set 14 level or higher.
3. A Web browser capable of displaying frames should be available locally. A browser that supports JavaScript is preferable, such as Netscape Navigator 3.0. This will be used for the administration of DFS Web Secure (and of the Netscape server). It could also be used across the network and so the browser does not necessarily have to reside on the same system as DFS Web Secure, but it might be useful for convenience.
4. Create an anonymous DCE principal and account with password anonymous. This is an account that you could publicize and use to allow people not defined in your DCE environment to access areas within your DFS filespace. A different ID and password combination can be chosen for this anonymous account. DFS Web Secure always requires that users log into DCE. If a browsing user leaves the user name and password fields empty in the DCE login window, then access to DFS will not be allowed even if the DFS filespace is open for some access by unauthenticated users.

4.1.3.1 DFS Web Secure Product Installation

Now make the DFS Web Secure install image fileset available on your system. DFS Web Secure can be installed from the command line with the following command:

```
# installp -acv -d <image directory> dce.dfsweb
```

Alternatively, to install DFS Web Secure using smit proceed as follows:

```
# smit install
-> Install and Update Software
-> Install/Update Selectable Software (Custom Install)
-> Install Software Products at Latest Level OR
(On AIX 4.2 select -> Install/Update from Latest Available Software)
-> Install New Software Products at Latest Level
```

Enter the appropriate directory for the `input` device/directory

Enter `dce.dfsweb` in the Software to Install field and change any of the other defaults as required. Then press **Enter** to start installation.

To verify that DFS Web Secure has installed successfully, enter the following command:

```
# lslpp -L | grep dce.dfsweb
```

and look for the following output:

```
dce.dfsweb.rte              1.1.1.0    C    DFS Web Secure
```

4.1.3.2 DFS Web Secure Basic Configuration

With information from the configuration of your Netscape server, you can configure DFS Web Secure with the following single command:

```
# mkdfsweb -n <netscape_server_homedir> -s <netscape_server_name>
           -i <netscape_server_username> -a anonymous -p anonymous
```

where <netscape_server_homedir> is the home directory where your Netscape server is installed. The default is /usr/ns-home, in which case specifying `-n` is optional. The <netscape_server_name> is the name of the Netscape server, and <netscape_server_username> is the AIX user ID under which the Netscape server runs.

Note:

DFS Web Secure installation with `mkdfsweb` makes changes to the Netscape server configuration files (`magnus.conf` and `obj.conf`). If you are concurrently using the Netscape Administration Server interface through a browser then you will need to reload the changed configuration files. This can be done through the **Apply** button at the top of the screen, followed by **Load Configuration Files**.

It is also important to notice that DFS Web Secure must be unconfigured with `rmdfsweb` prior to uninstalling it. This is to ensure that DFS Web Secure information is removed from the Netscape server configuration files.

4.2 Customizing

As you can see from 4.1.3, "DFS Web Secure" on page 50, the installation and setup of DFS Web Secure is very straightforward, with just two major steps. Before opening up your initial DFS Web Secure installation to the whole world, there are some aspects of customization that you should consider—even though DFS Web Secure is capable of (and may in fact be) running already.

This section covers some more advanced aspects of customizing and administration.

4.2.1 The Quick Start

Your DFS Web Secure installation is already up and running when the `mkdfsweb` command successfully completes—provided you have a Netscape server and the DFS client already running on your system. The following aspects of customization and administration are important and should be noted before opening up your DFS Web Secure site for business:

1. Set up Access Controls for the `anonymous` user account. Following the procedure outlined in 4.1, "Installation" on page 43 will produce a Web site where any browser user can authenticate with the DCE registry and stand at the entrance to the DFS filespace of your organization. Clearly, it is important to have ACLs in place at the top level of your DFS space that will restrict the filesets into which the `anonymous` user can go. If you have a file structure with reasonably open permissions for the `other_obj` type, then the `anonymous` user may be free to look around more of your site than is desired.

   For information on ACLs and how to secure files and directories in DFS using them, please see5.4, "DFS Access Control" on page 78.

2. Directory viewing is allowed with the default installation of the Netscape server. This is where a browser user can reference the URL of a directory entry in the document hierarchy that he/she is browsing (instead of an HTML document) and be presented with a list of the directory contents. The user can then move between directories and view other directory contents by clicking on directory entries in the list presented on the browser.

   In the DFS filespace the user's DCE account would have to have read permission on a directory in order to see the list of directory contents. It is possible to prevent the Netscape server from offering directory listings to a client by switching off **Directory Indexing**. This can be done through the Administration Server interface by clicking **Content Mgmt** in the Tab Frame (top of screen) and then clicking **Document Preferences** in the Category Frame (left side of screen). In the *Directory Indexing* section of the screen there is checkbox labeled *None* that allows you to switch off the ability for browsers to see directory information.

   If you want to allow directory viewing for certain users on parts of your filespace, then you have to allow it for all users because this panel configures it for your whole HTTP server instance. With DFS ACLs, however, you can easily allow only specific users (or groups of users) to see the contents of any directory through this facility. Granting `r` permission allows a user to see directory contents, but a user with just `x` permission cannot see the directory contents (but can still access specific files within the directory and pass through the directory to subdirectories).

3. Enlarge the DFS Client Cache size for better performance. In 4.1.2, "DFS Client" on page 48, instructions are given to prepare a filesystem to hold the default DFS cache (of size 10 MB). If you are expecting to serve significant amounts of data stored in DFS, then you can expand the size of the DFS client cache to improve the performance that will be perceived by browser clients. This will be particularly true where large documents are referenced repeatedly.

   To change the cache size permanently requires you to increase the size of the filesystem containing the cache and alter the CacheInfo file. These steps can be performed as follows:

   a. Run the command `# chfs -a size=<new_size> /var/dce/adm/dfs/cache` where `<new_size>` is the desired filesystem size in 512-byte blocks.

Remember that this should be 15 percent larger than you want the cache size to become. Alternately, you can use SMIT to run this command through `smit chfs`.

b. Edit the file /opt/dcelocal/etc/CacheInfo and set the number of 1 KB blocks to the desired value. For example changing the file to

```
/...:/opt/dcelocal/var/adm/dfs/cache:40000
```

will change the cache size to 40000 KB (40000 blocks) at the next DFS restart.

DFS can be stopped through `smit dce` using the dialog for Stop DCE/DFS Daemons or alternately by entering the following command:

```
/etc/dce/dce.clean dfs
```

The system needs to be rebooted before DFS can be restarted.

4. In our testing we noticed that DFS Web Secure's Session Manager process (started automatically by the Netscape server instance once DFS Web Secure is configured) is not always stopped when the server instance is turned off. If this happens and the processes persist through a subsequent start of the server instance, then you may need to end them with the `kill` command. The Session Manager processes associated with the server instance name can be identified in the process table for the Web server user (shown by running the command `ps -fu`, for example `ps -fu www`) as the entries containing /usr/bin/session_manager. An early update of DFS Web Secure that will be available before this book is published should resolve this issue.

4.2.2 The Sophisticated Setup

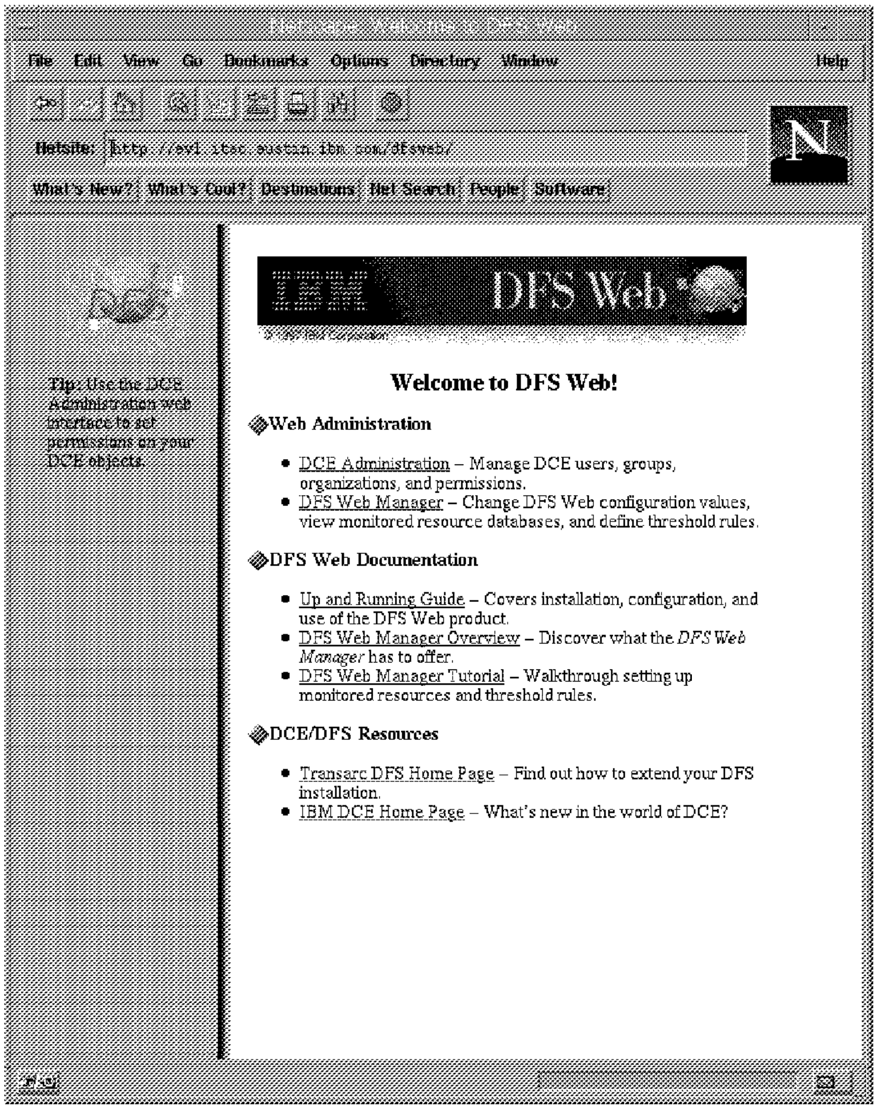
DFS Web Secure can be set up with a more sophisticated configuration, and it also comes with two administration interfaces that run within any Web browser capable of supporting frames. The default DFS Web Secure setup described so far can be customized with the DFS Web Manager interface.

The other browser interface supplied, DCE Administrator, allows you to perform:

- DCE user account administration
- DCE group administration
- Organization administration
- ACL management (in other words defining access permissions) for DFS file objects

Both interfaces can be accessed from the DFS Web Welcome screen shown in Figure 21 on page 54, which is reached by directing your browser at the URL `dfsweb` (relative to your server's home URL). In full, this would be `http://<full_server_name>:<server_port>/dfsweb/`

Sophisticated configuration of DFS Web Secure is mainly concerned with overriding the default options for the monitoring that DFS Web Secure performs on its resources. It also allows you to set up a trigger value for the Failed Login Counter and specify an action to be taken when that trigger value is reached. Monitoring does take place automatically with the default configuration of DFS Web Secure, but the default trigger action for failed logins is no action.


Netscape: Welcome to DFS Web
File Edit View Go Bookmarks Options Directory Window Help
Netsite: http://ev1.itsc.austin.ibm.com/dfsweb/
What's New? What's Cool? Destinations Net Search People Software
DFS Web

Tip: Use the DCE Administration web interface to set permissions on your DCE objects.

Welcome to DFS Web!

Web Administration

- DCE Administration – Manage DCE users, groups, organizations, and permissions.
- DFS Web Manager – Change DFS Web configuration values, view monitored resource databases, and define threshold rules.

DFS Web Documentation

- Up and Running Guide – Covers installation, configuration, and use of the DFS Web product.
- DFS Web Manager Overview – Discover what the *DFS Web Manager* has to offer.
- DFS Web Manager Tutorial – Walkthrough setting up monitored resources and threshold rules.

DCE/DFS Resources

- Transarc DFS Home Page – Find out how to extend your DFS installation.
- IBM DCE Home Page – What's new in the world of DCE?

Figure 21. DFS Web Secure Welcome Screen

Installing DFS Web Secure with mkdfsweb -v will take you through the detailed configuration options. They are summarized in the following command line help, which can be seen by running the command mkdfsweb -?.

```
# mkdfsweb -?
Usage: /usr/bin/mkdfsweb [-n netscape_dir ] [-s netscape_server ] -i userid
           [ -a anonymousid ]   [ -p anonymouspw ] [ -f LS_filename ]
           [ -r reset_int ] [ -e DCE_ID ] [ -u URLDB ]
           [ -d PageCountDB ] [ -b ByteCountDB ] [ -x FailedLoginDB ]
           [ -c action ] [ -g trigger_value ] [-h threshold_match ]
           [ -l logfile ] [ -o actionparm ]  [ -v ]  | -?
 Options
     -n netscapedir      Identifies the Netscape server root directory.
                         /usr/ns-home is the default.
     -s netscape_server  Identifies the name of the Netscape server to
                         configure.
     -i userid           Identifies operating system user ID for the
                         Netscape server to run under.
     -a anonymousid      Identifies the DCE account ID to user for anonymous
```

```
                      access.  'anonymous' is the default.
     -p anonymouspw   Identifies password for the anonymous id.
                      'anonymous' is the default.
```

The example above only shows the first few lines of the usage message; the remaining command line options are the key monitoring options which mkdfsweb -v will prompt you for:

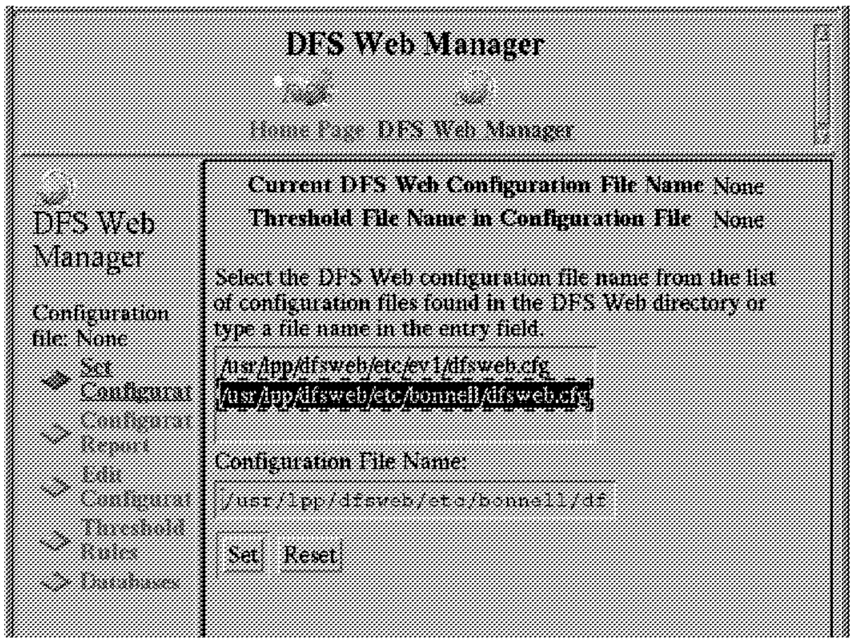
-f LS_filename	Identifies the full pathname of the DFS Web Manager Load Summary File. If the pathname is *none*, the startup records of the DFS Web Manager are not recorded when the server is started. The default is /usr/lpp/dfsweb/etc/<netscape_server>/loadsummary.
-r reset_int	Identifies the interval in seconds between the time a counter for a monitored resource database record was last updated and when it is reset to zero. Valid values range from 0 to 2678400 (31 days). The default is 0, which indicates that no automatic resetting should be performed.
-e DCE_ID	Identifies the DCE user ID permitted to access the DFS Web Manager administration Web pages. The cell_admin user ID will always be able to access these Web pages. The default is 'cell_admin'.
-u URLDB	Identifies the fully qualified file name of the URL Counter Database. If *none*, the URL Counter resource is not monitored. The default is /usr/lpp/dfsweb/etc/<netscape_server>/dburl.
-d PageCountDB	Identifies the fully qualified file name of the Page Counter Database. If *none*, the Page Counter resource is not monitored. The default is /usr/lpp/dfsweb/etc/<netscape_server>/dbpage.
-b ByteCountDB	Identifies the fully qualified file name of the Byte Counter Database. If the file name is *none*, the Byte Counter resource is not monitored. The default is /usr/lpp/dfsweb/etc/<netscape_server>/dbbyte.
-x FailedLoginDB	Identifies the fully qualified file name of the Failed Login Counter Database. If the file name is *none*, the Failed Login Counter resource is not monitored. The default is /usr/lpp/dfsweb/etc/<netscape_server>/dbfailed.
-c action	Identifies what action to take when a user reaches a specified number of failed logins. Valid actions are *none* (do nothing), *clear* (reset the counter), *program* (launch a program), and *note* (send e-mail note). The default is *none*.
-g trigger_value	Identifies the number of failed logins that must occur for a user before the action specified in the -c option is triggered. The default is 10.
-h threshold_match	Indicates whether only the first matching threshold rule should be triggered or if all matching threshold rules should be triggered when a valid Web server transaction occurs. Valid values are *all* or *first* (default).

-l logfile	Identifies the fully qualified file name to store records of threshold rules that were triggered. If *none*, triggered threshold rules are not recorded in a file. The default is `/usr/lpp/dfsweb/etc/<servername>/triggerlog`.
-o actionparm	Identifies the parameters for the action specified in the `-c` option. If the action is *program*, the `actionparm` must be the fully qualified program name and any parameters associated with the program. If the action is *note*, the `actionparm` must be the e-mail address and any additional text to send with the note. The default is *nothing*.

The use of DFS Web Secure monitoring and thresholds is covered in some detail in Chapter 5, "Running the Business With DFS Web Secure" on page 63. All of the parameters shown above can be configured through the DFS Web Manager interface, which runs in a browser as shown in Figure 22.

Note on File Permissions

In the first product releases of DFS Web Secure, we found that it was not able to write into a number of its database files due to insufficient file permissions. If `mkdfsweb` is run as root**,** then it is likely that the Web server, running under another user ID, will not have write permission in the directory /usr/lpp/dfsweb/etc/<web_server_name>. It is useful to check the Netscape server's error log file /usr/ns-home/httpd-<server_name>/logs/errors, where clues to this problem can be seen.



Current DFS Web Configuration File Name None
Threshold File Name in Configuration File None

Configuration file: None

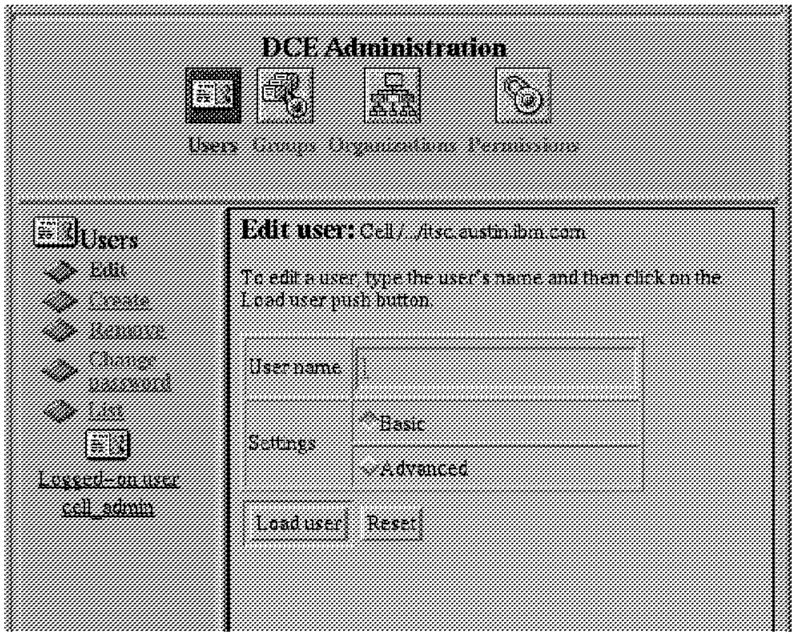
Select the DFS Web configuration file name from the list of configuration files found in the DFS Web directory or type a file name in the entry field.

/usr/lpp/dfsweb/etc/ev1/dfsweb.cfg
/usr/lpp/dfsweb/etc/bonnell/dfsweb.cfg

Configuration File Name:
/usr/lpp/dfsweb/etc/bonnell/df

Figure 22. DFS Web Manager Panel

With DFS Web Secure opening up your DFS filespace to Web access under the protection of DCE Security, it is clear that the administration of DCE security and DFS ACLs is going to be very important. The DCE Administration interface supplied with DFS Web Secure can be used to administer precisely those components of DCE. The initial panel for this DCE administration can be seen in Figure 23 on page 57.



Edit user: Cell /.../itsc.austin.ibm.com

To edit a user, type the user's name and then click on the Load user push button.

User name	
Settings	Basic
	Advanced

Figure 23. DCE Administration Panel of DFS Web Secure

4.3 Security Considerations on a Web Server

With DFS Web Secure in place, the end-to-end path from the Web browser to the resources served by the Netscape server and DFS Web Secure is shown in Figure 24 on page 60. For each major component in this path, there are security issues and means of applying security to them. Working down the diagram from the browser to the targeted resource, here are some important security considerations for each component in a DFS Web Secure scenario:

1. Browser-side security involves the browser holding the ID and password combinations for URLs requiring identification of the user to the Web server. This is also the case with any access to a URL serviced by DFS Web Secure. It is unfortunate that with the current generation of browsers it is necessary to restart the browser, or to start another instance of the browser application (not just another browser window), in order to change the user identity associated with a particular URL. As an example, this might be necessary if a user wants to change his/her DCE identity between administering DFS Web Secure and browsing information within DFS.

2. Security of the client system from applications downloaded by the browser and executed locally is enforced by the Java environment inside a Java-enabled browser or by any other plug-ins installed with the browser. We do not cover client security issues in more detail since this is a more general issue and not specifically relevant to DFS Web Secure.

3. The browser-to-server connection over a TCP/IP network, whether Internet or intranet, is where a firewall is the key component when implementing security. A Firewall is put in place to allow only TCP/IP traffic for a restricted

set of valid Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) port numbers. The Web server's port number must be configured into the firewall setup. Any browser requests for information through DFS Web Secure will pass through this port; so no additional firewall setup is required specifically for DFS Web Secure. The general use of a firewall with DCE is covered in Chapter 8, "Bridging Networks Together: DCE Over Firewalls" on page 143.

4. User Authentication with the Web server is carried out by a standard dialog between the browser and the Web server when a 401 message is generated in response to an HTTP request. In a standard DFS Web Secure installation, this will be the mechanism used for a user to authenticate with DCE Security. The dialog prompting the user for ID and password has a weak point in that the user's password is transmitted over the Internet/intranet using only basic HTTP security (Base64 encoding), which is pretty straightforward to decrypt. A better solution could be to set up the Web server to use SSL. Using SSL will result in all communication between the browser and the Web server being securely encrypted, which will mean that all of the data being served by DFS Web Secure will in turn be encrypted as it passes across the network.

   Setting up SSL is well covered in the Netscape server online help. A brief outline of the steps involved follows:

   a. Generate a key. There is a program in your Netscape server directory to do this as `root` (default location /usr/ns-home/bin/https/admin/bin/sec-key). Note the password that we chose for the key pair.
   b. Request a certificate from a Certificate Authority. This will most probably be done by e-mail.
   c. Install the certificate onto your server through the Administration Server interface (under the Encryption category). You will need the encryted Certificate sent to you by the Certificate Authority.
   d. Enable Encryption by activating SSL for your server from the Administration Server interface. You must specify a port number (the default port number for SSL is 443).

   Browsers should then enter DFS by requesting the URL with a prefix of `https` instead of `http`.

5. Netscape Access Control allows you to restrict access to the resources served by your Web server. Access rights can be defined for your Web server as a whole or selected resources within it. These rights are separate from the permissions defined in the AIX filesystem(s) containing the resources. Either read or write access can be granted or denied for the following categories of requester:

   - Users
   - Groups of users
   - Particular hosts
   - Groups of hosts

   Users and groups are defined to the Netscape server and held in user databases. User names can be imported from AIX into the Netscape user database but not their passwords because they are not held in /etc/passwd.

   For DFS Web Secure, it is not necessary to set up Netscape Access Control. In fact if you want to use Netscape Access Control on a server with DFS Web Secure installed, it is important that you do not restrict read access at the overall server level. This causes confusion as both authentication schemes try to get your credentials at the same time.

6. DFS Web Secure automatically uses the DCE Security Registry to authenticate users and then access to resources within DFS is controlled according to the DFS Access Control Lists (ACLs) enforced through the use of DFS Web Secure can support more detailed (finer-grained) security definitions for its resources than Netscape Access Control. In addition the administration of DCE/DFS access control is kept in one place, even though the resources being served by DFS Web Secure could span many server systems. This single point of administration for DFS access control is even more valuable when an organization sets up several Web servers to serve resources from an organization-wide DFS file space using DFS Web Secure.

   It is important that you have appropriate restrictions in place for the `anonymous` account as mentioned in 4.2.1, "The Quick Start" on page 52.

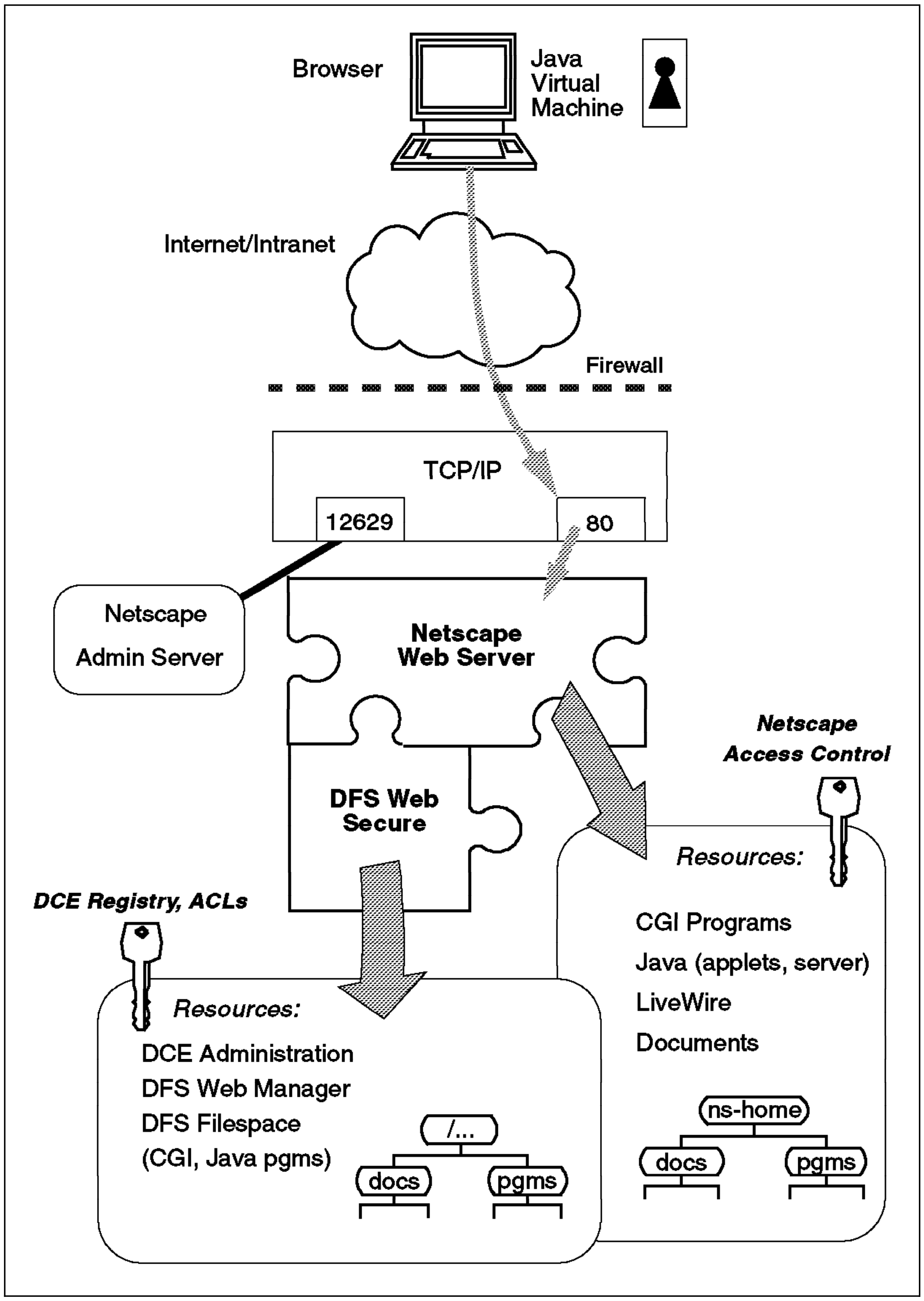


Figure 24. DFS Web Secure: End-to-End Components

Some general points on security not specifically placed on Figure 24 that apply to the Web server setup are:

- The Web server processes must be run under a separate user identity with limited privileges. Consider also the group memberships of this user and the privileges associated with those groups—perhaps even set up a separate group just for this user.
- Check other TCP/IP services running on the system. Some of these with a default setup will leave an inadequately secured back door to the system.

Examples of these have been Finger, Sendmail and TFTP. There are documents on the Web that cover exposures of this kind (for example, Satan).

- Filesystem permissions for Web server resources should offer protection. In particular they should not be writeable by the Web server ID (or group if you generally adopt group access control).
- Hard symbolic links in the Web resource areas can be followed by the Web server, which could offer more open AIX file permissions than you intended for the target files of these links. It is best to avoid having links in the filesystem(s) containing Web resources.
- The underlying AIX Operating System setup can be made more secure by setting up the Trusted Computing Base (TCB), which keeps track of modifications to key system files.
- Physical security and sound security procedures also need to be in place. This means controlling physical access to the system as well as procedures by which key information about the system is held. For instance, how are important passwords kept, and what are the rules for their content, frequency of changes and so forth?

  On an ongoing basis, monitor logs for the system, TCP/IP and the Web server.
- Screen executable resources that are being placed into Web Resources area, particularly CGI shell scripts.

This is not a complete list of the security considerations that apply to a Web server setup. There is more detailed coverage of how to build a secure Web server environment in the Redbook *Safe Surfing: How to Build a Secure WWW Connection*, SG24-4564.

Chapter 5. Running the Business With DFS Web Secure

With DFS Web Secure installed in addition to a Netscape Web server, there is instantly the potential for standard Web browsers to access the resources of a worldwide DFS filespace. This is true whether an organization has one or multiple DCE cells set up across the world. Figure 25 shows the capability provided by a single installation of DFS Web Secure in an organization with three DCE cells implemented worldwide. The huge amounts of data and applications (of a wide range of types) that can be served through DFS Web Secure are still subject to the file-level security capabilities of DFS. This is because DFS Web Secure integrates with the standard DCE Security and DFS Access Contol Lists that already safeguard access to the DFS filespace.

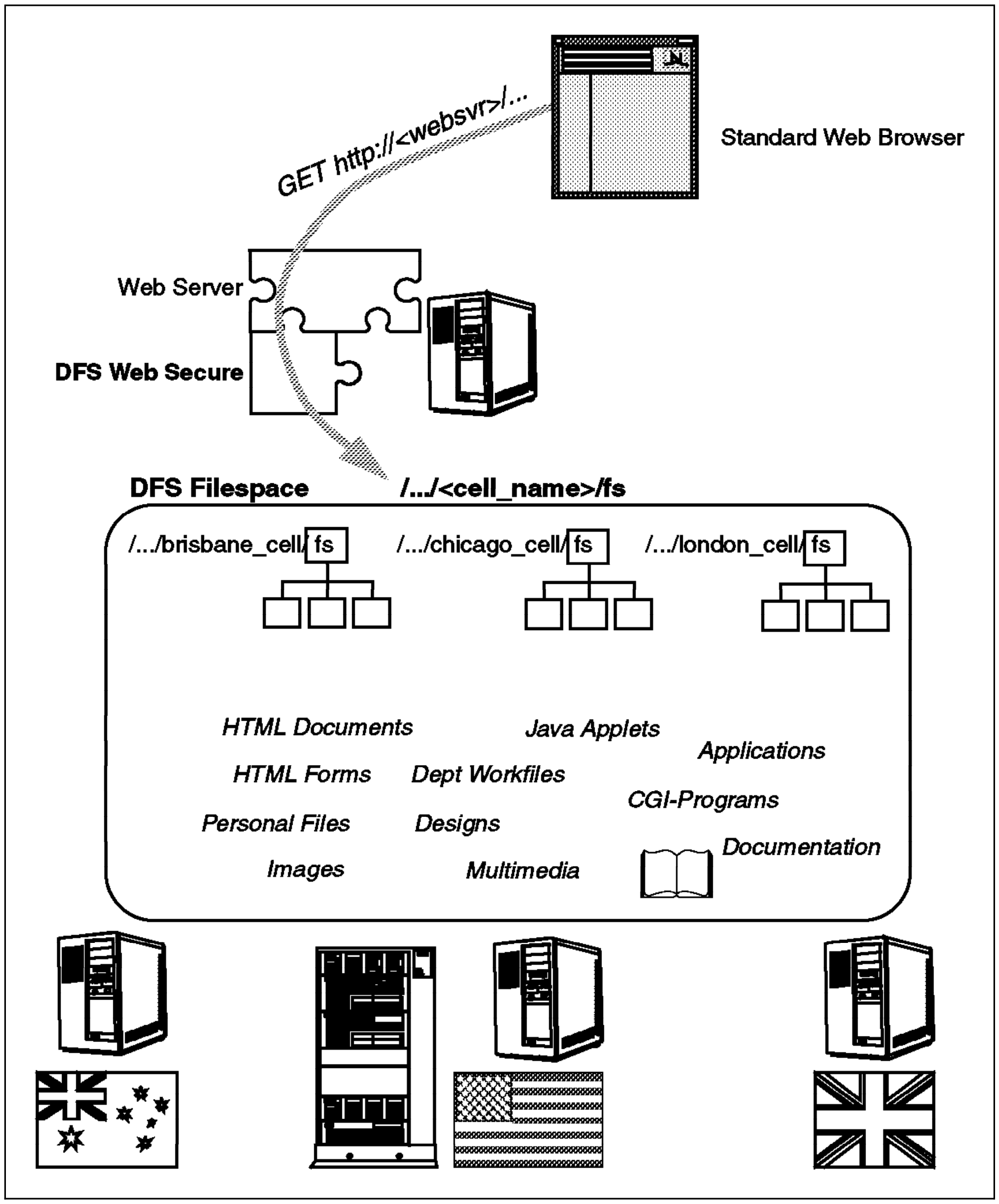


Figure 25. DFS Web Secure Providing a Global Web Resources Pool

DFS has several advantages over a stand-alone file server and also over alternative remote file serving products. These include:

- Higher availability of data and resources than can be achieved with replication. A copy of a file can be accessed even if a machine serving the file goes down.
- The ability to share information throughout a very large-scale system and use a single, consistent global namespace. Multiple servers of different machine architectures can export their file systems to this global namespace.
- Fast file access performance with cache technology that also maintains file consistency across mutliple, concurrent client accesses.

To a Web server administrator (sometimes named the Webmaster) running a Web service in a large organization, whether attached to the Internet or an intranet within the organization, these features of DFS provide some key capabilities for building a reliable, highly available and scalable Web service. In addition there are some important benefits in terms of flexibility and ease of administration for the Webmaster:

- There is a single point of administration for the DCE registry containing user and group information and also for the ACLs on each file stored within DFS. This would also be the case if there were multiple Web servers installed with DFS Web Secure providing multiple Web access points to DFS. A standard Access Control setup on two Web servers, such as the one implemented in Netscape's servers, would have to be maintained on each Web server and both setups would have to be kept in step with each other.
- DFS allows files to be replicated on separate server machines. This can be used to improve performance as well as to provide an alternative, live copy in case of a server system crash (which would then go unnoticed by the client).
- Files and filesets in DFS can be moved between physical file server machines without disrupting client access to those files. The path to the file, which may be referenced in links within multiple HTML documents, remains the same. This type of file location transparency provided by DFS becomes more valuable as the Web service scales up. For example if a set of files needs to be moved to a higher performance file server, there is no maintenance effort required for all the HTML links to those files in any existing documents.

A lot of the operational issues to be tackled when running the business with DFS Web Secure echo the operational issues of running a DCE environment with a DFS filespace. This chapter covers the major ones.

5.1 Managing User Access

Every browser user who wants to access a URL that is provided by DFS Web Secure has to authenticate with DCE security. This means that a valid user name and password combination has to be entered into the dialog box that pops up on the Web browser when the user first references a URL that is being provided by the DFS Web Secure extension to the Web server. A null response in this DCE authentication dialog (that is when the user enters nothing or blanks) will be rejected; unauthenticated access to DFS Web Secure is not permitted, even if DFS itself is open to some unauthenticated access.

Managing user access involves setting up users within the DCE registry and also the groups and the organization to which those users belong. Access to the resources served by DFS Web Secure, both files within DFS and its administration functions, is then controlled according to rules based on this information about the DCE user. This information about a DCE user is held within a DCE credentials file, and access to a resource is granted if these credentials pass the access control criteria set for the resource.

5.1.1 Standard DCE User Administration

There are a number of utility programs that can be used for adding and modifying the DCE registry. Original DCE implementations usually included the rgy_edit command, and more recently (since OSF DCE 1.1) these facilities are also available through the DCE control program, dcecp. There is also a range of vendor products available which provide graphical tools for administering the DCE registry. DFS Web Secure also includes a Web browser interface for carrying out administration of the DCE registry and ACLs.

For a first example, we look at using dcecp, which is provided in the base DCE implementation on a variety of operating system platforms. Here are the options that are provided within dcecp for creating a user definition in the DCE registry:

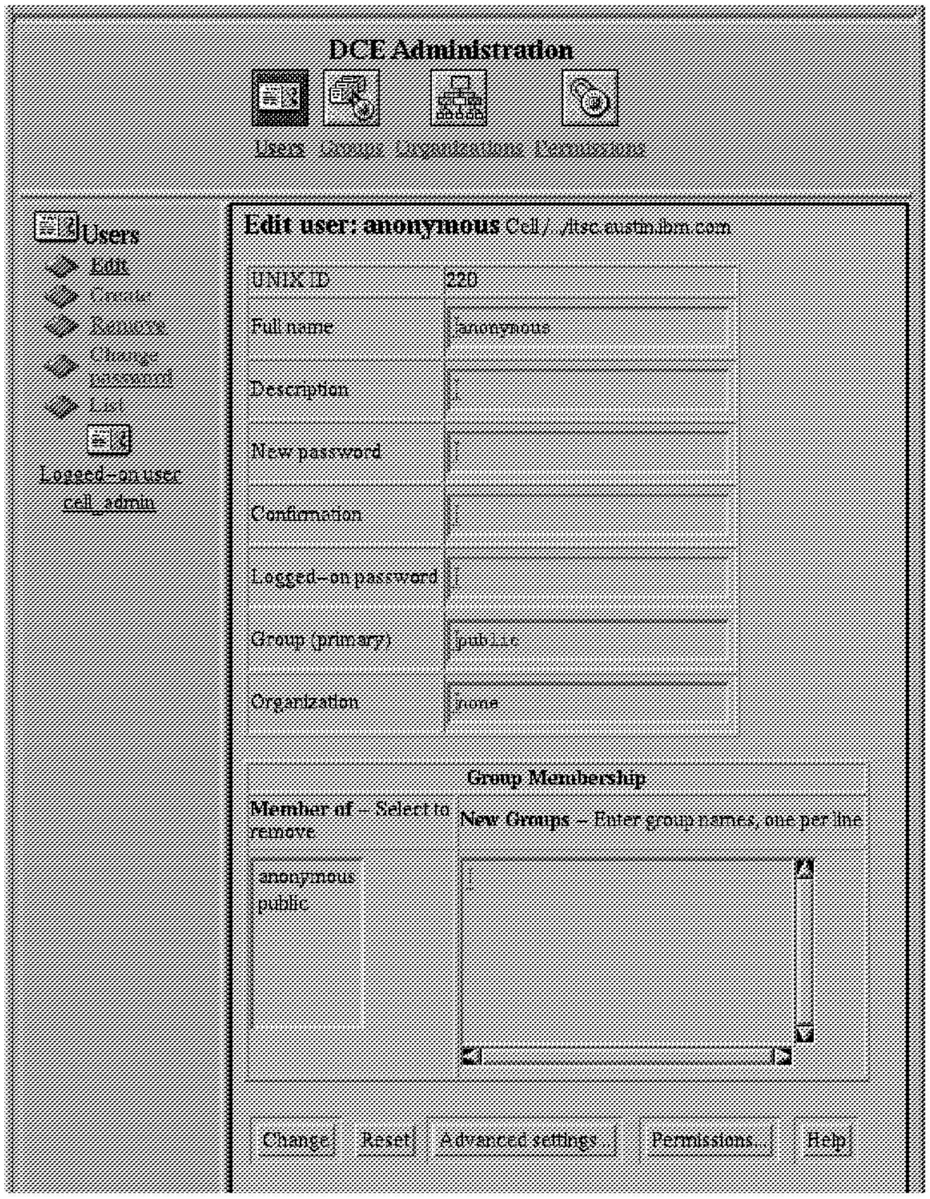
```
dcecp> user help create
-alias          Add principal named as an alias of specified uid.
-attribute      Provide attributes in an attribute list format.
-client         Can the account principal be a client.
-description    A general description of the account.
-dupkey         Can the accounts's principal have duplicate keys.
-expdate        When does the account expire.
-forwardabletkt Can the account receive a forwarded ticket.
-fullname       Fullname of user.
-force          Force creation of group or organization if they don't exist.
-group          Primary group user should be member of.
-home           Users home directory.
-organization   Primary organization user should be member of.
-maxtktlife     The maximum ticket life for the account.
-maxtktrenew    The maximum ticket renewal time.
-mypwd          Password of user adding account.
-password       Initial password of user.
-postdatedtkt   Can the principal receive a post dated ticket.
-proxiabletkt   Can the account issue a proxiable/forwarded ticket.
-pwdvalid       Is the password valid.
-renewabletkt   Can the account receive a renewed ticket.
-server         Can the principal act as a server.
-quota          Quota of the principal to be added.
-shell          Initial shell of user.
-stdtgtauth     Can the account's principal use the standard ticket granting service.
-uid            User Identifier of the principal to be added.
```

This is a complete list of the attributes in the DCE registry pertinent to a user account. In practice the defaults are acceptable for a number of them. Here is an example of the command to add the recommended anonymous user for DFS Web Secure to the DCE registry. This should be run as cell_admin (note the password for cell_admin has to be included), and the group and organization need to have been created already.

```
dcecp> user create anonymous -group public -org none
                -force -password anonymous -mypwd -dce-
```

5.1.2 DFS Web Secure User Administration GUI

DFS Web Secure offers a GUI-based alternative for creating new accounts or for changing the attributes of existing user accounts. A sample screen for editing the basic parameters is shown in Figure 26, and clicking the Advanced Settings button allows you to access the full set of attributes. Since it provides a Web browser interface, this application can be used remotely from machines that are not running a DCE client.



Edit user: anonymous Cell /.../itsc.austin.ibm.com

UNIX ID	220
Full name	anonymous
Description	
New password	
Confirmation	
Logged-on password	
Group (primary)	public
Organization	none

Group Membership

Member of – Select to remove	**New Groups** – Enter group names, one per line
anonymous public	

Figure 26. Basic User Administration Screen from DFS Web Secure Administrator GUI

Note on Credential Files

Since DFS Web Secure acts as a proxy for the user, it performs a login to DCE in order to get the login context, which is stored in credential files. These files, stored in /var/dce/security/creds, are not removed automatically and may fill up the filesystem over time. It is therefore an important housekeeping task to run the `rmxcred` command regularly on the system where DFS Web Secure is installed and running. A good solution is to run `rmxcred` once every night, controlled by a cron job.

5.2 Supervising and Tracking Access

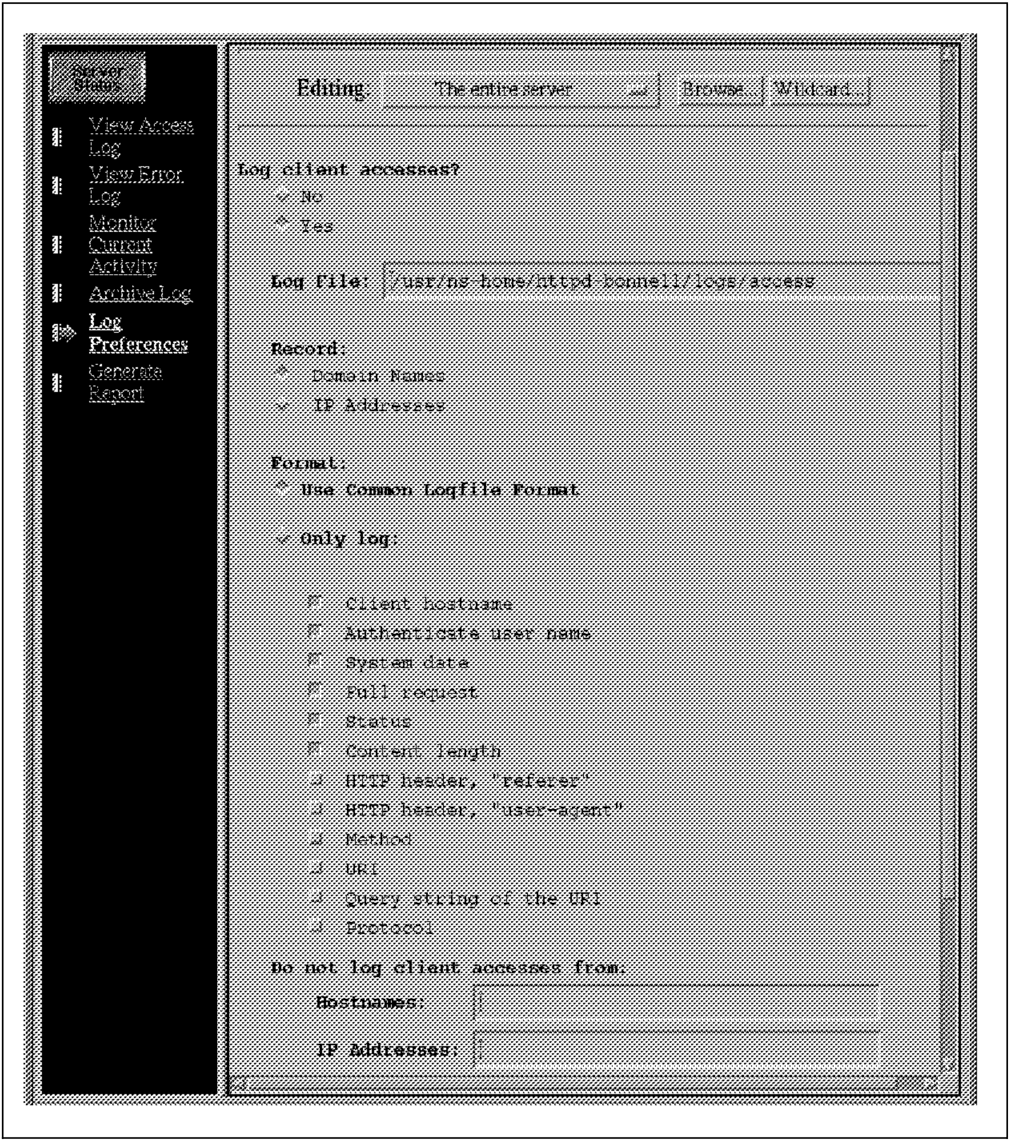
Whether a Web server is open for business on the Internet or on a company wide intranet, it is going to be important to have the capability to track the accesses that are made to the server. This information is vital if you want to understand and maintain the security of a Web server and is useful for assessing the demands of its users and potentially for planing additional capacity that may be required. Problems within the configuration of the Web server and some of its HTML content can also be identified if invalid accesses are consistently being logged.

So a Web server should log information about accesses (both valid and invalid) that are made to its resources. The Netscape FastTrack Server that we were using maintains both an access log and an error log. By default, information is also logged into these files for accesses that are being made to DFS Web Secure and its resources.

DFS Web Secure also includes a separate monitoring facility within its DFS Web Manager component. As well as monitoring resource accesses and failed logins to DCE, this facility also allows you to set threshold rules and trigger automatic actions in case thresholds rules are met. This is covered in some detail in 5.3, “Monitoring and Thresholds in DFS Web Manager” on page 72. DFS Web Manager can also be used to monitor accesses that are made to non-DFS resources after a user has authenticated with DCE. Although this may appear to be just duplicating the Web server access logging, it does provide a useful way of focusing on the resources accessed by each individual DCE user, and it enables the use of the threshold rules function for resources outside DFS. If users do not have to authenticate with a Web server, then it can be very difficult, or even impossible, to ascertain where accesses are coming from if socks servers are in use on the network, since only IP addresses are logged by the Web server, unless users were required to authenticate.

The Netscape servers automatically record information in the `access` and `error` log files. These correspond with files of the same names in the directory /usr/ns-home/httpd-<server_name>/logs. These files contain clear text; so they can be viewed directly, but Netscape also provides a log analyser tool that can be used to generate reports based upon the log data within these files. This tool can be run from the command line or from within the Netscape Administration Server interface, which also allows you to customize the server logging. To work with the server logs from the Netscape Administration Server interface, select the server instance you want to work with from the Server Selector screen. Click on **Server Status** in the frame at the top of the screen, and the relevant log analysis operations will be listed on the left hand side of the upcoming screen. Here is a summary of the capability provided by these operations:

- The View Access Log and View Error Log panels allow you to see the last specified number of entries of the relevant log file, and you can also limit the display by entering a search string in the field *Only show entries with*.
- The Monitor Current Activity panel lets you run the interactive server monitor. This allows you to see the current server activity and see how heavily your server is being used. A separate window is displayed with the graphical monitor output, which is refreshed at regular intervals. For a new Web server installation, this can help you confirm how many threads you should configure.
- The Archive Log panel is where you can archive log files and specify times to rotate the log. Clearly the logs will be fast growing files on a busy Web site, and so this allows you to manage the space requirements better in the Web server file system. It is recommended that you archive log files before running the log analyzer. After taking an archive, it is necessary to restart the Administration Server.
- The Log Preferences panel shown in Figure 27 on page 69 is where you customize the information that is being written into the logs. From this screen, you can specify logging criteria for different parts of the server (select the area you want to work with on the line with the *Editing* field) and also limit logging to exclude ranges of hostnames or IP addresses. The panel shown in the figure shows the default options. With the field *Authenticate user name* flagged on, the log will record this information, which will cover both Netscape Access Control user names and DCE user names (for instance, logged against all accesses through DFS Web Secure).



Editing: The entire server

Log client accesses?
No [✓ Yes]

Log File: /usr/ns-home/httpd-bonnell/logs/access

Record:
[✓ Domain Names] IP Addresses

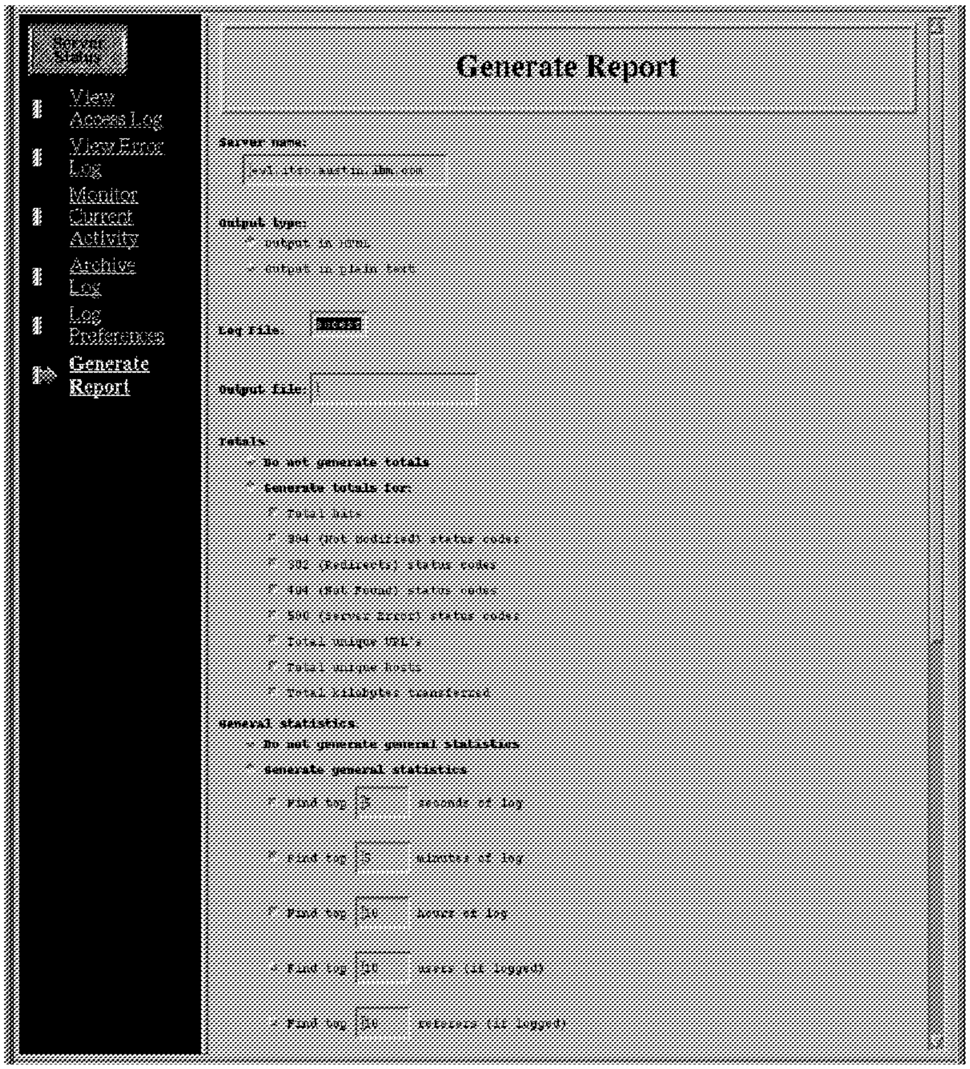
Format:
[✓ Use Common Logfile Format]
Only log:

- ✓ Client hostname
- ✓ Authenticate user name
- ✓ System date
- ✓ Full request
- ✓ Status
- ✓ Content length
- HTTP header, "referer"
- HTTP header, "user-agent"
- Method
- URI
- Query string of the URI
- Protocol

Do not log client accesses from:
Hostnames:
IP Addresses:

Figure 27. Setting Log Preferences Panel for Netscape FastTrack Server

- The Generate Report panel is where you can run the log analyzer and choose the analysis that you want to have performed on a log file. A report is generated and either displayed or written into a separate file that can be either HTML or plain text. Alternately, you can run the log analyzer from the AIX command line. The command `flexanlg` is under the extras subdirectory in your server home directory (usually /usr/ns-home/extras/flexanlg). The panel shown in Figure 28 on page 70 shows some of the types of analysis that you can ask to be performed on the specified log file.



Generate Report

Server name: ev1.itso.austin.ibm.com

Output type:
- Output in HTML
- Output in plain text

Log File: access

Output file:

Totals:
- Do not generate totals
- Generate totals for:
  - Total hits
  - 304 (Not Modified) status codes
  - 302 (Redirects) status codes
  - 404 (Not Found) status codes
  - 500 (Server Error) status codes
  - Total unique URL's
  - Total unique hosts
  - Total kilobytes transferred

General statistics:
- Do not generate general statistics
- Generate general statistics
  - Find top 5 seconds of log
  - Find top 5 minutes of log
  - Find top 10 hours of log
  - Find top 10 users (if logged)
  - Find top 10 referers (if logged)

Figure 28. Generate Report Panel for Netscape FastTrack Server

Here is some sample output from a plain text report file:

```
--- ev1.itsc.austin.ibm.com statistics ---
--------------------------------------------------------------
For the period between 17/Feb/1997:09:55:27 and 11/Mar/1997:09:23:36 -

                   Total hits:     2489
         Not Modifieds (304's):     231
             Redirects (302's):      40
            Not Founds (404's):     103
         Server Errors (500's):      51
        Number of unique URL's:     265
     Total unique client hosts:       7
Total unique destination hosts:       0
   Total kilobytes transferred:    8746


The 5 top one second periods
Count       Bytes  304's  Date/Time
--------------------------------------------------------------
   20           0     20  17/Feb/1997:12:27:08
   10           0     10  04/Mar/1997:12:20:16
    8        6027      0  03/Mar/1997:23:16:52
    8       10822      0  04/Mar/1997:12:20:08
    8       80816      0  21/Feb/1997:17:34:18
```

```
The 5 top one minute periods
Count       Bytes  304's  Date/Time
--------------------------------------------------------------
   48      172941      0  03/Mar/1997:23:16
   41       98025     12  04/Mar/1997:12:20
   31      217919      0  04/Mar/1997:11:39
   30      217720      0  17/Feb/1997:12:26
   27       94338      0  25/Feb/1997:18:49

The 10 top one hour periods
Count       Bytes  304's  Date/Time
--------------------------------------------------------------
  268     1044126     21  04/Mar/1997:12
  203      788912      2  04/Mar/1997:11
  178     2097199      0  04/Mar/1997:16
  151      805576      9  04/Mar/1997:10
   93       39634     39  17/Feb/1997:10
   93      247694      9  21/Feb/1997:17
   88       60768      9  09/Mar/1997:16
   65      287888      0  03/Mar/1997:23
   64      184744     10  19/Feb/1997:09
   62      337526      0  26/Feb/1997:16

The 10 top users of the period
Count       Bytes  304's  Value
--------------------------------------------------------------
 1087     3575936    168  -
 1054     4817071     38  cell_admin
  199      273479     22  anonymous
  101       90120      3  cio
   22      193179      0  surfer
   21       77911      0  user0001
    5       48973      0  member

The most commonly accessed URL's
Number of         Total
 Accesses         Bytes    Size  304's  URL
--------------------------------------------------------------
      288        347594    2453      0  /dfsweb/dwcgi-bin/am/dwxacts.x
      136        117441    1938      0  /dfsweb/dwcgi-bin/dfswebi
      104       1184172   55564      0  /dfsweb/dwcgi-bin/am/dwxdb.x
       91        373114   10624      0  /dfsweb/dwcgi-bin/am/dwxcfg.x
       76         94035    2800      1  /dfsweb/
       55          2520      60     13  /mc-icons/blank.gif
       55          7308     174     13  /mc-icons/back.gif
       53       1704764 1613784      0  /dfsweb/dwcgi-bin/dfswebu
       48         73302    2785     18  /
       45          4765     305      0  /:/
       45       1195101   44263     18  /title.gif
       43         16845    1484      0  /cgi-bin/simple_cgi.cgi
       40         22735     953      8  /:/data/web/private/staff/andy.html
       35        139018    6319     13  /:/data/web/private/staff/wire.gif
       34          9782     336      0  /:/data/web/private/staff.html
       33        152789    6643     10  /dfsweb/icons/dfswebm.gif
       33       1502184   62591      9  /dfsweb/icons/dfswebt.gif
       33          7301     224      0  /:
       33         10804     859     10  /form.html
       30          3036     132      7  /dfsweb/icons/bullets.gif
       28         20424     888      5  /dfsweb/icons/backgrnd.gif
       28        143026    7331      0  /dfsweb/dwcgi-bin/am/dwxthr.x
       26          4711     305      0  /cgi-programs/simple_cgi
       25          5483     229      0  /dfsweb
       21         18445    1085      4  /dfsweb/icons/dfswmgr.gif
       20         22355    1315      3  /dfsweb/icons/dfswebs.gif
       20          3240     162      0  /dfsweb/icons/bulletu.gif
       20          2125     125      3  /dfsweb/icons/bulletna.gif
       19           833     305      0  /:/data/web/
       19         14432     902      3  /dfsweb/icons/bytesdb2.gif
       19         14896     931      3  /dfsweb/icons/urldb2.gif
       19           811     223      0  /:/data/
       19         15792     987      3  /dfsweb/icons/failgdb2.gif
       12             0       0      0  /:/home/
```

```
The top 5 client hosts accessing server
Number of         Total
 Accesses          Bytes   304′s  Host/IP
-------------------------------------------------------------
     1124       5960947      52  itsorusi.itsc.austin.ibm.com
      708       1655007     117  9.3.1.74
      425        763689      42  ev2.itsc.austin.ibm.com
      137        304111      19  9.3.1.15
       60        290478       0  129.35.223.11
```

Notice that DFS pathnames are being logged amongst the URLs and that DCE user names, such as `cell_admin`, also appear in the authenticated user analysis. The analysis of top 10 users had to be selected in the Generate Report panel since this was not a default report item.

For DFS administrators, the counters for the various status codes can be of interest (for instance 404 - file not found). Details for these accesses can be obtained from the access log file. The error log file has detailed information on a range of things, including failed attempts to authenticate with DCE, which may be of interest to DCE administrators. Here is an extract from an error log file showing an invalid attempt to log in as DCE user x1:

```
[10/Mar/1997:13:40:14] info: for host asocks1.server.ibm.com trying to GET /dfsweb,
      dce-s-auth reports: session_manager reports error 387063936 returned from routine
      sec_login_validate_identity: ″Invalid password (dce / sec)″
[10/Mar/1997:13:40:14] security: for host asocks1.server.ibm.com trying to GET /dfsweb,
      dce-service reports: unknown user/pw x1
```

In addition to these logging functions provided by the Web server, DFS Web Secure includes a monitoring facility with the capability to set thresholds, as explained in the next section.

5.3 Monitoring and Thresholds in DFS Web Manager

The DFS Web Manager is a plug-in to the Netscape server provided with DFS Web Secure. It allows you to monitor Web interactions that are made through DFS Web Secure and it also allows you to define threshold rules for this information which can trigger actions if rule conditions are met. When monitoring is turned on for all resources, you effectively have a set of counter databases that are being constantly updated with the following information:

- The number of times a URL is accessed. The URL Counter database can be used to determine the popularity of certain URLs and to help you improve load balancing. This information can be used in determining which popular documents within DFS may be candidates for replication or location on a fileserver nearer to the Web server for availability and performance reasons. This is particularly relevant if the DFS file space spans several networks with varying link capabilities in terms of throughput and those documents are frequently updated. This shortens the time to reload the updated documents into the DFS cache on the Web server.
- The number of bytes that a user has accessed. The Byte Counter database can be used to help you to determine the download needs of your users and for related accounting information. This information can be used to assess whether the size of the DFS cache on the Web server machine is sufficient to support typical throughput requirements.

- The number of transactions a user has performed, shown as a count of the number of page accesses made by a user. The Page Counter database helps you determine the load users generate on your Web server.
- The number of times a user has made a failed attempt to login. The Failed Login Counter database helps you determine inadvertant or unauthorized login attempts to DCE/DFS through your Web server.

Monitoring with DFS Web Manager can in fact be switched to count Web transactions across the whole Web server and not just interactions passing through DFS Web Secure, so long as they invoke a DCE login.

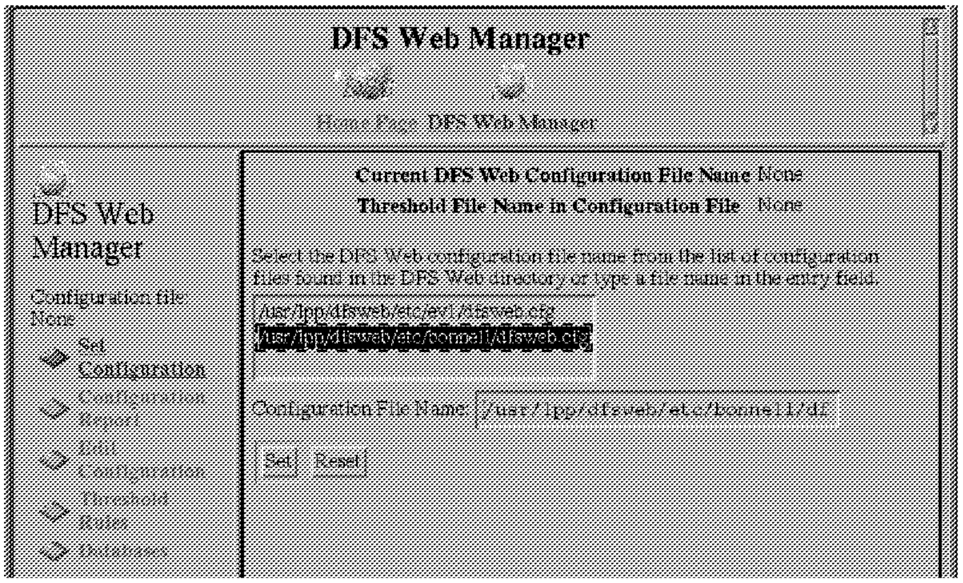
Each monitored resource can have a set of threshold rules defined for it. With each update to any of the database counters, DFS Web Manager checks the updated counter value against the set of threshold rules associated with that counter. If the threshold rule matches, DFS Web Manager initiates one of the following trigger actions:

1. A program is run. For example, this could be used to call a pager.
2. An e-mail is sent to an e-mail address. This can include details of the threshold that has been exceeded.
3. The counter is cleared (reset to 0).
4. No action is taken, but a record is written to a log file with details of the triggered threshold rule.

The DFS Web Manager interface, which allows you to configure the monitoring and set threshold rules, runs on a frames-capable browser. The interface also allows you to view the contents of the databases with the current counter values. To access the DFS Web Manager interface, direct your browser to the DFS Web Administration home page, which is usually at `http://<server_name>/dfsweb/index.html`. From there, click on **DFS Web Manager**.

There is an online tutorial available from the DFS Web Administration home page. In the following pages we cover some of the key points for setting up monitoring of a single resource with a threshold rule and multiple triggers (essential if you want to mail a distribution of e-mail addresses, for example).

The initial panel, shown in Figure 29 on page 74, is where you select the configuration file that DFS Web Manager is going to use. A DFS Web configuration file is automatically created at DFS Web Secure installation time (by `mkdfsweb`), and there should be one file listed per DFS Web Secure instance on your system. You must select the relevant file name from the list, corresponding to the server instance that you want to work with, and then click the **Set** button before proceeding any further.



Current DFS Web Configuration File Name None
Threshold File Name in Configuration File None

Select the DFS Web configuration file name from the list of configuration files found in the DFS Web directory or type a file name in the entry field.

/usr/lpp/dfsweb/etc/ev1/dfsweb.cfg
/usr/lpp/dfsweb/etc/bonnell/dfsweb.cfg

Configuration File Name: /usr/lpp/dfsweb/etc/bonnell/df

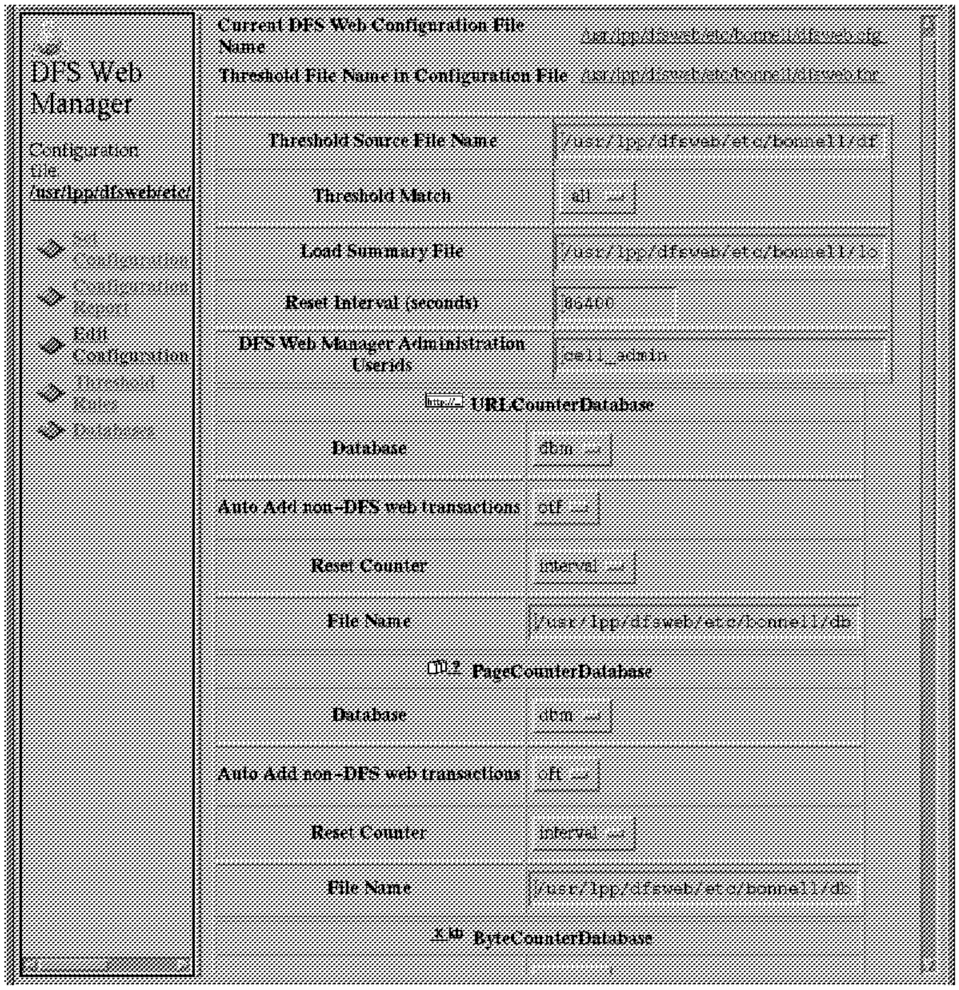
Configuration file: None

Figure 29. DFS Web Manager's Initial Set Configuration Panel

Monitoring is switched on and customized from the Edit Configuration panel, reached by selecting **Edit Configuration** from the DFS Web Manager panel on the left of the screen. Some key fields on this panel, shown in Figure 30 on page 75, are:

- All file name fields should contain valid file names on the server system. For thresholds to be checked, a file containing valid threshold rules must be present at the path specified in Threshold Source File Name. If the rules are invalid then the whole configuration file is invalid. There is a Validate Configuration button at the bottom of the panel that should always be used after making changes in this panel.
- The Threshold Match field can be set to *all* or *first*. Set this field to *all* if you want to have multiple trigger actions executed when a threshold is exceeded, for instance if you want to mail several addresses or if you want to run a program and reset the counter. This switch applies to all monitoring.
- The Database field for each monitored resource is used to switch on monitoring for that resource. Select **dbm** to switch on monitoring (this is the default) or **none** to switch monitoring off.
- The Reset Interval in seconds can be used in conjunction with the Reset Counter field for each monitored resource. If the Reset Counter field is set to *interval,* then the counters in the database will be reset when the last reset time for the transaction plus the reset interval has passed or equals the current time. If it is set to *never,* then the counters will be incremented continually, until reset by a *clear* trigger action or manually reset by the administrator through the interface.
- The Auto Add non-DFS Web transactions switch for each monitored resource allows you to include non-DFS transactions in the database for users who have logged into DCE. It will not include transactions for a browser user accessing the Web server who has not authenticated with DCE via DFS Web Secure.

After making changes in this panel, it is important to press the buttons at the bottom to **Validate Configuration** and if that is successful **Change Configuration**. You will need to restart the Web server in order for the changes to take effect.



Current DFS Web Configuration File Name /usr/lpp/dfsweb/etc/bonnell/dfsweb.cfg
Threshold File Name in Configuration File /usr/lpp/dfsweb/etc/bonnell/dfsweb.thr

Threshold Source File Name	/usr/lpp/dfsweb/etc/bonnell/df
Threshold Match	all
Load Summary File	/usr/lpp/dfsweb/etc/bonnell/lo
Reset Interval (seconds)	86400
DFS Web Manager Administration Userids	cell_admin
URLCounterDatabase	
Database	dbm
Auto Add non-DFS web transactions	off
Reset Counter	interval
File Name	/usr/lpp/dfsweb/etc/bonnell/db
PageCounterDatabase	
Database	dbm
Auto Add non-DFS web transactions	off
Reset Counter	interval
File Name	/usr/lpp/dfsweb/etc/bonnell/db
ByteCounterDatabase	

Figure 30. DFS Web Manager's Edit Configuration Panel

Threshold rules can be added and modified for each monitored resource from the Threshold Rules panel (selected from the panel on the left side of the screen). From this panel you can either add, modify or delete a threshold rule for a monitored resource. An example of a threshold rule is shown in Figure 31 on page 77. The fields of interest are:

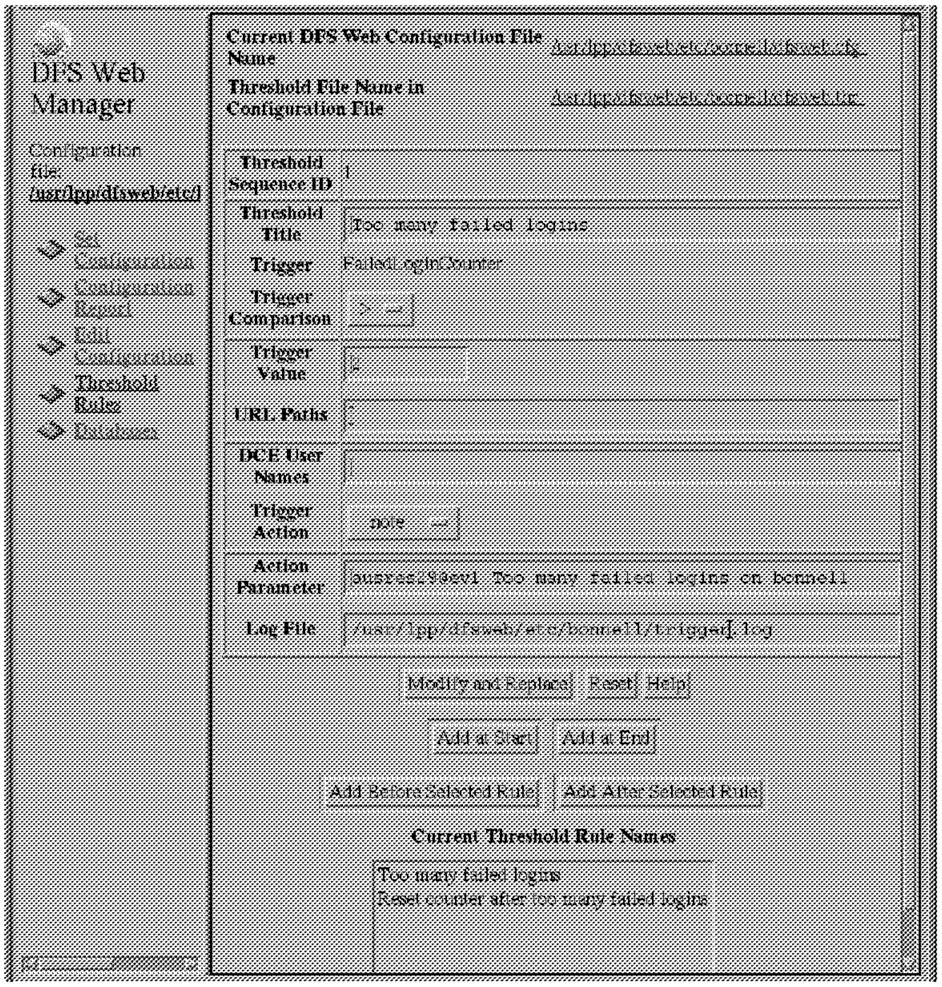
- The threshold rule must have a title in the Threshold Title field.
- The Trigger Comparison field is where you set the logical comparator that will be used to test the counter with the Trigger Value. In the example screen capture, we are setting the threshold to trigger when the FailedLoginCounter *is greater than 2*.
- The Trigger Action field can be set to either *note* (to send e-mail), *program*, *clear,* or *none*. With *note* and *program,* it is necessary to fill in an Action Parameter to specify the e-mail address (with some optional message text) or to instruct the program to execute.

In the screen shot in Figure 31 on page 77, we are specifying that trigger e-mail be sent to the ID ausres29@ev1 with some additional text that will be shown within the e-mail in a Note parameter field. The actual e-mail that is sent by a trigger will contain information about the threshold rule to which it applies; so it is not essential to include additional text with the e-mail ID. Here is the e-mail that is triggered by the threshold rule shown in the screen capture:

```
To: ausres29@ev1.itsc.austin.ibm.com
Subject: DFS Web Threshold ′Too many failed logins′ Triggered

Monday, 24-Mar-97 10:50:49 CST

Rule name:  Too many failed logins
Note parameter:  Too many failed logins on bonnell
FailedLoginCounterDatabase record:
   cio  0000000002

Web transaction information:
   Path:  /:
   Hostname:  itsorusi.itsc.austin.ibm.com
   Login name:  cio
```

- The Log File is optional if you want to keep a log of triggers.
- As well as applying your threshold to the rules file, the buttons near the bottom of the panel allow you to control the sequence of the rules that will be checked for this monitored resource. The sequence of the rules is important if you are only going to have the *first* rule that matches its check triggered (this is governed by the Threshold Match parameter in the Edit Configuration panel).

You will need to restart the Web server in order to pick up the changed threshold rules.



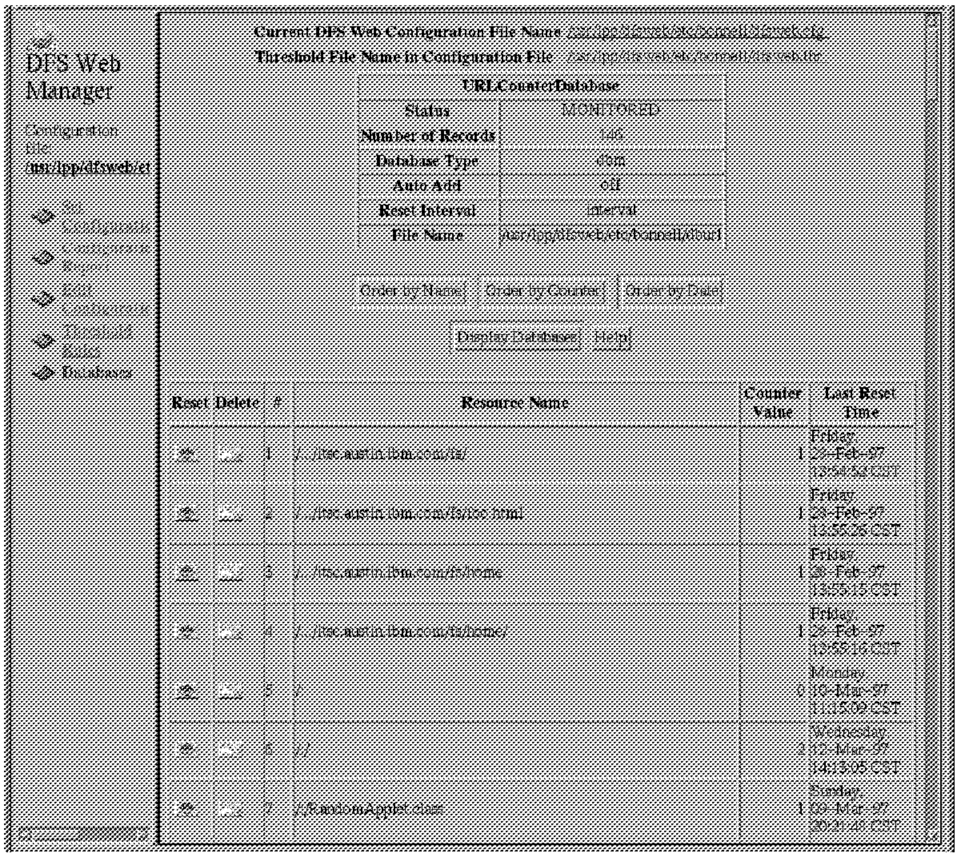
Current DFS Web Configuration File Name	/usr/lpp/dfsweb/etc/bonnell/dfsweb.cfg
Threshold File Name in Configuration File	/usr/lpp/dfsweb/etc/bonnell/dfsweb.thr
Threshold Sequence ID	1
Threshold Title	Too many failed logins
Trigger	FailedLoginCounter
Trigger Comparison	>
Trigger Value	2
URL Paths	
DCE User Names	
Trigger Action	note
Action Parameter	ausres29@ev1 Too many failed logins on bonnell
Log File	/usr/lpp/dfsweb/etc/bonnell/trigger.log

Current Threshold Rule Names

Too many failed logins
Reset counter after too many failed logins

Figure 31. DFS Web Manager's Panel for Customizing a Threshold Rule

The DFS Web Manager interface also allows you to see the current databases that list the monitored resources along with their current counter values. This is reached by selecting the **Databases** panel from the left-hand side of the DFS Web Manager screen. The databases are listed on the panel, and from here they can be displayed, have all counters reset or be deleted. An example display is shown in Figure 32 on page 78 for the URLCounter database. There are buttons in the left column that allow you to *delete* records or *reset* counters. Various sort options are provided by pushbuttons on that screen.



Current DFS Web Configuration File Name /usr/lpp/dfsweb/etc/bonnell/dfsweb.cfg
Threshold File Name in Configuration File /usr/lpp/dfsweb/etc/bonnell/dfsweb.thr

URLCounterDatabase	
Status	MONITORED
Number of Records	146
Database Type	dbm
Auto Add	off
Reset Interval	interval
File Name	/usr/lpp/dfsweb/etc/bonnell/dburl

Reset	Delete	#	Resource Name	Counter Value	Last Reset Time
		1	/.../itsc.austin.ibm.com/fs/	1	Friday, 28-Feb-97 13:54:52 CST
		2	/.../itsc.austin.ibm.com/fs/toc.html	1	Friday, 28-Feb-97 13:55:26 CST
		3	/.../itsc.austin.ibm.com/fs/home	1	Friday, 28-Feb-97 13:55:15 CST
		4	/.../itsc.austin.ibm.com/fs/home/	1	Friday, 28-Feb-97 13:55:16 CST
		5	/	0	Monday, 10-Mar-97 11:15:09 CST
		6	//	2	Wednesday, 12-Mar-97 14:13:05 CST
		7	/:/RandomApplet.class	1	Sunday, 09-Mar-97 20:21:48 CST

Figure 32. DFS Web Manager Displaying URLCounter Database

5.4 DFS Access Control

Since DFS ACLs are essential for access control in a DFS Web Secure environment, we will have a closer look at them in this section.

5.4.1 DFS ACL Permissions

DFS ACLs are used to control access to files and directories stored within DFS filesets. In a sense they operate like the file permissions (or mode bits) in a UNIX filesystem (read write and execute). The difference with DFS ACLs is that there is a wider range of permissions that can be granted for files and an even wider range for directories. These permissions can also be specified for a broader range of potential users of the files/directories than just the *owner*, his/her *group* and *others*. Access permissions are granted to classes of potential users (known as *types)* such as the file owner, a specified principal, a named group, principals from another DCE cell, and so forth. The range of permissions that can be granted (or denied) for files and directories is shown in Table 2 on page 79. The range of principal and group *types* to whom these permissions can be granted (or denied) is shown in Table 3 on page 81.

Table 2. Valid DFS ACL Permissions

File Permissions	Directory Permissions
r read **w** write **x** execute **c** control (*the permissions to change the ACLs on a file*)	**r** read **w** write **x** execute **c** control **i** insert **d** delete

Here is a sample ACL for the root directory of DFS in a cell:

```
# SEC_ACL for ./fs:
# Default cell = /.../itsc.austin.ibm.com
mask_obj:rwxcid
user_obj:rwxcid
user:cell_admin:rwxcid
group_obj:rwxcid
other_obj:r-x---
any_other:r-x---
```

The entry `user_obj:rwxcid` shows the permissions granted for the directory owner. Entries are of the form *type*[*:key*]*:permissions* where a dash (-) means the permission is denied. In the example the user entry for `cell_admin` shows the use of a *key*, meaning the user parameter needs to specify the user name. `cell_admin` is granted full privileges on this directory.

5.4.2 DFS ACL Entry Types

The minimum set of ACL entry types that will be seen on any file or directory are:

user_obj Equivalent to the UNIX mode bits for the owner of the file/directory.

group_obj Equivalent to the UNIX mode bits for the owner's group if there is no mask_obj entry. If there is a mask_obj entry, the permissions for group_obj will be defined by the mask_obj entry permissions. More on mask_obj below.

other_obj Equivalent to the UNIX mode bits for other.

Be Aware of the Influence of Masks

Masks establish the maximum permissions that can be granted to a principal or group. They protect the administrator against giving a user or group more permissions than intended.

A mask_obj entry is automatically created when any entry is added to the minimum set of ACLs (user_obj, group_obj and other_obj). The mask_obj can be modified like other ACL entries, but it cannot be deleted if there is a user or group type entry. Because they appear automatically, they can sometimes cause confusion when they restrict permissions that you believe have been explicitly granted to a particular user or group.

If it exists, the mask_obj applies to all entry types except user_obj and other_obj. For example, if you create an entry for `user:jdoe:rw` but the mask_obj gives permissions of `r`, then `user:jdoe` will have effective permissions of only `r`. This is shown when listing the ACL entry (with the command `acl_edit <path> -l`).

Setting mask_obj open (for instance, with all permissions) will prevent it from causing you surprises.

The `umask` setting of a process that is creating files/directories also influences the initial ACL permissions.

Table 3. DFS ACL Entry Types for Principals and Groups

Type	Key	Description and entry format
user_obj owner in AIX	none	Permissions for the object's real or effective user. **user_obj**:*permissions*
group_obj group in AIX	none	Permissions for members of the object's real or effective group. **group_obj:***permissions*
other_obj other in AIX	none	Permissions for all other principals in the local cell, provided they are not named in a **user** entry type and are not a member of a group named in a **group** entry type. **other_obj**:*permissions*
user	principal_name	Permissions for a specific principal in the local cell. **user**:*principal_name*:*permissions*
group	group_name	Permissions for members of a specific group in the local cell **group**:*group_name*:*permissions*
foreign_user	cell_name, principal_name	Permissions for a specific principal in a foreign cell. **foreign_user**:*cell_name/principal_name*:*permissions*
foreign_group	cell_name, group_name	Permissions for a specific group in a foreign cell. **foreign_user**:*cell_name/group*:*permissions*
foreign_other	cell_name	Permissions for a other principals in a foreign cell provided they are not named in a **foreign_user** entry type and are not a member of a group named in a **foreign_group** entry type. **foreign_user**:*cell_name/principal_name*:*permissions*
any_other	none	Permissions for all other principals in foreign cells unless they match a more specific entry type in the ACL. This would also apply to an unauthenticated user. **any_other**:*permsissions*

Types of ACLs: Files have just one type of ACL, the object ACL, which controls access to the file itself.

Directories are both objects and containers of objects and consequently they have three different types of ACL:

Object ACL	Controls access to the directory object itself
Initial Object ACL	The ACL that determines the default protection assigned to an object(file) created within this directory

Initial Container ACL	The ACL that determines the default protection assigned to directory objects created within this directory

It is important to get the Initial Object and Initial Container ACLs set sensibly when you create a new fileset or directory. It will be much more work to rectify problems later when the fileset has been populated with numerous directories and files. The following list shows the factors that determine the ACL that will be inherited by a newly created object:

1. AIX mode bits specified with the `create`, `open` or `mkdir` system call
2. Initial Object ACL from the object's parent directory
3. `umask` of the process creating the object

You can edit any ACL type with the interactive `acl_edit` command from an AIX shell. To edit the Object ACL for a file or directory:

```
# acl_edit <pathname>
```

To edit the Initial Object ACL for a directory:

```
# acl_edit <directory_pathname> -io
```

To edit the Initial Container ACL for a directory:

```
# acl_edit <directory_pathname> -ic
```

Tips for using `acl_edit`:

- Commit changes before leaving `acl_edit` using the `co` subcommand. If you do not do this and there are some errors in your operations, then all your updates will be lost when you exit, and the implicit commit fails.
- Favorite or staple ACL specifications can be placed in a file and used as input to `acl_edit` using its `as` (assign_file) subcommand. All other ACLs will be overwritten in this case.

Changes can be made to ACL entries by using the `chmod` command, which modifies the AIX permission bits. The ACL entries for user_obj, group_obj and other_obj are synchronized with the AIX permissions for owner, group and other as far as `rwx` permissions are concerned.

5.4.3 The Permissions the Users Require

The permissions that a browsing user of DFS Web Secure needs in order to access resources stored within DFS are the same as the permissions required by any DCE user accessing DFS. They are summarized in Table 4 on page 83.

Table 4. File and Directory Operations and Required Permissions within ACLs

Operation	Required Permissions
Change to a directory	`x` on all the directories leading to the directory `x` on the directory itself
List the contents of a directory	`x` on all the directories leading to the directory `r` on the directory itself
List information about objects in a directory	`x` on all the directories leading to the directory `r` and `x` on the directory itself
Read or read lock a file or download a Java applet	`x` on all the directories leading to the file `r` on the file (or Java Class file) itself
Write or write lock a file	`x` on all the directories leading to the file `w` on the file itself
Execute a binary file	`x` on all the directories leading to the file or applet `x` on the file itself
Execute a shell script	`x` on all the directories leading to the script file `r` and `x` on the script file itself
Create an object	`x` on all the directories leading to the directory where the object is to be placed `w`, `x` and `i` on the directory where the object is to be placed
Delete an object	`x` on all the directories leading to the directory from which the object is to be deleted `w`, `x` and `d` on the directory where the object is to be deleted
Rename an object	`x` on all the directories leading to the object's current directory `x` on all the directories leading to the object's new directory `w`, `x` and `d` on the object's current directory `w`, `x` and `i` on the object's new directory
List the ACLs on an object	`x` on all the directories leading to the object
Change the ACLs on an object	`x` on all the directories leading to the object `c` on the object itself

To determine the permissions granted to a user requesting to access a file or directory, the DFS ACL Manager checks the ACLs in the following sequence (shown in detail in Figure 33 on page 84):

1. First *user* entries, then *group* entries, then *other_obj*, then *foreign_other* and finally *any_other*, (most specific to least specific). It stops when it matches an entry. Therefore, if a user is listed both as a user and as a member of a group, the ACL Manager will grant the permissions for the user only; it will never get to the group permissions.

2. The mask_obj is applied, if it is set, to any entry types except user_obj and other_obj. Only the permissions granted in the ACL entry **AND** in the mask are granted.

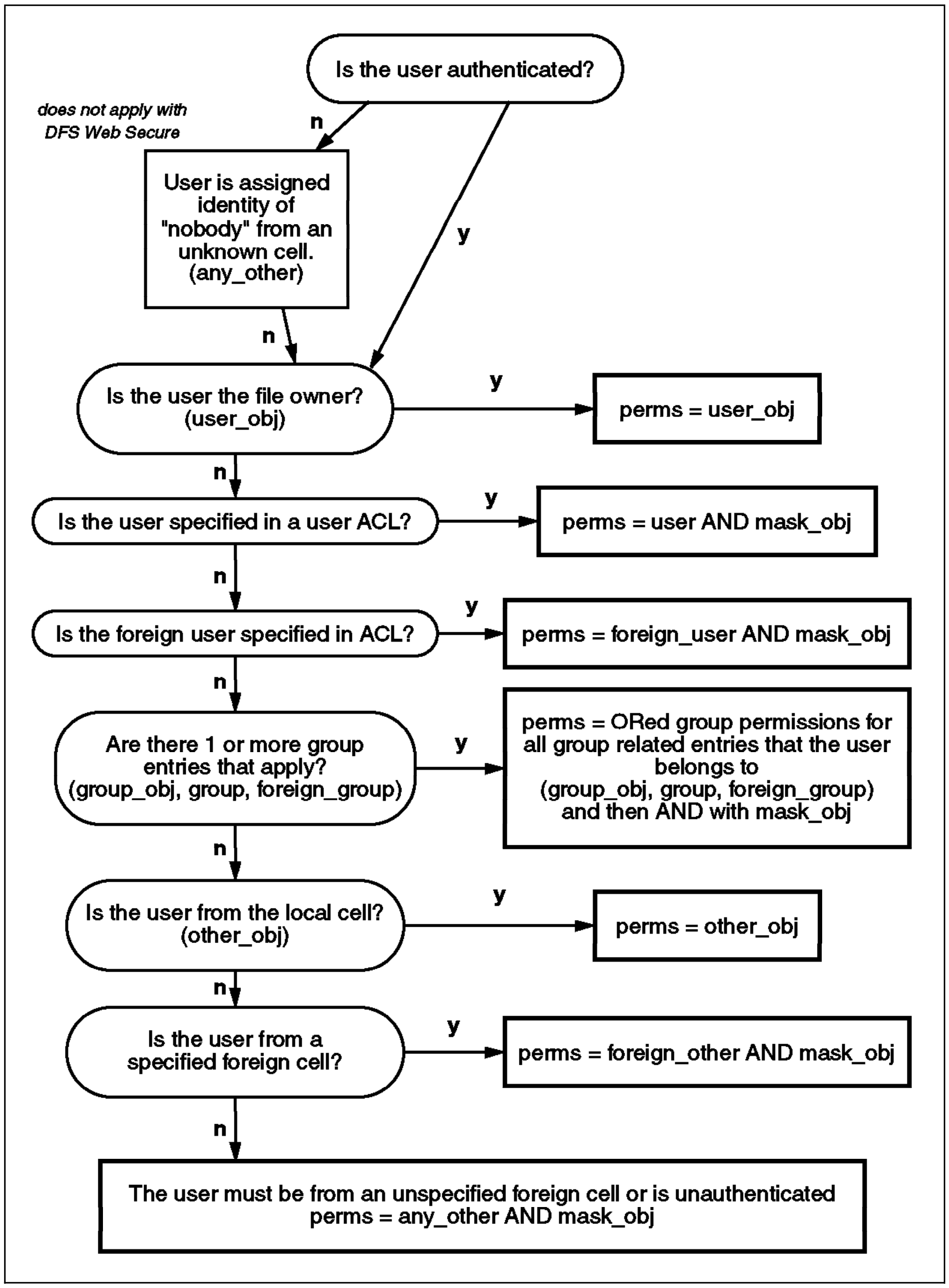

Is the user authenticated?
does not apply with DFS Web Secure
n
User is assigned identity of "nobody" from an unknown cell. (any_other)
y
n
Is the user the file owner? (user_obj)
y
perms = user_obj
n
Is the user specified in a user ACL?
y
perms = user AND mask_obj
n
Is the foreign user specified in ACL?
y
perms = foreign_user AND mask_obj
n
Are there 1 or more group entries that apply? (group_obj, group, foreign_group)
y
perms = ORed group permissions for all group related entries that the user belongs to (group_obj, group, foreign_group) and then AND with mask_obj
n
Is the user from the local cell? (other_obj)
y
perms = other_obj
n
Is the user from a specified foreign cell?
y
perms = foreign_other AND mask_obj
n
The user must be from an unspecified foreign cell or is unauthenticated perms = any_other AND mask_obj

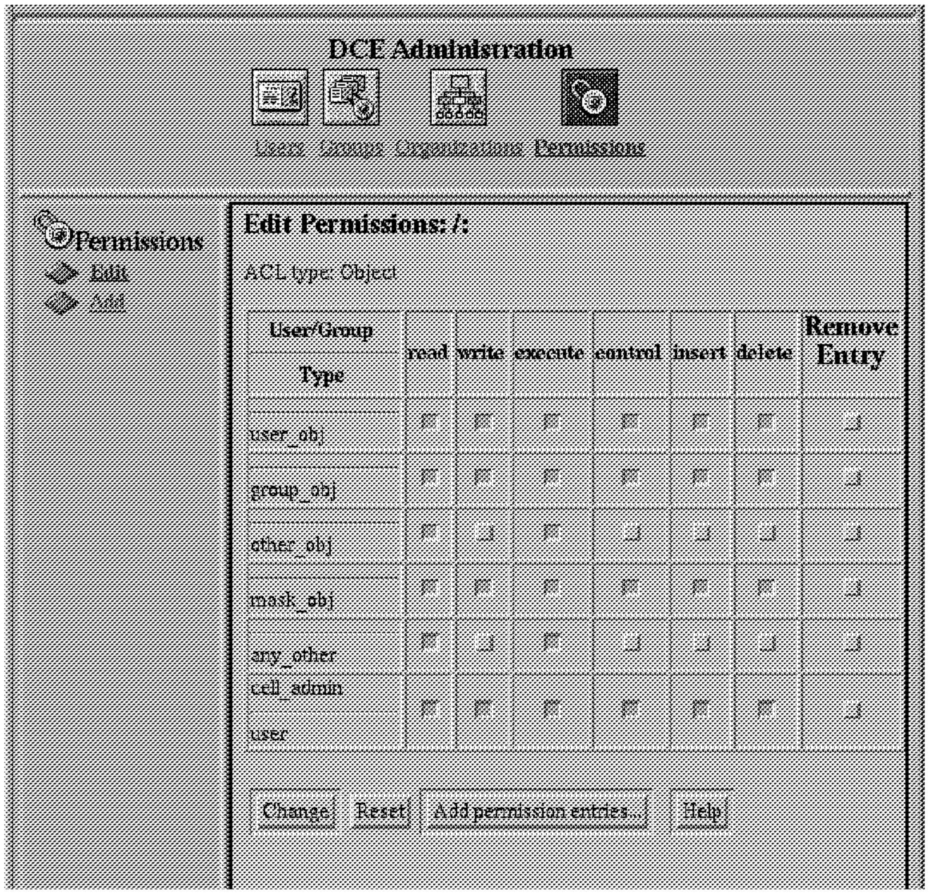
Figure 33. DFS ACL Checking Sequence

In practice, allowing access to resources based on group memberships is probably the best approach, but be aware that denying a named group access to a resource may not necessarily deny all members of that group, since a member of the denied group may also be a member of a permitted group and group memberships are OR'ed together.

It is also important to note that if you want to be absolutely sure that a particular file is protected in a specific way, then you must set an ACL on that file and not rely solely on directory ACLs on the path to that file. This is because it is possible to set up links within the DFS file space that may inadvertently open a less secure path to your secret file.

5.4.4 Managing DFS ACLs with DFS Web Manager

In a scenario with DFS Web Secure installed, it is also possible to use the DFS Web Manager interface from a Web browser to administer file and directory ACLs within DFS. In the panel shown in Figure 34, we see that the full range of permissions can be manipulated for an ACL entry type, in this case for our root fileset (/:). Permissions for further entry types can be added by clicking on the Add permission entries button. Unfortunately, like `acl_edit`, it does not support recursive ACL manipulation.



Edit Permissions: /:

ACL type: Object

User/Group Type	read	write	execute	control	insert	delete	Remove Entry
user_obj	☒	☒	☒	☒	☒	☒	☐
group_obj	☒	☒	☒	☒	☒	☒	☐
other_obj	☒	☐	☒	☐	☐	☐	☐
mask_obj	☒	☒	☒	☒	☒	☒	☐
any_other	☒	☐	☒	☐	☐	☐	☐
cell_admin user	☒	☒	☒	☒	☒	☒	☐

Figure 34. DFS Web Administration GUI Interface Editing DFS ACLs

5.4.5 Recursive ACL Manipulation

Unfortunately, `acl_edit` does not provide the functionality of recursively manipulate ACLs. There is a sample script (`rchacl`) in Appendix B, "Program Listings" on page 159, that can be used to apply ACL modifications recursively to a directory structure. From higher-level directories within a large, established DFS filespace, it should be used with care since the changes will sweep through all subdirectories. It will be useful when you have started to build a new fileset and are perhaps unsure of the defaults that were in force when you created some initial directories or copied some files into the new fileset.

5.4.6 A Practical Example

Here is an example of the use of ACLs and DCE group memberships with a simple DFS filespace to be served by DFS Web Secure for a small organization. The structure of the DFS filespace is shown in Figure 35. There are basically three types of users who will access DFS, and there are corresponding areas in DFS in which they will be allowed to browse:

1. Any user unknown to the organization will be able to authenticate as `anonymous`. We created a public group for this user, and there is an area of DFS where information will be placed for public consumption: /:/data/web/public.
2. Users who have signed up for programs run by the organization will be able to authenticate with DCE as named affiliates and access the relevant program areas of the filespace, here residing in the /:/data/web/club directory as well as the public area. In this example, the organization has two affiliation programs, Silver Club and Gold Club, where only affiliates of the Gold Club will be able to access the newproducts subdirectory.
3. Company employees who will have rights to the public and club directories of /:/data/web as well as a private area containing department and private directories.

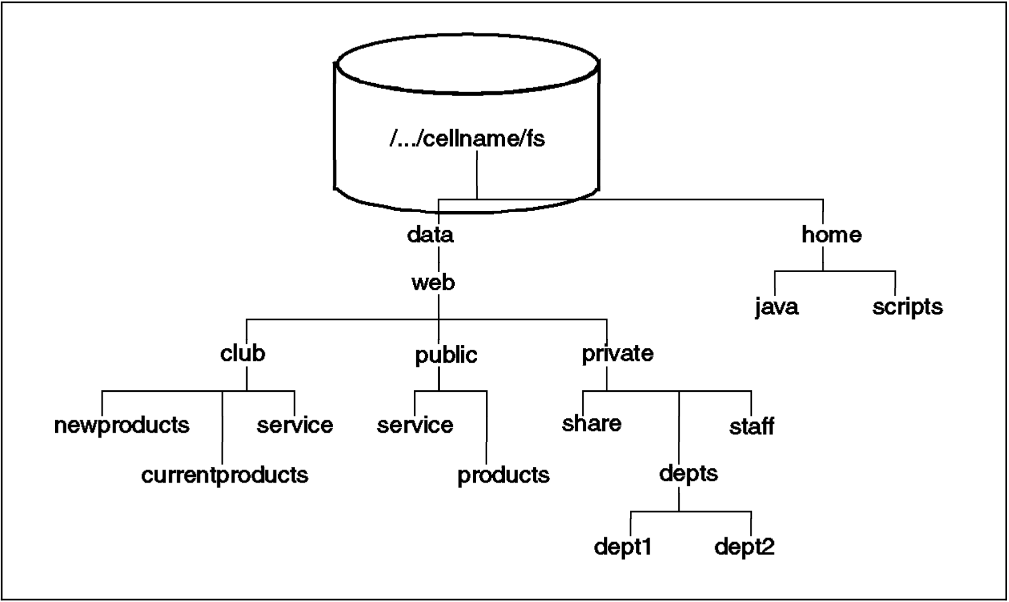


Figure 35. DFS Filespace for ACL example

The DCE users and groups that were set up for this example are shown in the left two columns of Table 5 on page 87. The table also notes the group ACLs that were defined for directories in the filespace to restrict users as outlined above. The following ACL for the /:/data/web/club/newproducts directory shows how the permissions are set up to enforce the required access control:

```
# SEC_ACL for newproducts:
# Default cell = /.../itsc.austin.ibm.com
mask_obj:rwxcid
user_obj:rwxcid
group_obj:rwx-id
group:staff:rwx-id
group:goldclub:r-x---
other_obj:------
```

With other_obj denied any permissions, a user needs to be named in an entry or be a member of a group that is named. Hence, anonymous (in group public, see Table 5 on page 87) cannot access this directory, nor can affiliate2 who is a member of silverclub, which is not named. employ1, employ2 and cio can access the directory (and even modify its contents) as members of the staff group.

In this example there are no permissions defined for other_obj, but in many cases DFS ACLs, influenced by a default umask and Initial Container ACL, will allow other_obj read permission when you are creating directories. This could have the effect of allowing the user anonymous access to more areas of your DFS setup than you want. It is important to set some ACLs at the highest levels of DFS to restrict anonymous to only the areas of DFS that you intend to be ″public″. In this example, this would mean denying other_obj access for directories at the /:, /:/data and /:/data/web levels and then allowing the group public execute (x) permission on those directories as well as read permission on the /:/data/web/public directory, which starts the intended ″public″ area.

Table 5. DCE Registry and ACLs Setup for Sample Scenario

DCE Groups	DCE Users in Group	DFS Directories (allowed by group ACL)	Remarks
public	**anonymous**	**/:/data/web/public**	**also only execute (x) permission on parent directories: /:/data and /:/data/web**
silverclub	**affiliate1 affiliate2**	**/:/data/web/public /:/data/web/club /:/data/web/club/currentproducts /:/data/web/club/service**	
goldclub	**affiliate1**	**/:/data/web/club/newproducts**	
staff	**cio employ1 employ2**	**/:/data/web/private /:/data/web/public /:/data/web/club /:/data/web/club/currentproducts /:/data/web/club/newproducts**	
dept1	**employ1**	**/:/data/web/private/depts/dept1**	
dept2	**employ2**	**/:/data/web/private/depts/dept2**	

Although this example uses the concept of directory-based administration, which is probably the most common way to keep administration efforts low, it is perfectly possible to apply file protection to individual files in a directory tree. This would, for example, happen if users have their home directories in DFS and want to grant or deny access to other users or groups on an individual file basis.

5.5 Running Applications with DFS Web Secure

Web sites have developed from being structures housing static HTML documents to being dynamic showcases that can present users a range of active controls to drive and interact with live applications. There are a number of architectures available for building these applications and making them available on a Web site. With DFS Web Secure installed on a Web server, it is possible to make use of DFS as the store for these applications. In fact, in most cases, placing these applications within DFS means that they can be brought under the control of DCE security without requiring any modifications to the application code.

There is another interesting point regarding DCE credentials when accessing an application through DFS Web Secure. Having authenticated with the DCE registry through the browser when first contacting a page served by DFS Web Secure, a server program subsequently executed by that browser user will inherit those same DCE credentials.

This section looks at the major types of Web applications and how they can be used in conjunction with DFS Web Secure.

5.5.1 CGI Applications

The Common Gateway Interface (known as CGI) is a standard for interfacing external applications with Web servers. This standard is widely used for making live applications available on the Web, and it continues to be popular because CGI programs can be written in a number of languages including:

- C/C++
- UNIX shell script
- Perl
- TCL
- AppleScript, and others

A CGI program is basically an executable that will run on the Web server machine and will (hopefully) send a response back to the Web browser. The CGI program is called by a reference to its URL within a Web page (or HTML document) that the browser is viewing. Parameters from the Web page can be passed to the CGI program through environment variables.

A common example is forms-handling, where a Web page of data entry fields is filled in on the browser and then the user clicks the **Submit** button. This event causes the Web server to call the CGI program in real time, passing the data from the form within an environment variable. The CGI program can process the data and build a response, in a range of formats including HTML, to be passed back to the Web browser through the Web server. This basic sequence of steps can be seen (labelled 1 to 4) in Figure 36 on page 89. The steps are:

1. The Browser requests the Web form through the Web server. The form is completed by the user who then clicks on the **Submit** button.
2. The Web server calls the CGI program referenced in the HTML form passing the form data in an environment variable.
3. The CGI Program executes on the Web server system and builds a response as HTML. With the CGI program residing in DFS, this will be subject to the DFS ACLs on the CGI program file.
4. The HTML output is returned to the Browser via the Web server.

The figure is also depicting two alternative locations for HTML forms and CGI programs and hence the alternative path (step 1a replacing step 1). The location of CGI programs must be configured within the Web server setup so that the server knows to execute the CGI program file instead of displaying it. For security reasons, it is also important that these locations are restricted—after all, you are potentially allowing anyone to execute the files located in them. The figure shows HTML and CGI program files stored in both the Web server file area and in DFS.

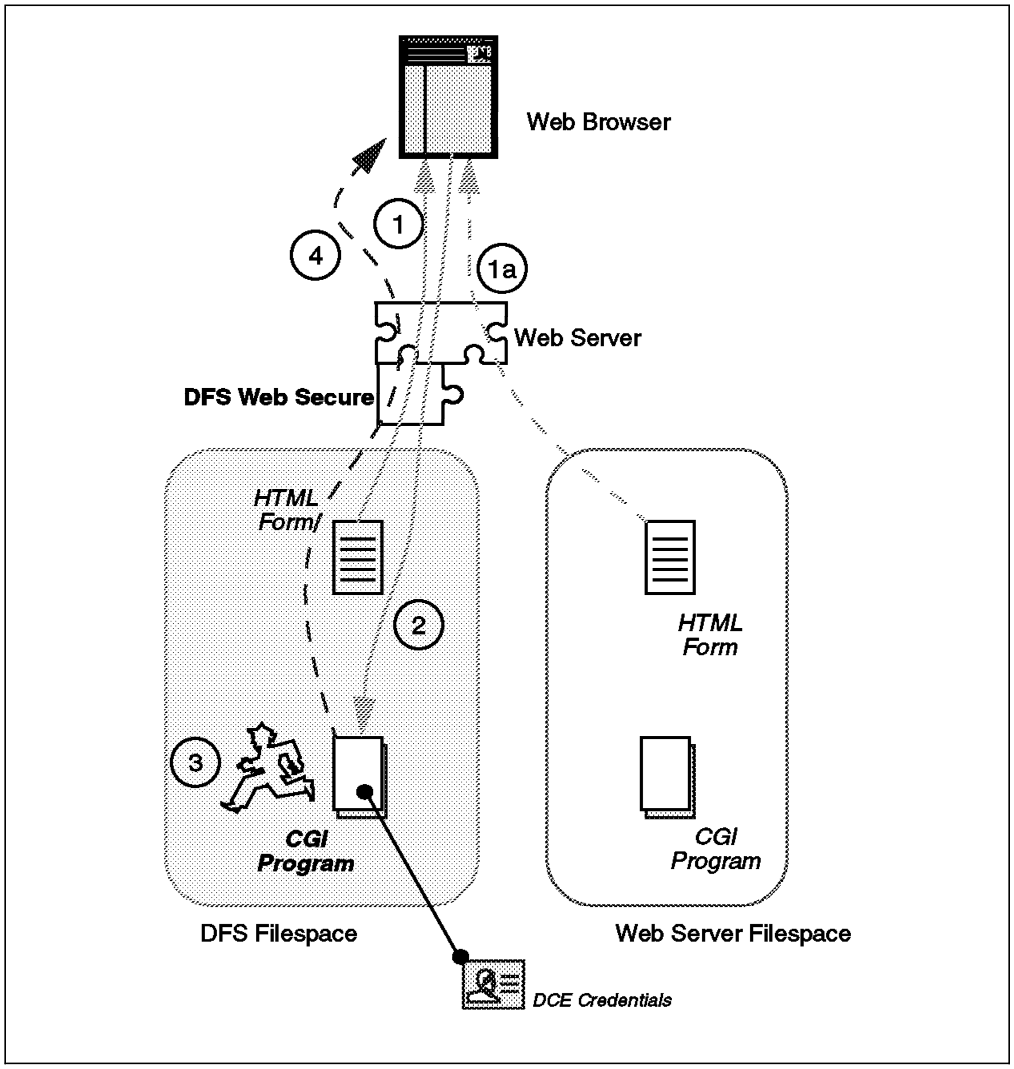


Figure 36. Running a CGI Program with DFS Web Secure

With DFS Web Secure installed and involved in serving the files stored in DFS, what difference does this make for the execution of CGI programs under DFS? CGI Programs do not need to be specially modified in order to run under DFS. Here are some additional possibilities that are made available by using DFS Web Secure:

- Placing CGI programs within DFS puts them under the control of DCE security, which is applied according to DFS ACLs defined at both file and directory levels. This can also be done for the HTML source files for the Web pages and forms that launch the CGI programs (but it is not essential to place them within DFS). This is why the alternative path *1a* is shown in Figure 36, since security based on DCE and a DFS ACL will still be applied to the user trying to run the CGI program even if the form referencing the CGI program is not stored within DFS. The corollary of this is also true: for a form within DFS referencing a CGI program outside DFS, the user must authenticate with DCE to reach the form.

In a sense, you can bring CGI programs that are not DCE aware and place them under the control of DCE security. No programming changes are required.

- The DCE credentials of the authenticated DCE user passing through DFS Web Secure are available to any CGI program residing in DFS launched by that browser user. This can be very useful where you have CGI programs that users want to use to update files in protected areas of DFS. The CGI program does not have be built with a series of DCE security API calls to establish a login context with DCE before it can access DFS. The login context will be available to it automatically through the use of DFS Web Secure.

  This capability for CGI programs to inherit DCE credentials can also be extremely useful for building CGI programs to access native DCE applications and servers. This use of CGI will be covered further in 5.5.4, “Native DCE Applications” on page 97.

A CGI Example with Inherited DCE Credentials: Let’s look at an example showing the key elements of using CGI and also this capability of DFS Web Secure to provide inheritance of DCE credentials.

1. First of all, it is necessary to inform the Web server about the intended location of CGI programs, in this case a location within DFS. For our Web server we configured the URL of `cgi-bin` (relative to the Web server’s root URL) to be mapped to our CGI programs directory /:/home/cgi-programs in DFS. With the Netscape FastTrack Server that we used for testing, this is configured with the Administration interface through the following sequence of panels:

   - From the **Server Selector** panel, click on the desired server name.
   - From the **FastTrack Server** panel, click on *Programs* at the top of the screen.
   - Select the **CGI Directory** panel from the left of the screen.

   Enter details of the URL prefix and the CGI location directory and click **OK**. Note that the URL prefix does not need to contain any element of a DFS path (`/:`, `/...`, `/.:`, or /fs). DFS Web Secure will be aware of any request through this URL as long as the CGI directory path is in DFS. Further down this panel, you may notice existing CGI directory entries that have been added by DFS Web Secure for its own administration, which is driven through the DFS Web Manager interface.

2. Write the CGI program and place it in the CGI program directory under an appropriate ACL to allow execution by relevant DCE users and groups. Here is a very simple sample written in Korn shell script that generates an HTML response containing the system date and shows the DCE credentials of the process running the script (using the `klist` command).

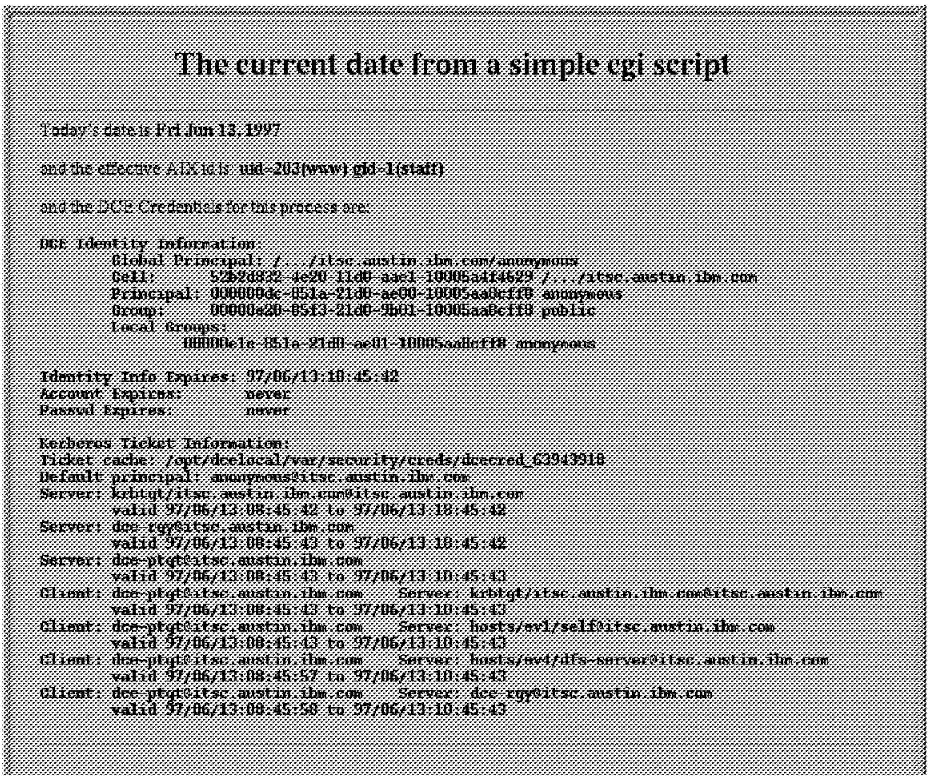
   ```
   #!/usr/bin/ksh
   # file: simple_cgi.cgi
   # Print out necessary header and empty line
   echo ″Content-type: text/html″
   echo ″″
   ```

```
# Print out a title and text. Note the HTML tags in the strings.
echo ″<TITLE>The current date!</TITLE>″
echo ″″
echo ″<H1 align=center>The current date from a simple cgi script</H1>″
echo ″<hr>″
echo ″″
echo ″Today′s date is <b>″

# Echo the date
date +″%a %b %e, %Y″
echo ″</b><p>″
# Echo the uid
echo ″and the effective AIX id is:<b>″
id
echo ″</b></p>″
# and the DCE credentials also
echo ″<p>and the DCE Credentials for this process are:<b></p><pre>″
klist
echo ″</pre>″

# Clean up the HTML page
echo ″</b>″
echo ″<hr>″
```

3. Include a reference to the CGI program in an HTML document or form. Here is a minimal sample HTML document that has a suitable reference to execute the example script, simple_cgi.cgi:

```
<HTML>

<HEAD>
<TITLE>CGI Script Testing Form</TITLE>
</HEAD>

<BODY>
<BR>
<H1 align=center>CGI Program Testing Page</H1>

Call a simple cgi script stored under DFS -
<A href=″/cgi-bin/simple_cgi.cgi″>Simple script</a> <P>

</BODY>
</HTML>
```

To run the example CGI script, the browser user browses the HTML file above (which can be placed within or outside DFS) and clicks on the highlighted reference, **Simple Script**. The output is shown in Figure 37 on page 92, which reflects the DCE credentials of the browser user, in this case anonymous.



The current date from a simple cgi script

Today's date is **Fri Jun 13, 1997**

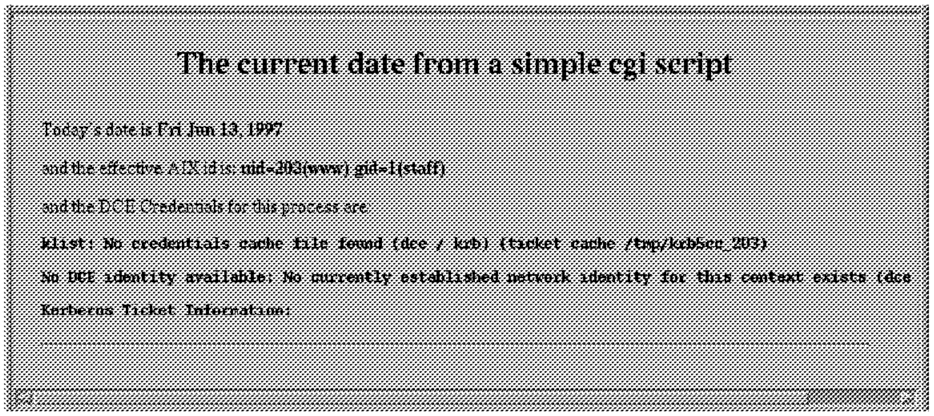
and the effective AIX id is: **uid=203(www) gid=1(staff)**

and the DCE Credentials for this process are:

```
DCE Identity Information:
        Global Principal: /.../itsc.austin.ibm.com/anonymous
        Cell:        52b2d832-4e20-11d0-aae1-10005a4f4629 /.../itsc.austin.ibm.com
        Principal: 000000dc-851a-21d0-ae00-10005aa8cff8 anonymous
        Group:       00000a20-85f3-21d0-9b01-10005aa8cff8 public
        Local Groups:
                000000e1e-851a-21d0-ae01-10005aa8cff8 anonymous

Identity Info Expires: 97/06/13:18:45:42
Account Expires:        never
Passwd Expires:         never

Kerberos Ticket Information:
Ticket cache: /opt/dcelocal/var/security/creds/dcecred_63943918
Default principal: anonymous@itsc.austin.ibm.com
Server: krbtgt/itsc.austin.ibm.com@itsc.austin.ibm.com
        valid 97/06/13:08:45:42 to 97/06/13:18:45:42
Server: dce-rgy@itsc.austin.ibm.com
        valid 97/06/13:08:45:43 to 97/06/13:18:45:42
Server: dce-ptgt@itsc.austin.ibm.com
        valid 97/06/13:08:45:43 to 97/06/13:10:45:43
Client: dce-ptgt@itsc.austin.ibm.com    Server: krbtgt/itsc.austin.ibm.com@itsc.austin.ibm.com
        valid 97/06/13:08:45:43 to 97/06/13:10:45:43
Client: dce-ptgt@itsc.austin.ibm.com    Server: hosts/ev1/self@itsc.austin.ibm.com
        valid 97/06/13:08:45:43 to 97/06/13:10:45:43
Client: dce-ptgt@itsc.austin.ibm.com    Server: hosts/ev4/dfs-server@itsc.austin.ibm.com
        valid 97/06/13:08:45:57 to 97/06/13:10:45:43
Client: dce-ptgt@itsc.austin.ibm.com    Server: dce-rgy@itsc.austin.ibm.com
        valid 97/06/13:08:45:58 to 97/06/13:10:45:43
```

Figure 37. Sample CGI Script Output with DCE Credentials

If the same CGI script is placed outside DFS, then the output (shown in Figure 38) reflects the fact that the script is running without DCE credentials. This is the case even if the browser user has previously authenticated with DCE for a URL served by DFS Web Secure. If the path to the CGI program does not feature DFS in either the URL or in the corresponding directory location, then DFS Web Secure will not triggered to make the DCE credentials available to the CGI program.



The current date from a simple cgi script

Today's date is Fri Jun 13, 1997

and the effective AIX id is: **uid=203(www) gid=1(staff)**

and the DCE Credentials for this process are:

```
klist: No credentials cache file found (dce / krb) (ticket cache /tmp/krb5cc_203)

No DCE identity available: No currently established network identity for this context exists (dce

Kerberos Ticket Information:
```

Figure 38. Sample CGI Script Output without DCE Credentials

Further examples of CGI programming using alternative languages and including handling of a form are included in 5.5.4, "Native DCE Applications" on page 97.

5.5.2 Java Applications

Java is a technology developed more recently than CGI that can be used to make Web pages dynamic in appearance and allow browser users to interact with live programs. Java is also a language for developing object-oriented applications, and in the context of the WWW, the most common type of Java application is called an *applet*. A fundamental difference between Java applets and CGI programs is the location of the program's execution. Where CGI invokes an application on the Web server, a Java applet is downloaded over the network to the browser system and there it is run within the browser itself. This distinction is becoming blurred as the Java architecture is being extended to allow server-side execution of Java programs; so here we look at the use of Java applications with DFS Web Secure in both cases.

5.5.2.1 Client-Side Java Applications

The typical flow of information when a browser user accesses a new Java applet is shown in Figure 39 on page 94. In this case we are showing the browser accessing a Java applet that is residing in DFS. The key steps labelled 1 to 4 represent the following interactions:

1. From the Web server that is either stored within DFS (step 1) or on another filesystem accessible by the server (step 1a), the browser gets a Web page.

   For path 1a to work when an HTML file outside DFS is pointing to Java files stored in DFS (using the `codebase` parameter within the `applet` HTML tag), the user must have already passed through DFS Web Secure and authenticated with DCE. If authentication has not already been done, DFS Web Secure will not prompt you based on the `codebase` parameter in the HTML referring to a location within DFS. To avoid this possibility, it would be good practice to store the HTML files, as well as the Java files, in DFS.
2. The Web page contains an `applet` reference to a Java applet at a URL located within the Web server, in this case within DFS.
3. The Java applet file (and Java class files referenced by the applet) are passed back to the browser by the Web server. With the Java files residing in DFS this will be subject to the DFS ACLs on the Java files. The DCE identity that the browser user has established through DFS Web Secure will be used to determine access rights that need to be at least read (`r`) to allow the Java files to be downloaded.
4. The Java applet is executed within the browser's Java Virtual Machine. .

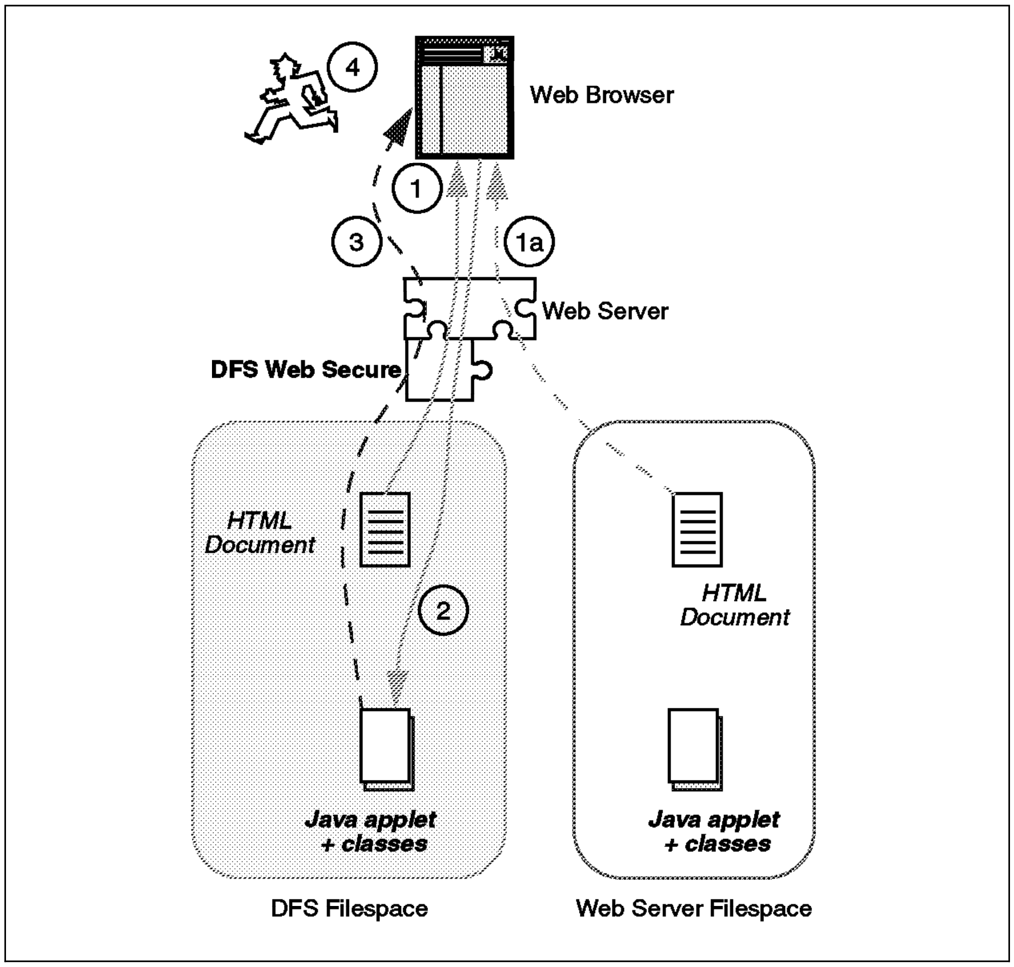


Figure 39. Running a Java Applet with DFS Web Secure

There are no configuration changes that need to be made to the Web server to enable the distribution of Java applets and Java class files to browsers. Java files are treated like ordinary HTML files and do not have to reside in special directories that are configured in the Web server. The difference when storing them inside DFS and using DFS Web Secure is that access control is based on DCE credentials and DFS ACLs. Read (r) permission on a Java class file and execute (x) permissions on the directories leading to the file will be sufficient for a client browser to be able to download the file.

Securing Java applets and class files by placing them within DFS does not require any modification of the Java code itself. Here is an example of an HTML reference to a Java applet that is stored in DFS:

```
<applet codebase=/:/home/java code=BillingView.class width=540 height=320>
</applet>
```

The *codebase* attribute is directing the Web server to find the Java applet `BillingView` in the DFS directory `/:/home/java`.

5.5.2.2 Server-Side Java Applications

The Java landscape is evolving quickly, and programmers want to have the flexibility to run Java programs on the server as well as on the client. The Java API is being extended in this direction with a server API. Web server producers are also building the capabilities into their products to run Java applets on the Web server. For instance, the Java interpreter can be enabled within the Netscape FastTrack Server, allowing applications and applets written in Java to execute on the server system. In this way, the architecture shown previously with CGI programs executing on the server can now be achieved for Java programs on the server. In this case, HTML output is generated by the Java application and is passed back to the browser; the Java applet and Java class files are not passed over the network to the browser. This flow is shown in Figure 40.

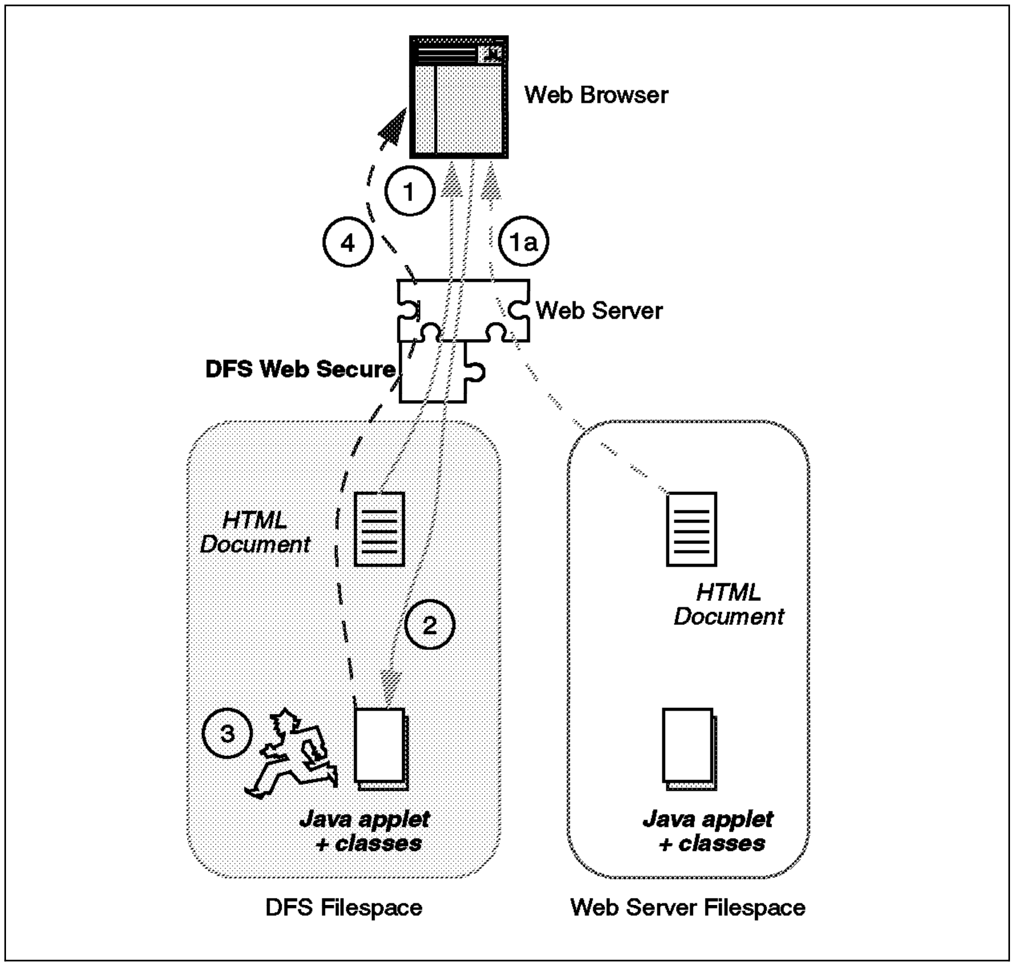


Figure 40. Running Java on the Server with DFS Web Secure

The flow of execution is:

1. The browser gets a Web page from the Web server that is either stored within DFS (step 1) or on another filesystem accessible by the server (step 1a).
2. The Web page contains an HTML reference to a Java applet at a URL located within the Web server, in this case within DFS. The Web server may also be configured to know that the location is a valid one for server-side Java applets.

3. The Java applet file (and Java class files referenced by the applet) are loaded and run by the Java interpreter in the Web server. With the Java files residing in DFS this will be subject to the DFS ACLs on the Java files.
4. Any HTML output generated by the Java applet is passed back to the browser.

The Java interpreter has to be enabled within the Netscape server, and the directory containing applets must also be configured within the Netscape server. A Java program in that directory can then be launched by an HTML document with a straightforward reference to a URL /server-java/<java_pgm_name>. For the Netscape FastTrack Server this configuration is carried out under the Programs panels that can be reached from the Server Selector screen in the Administration Interface. Having selected the **Programs** panel, an administrator can then select the **Java** panel where this configuration can be done.

Server-side Java programs can be placed in an applet directory in DFS that is configured within the Netscape server as described above. However, there is a restriction. This directory must be open for unauthenticated access because the DFS Web Secure plug-in is not made aware when a server-side applet in DFS is being called. Therefore, the DCE credentials are not made available to the Web server process that goes to execute the applet. This means that the ACL entry type any_other (used for unauthenticated users) must have x permission granted for the server-side applet directory (and the directories leading to it) as well as r and x permissions on the applet files themselves.

Here is an example of an HTML form that calls a server side Java Applet when the user clicks the **Submit** button.

```
<html> <head>
<title>A Server side Java Test Form</title>
</head>
<body>
<h1>A Java server test</h1>

<form action="/server-java/RandomApplet/ns-icons" method="post">
<input type="HIDDEN" name="success" value="thanks.html">
<p>
<input type="RESET" value="Clear">
<input type="SUBMIT" value="Submit">
</form>
<hr>
</body> </html>
```

The form tag that invokes the applet RandomApplet has the same format as one invoking a CGI program. The flow of execution shown in the figure is also the same as for CGI programs.

A simple href HTML tag can also be used to invoke a server-side Java applet without passing any parameters, as shown here:

```
<A href="/server-java/RandomApplet/ns-icons">RandomApplet</a> <P>
```

5.5.3 JavaScript Applications

JavaScript is a compact, object-oriented scripting language created by Netscape as a way to make Web pages interactive. Unlike Java, which also enables interactive Web pages, JavaScript is an interpreted language, and it is generally used for doing smaller interactive tasks in the browser such as dialogs, scrolling text and validating input to entry fields. A server-side form of JavaScript is also available in a separate product from Netscape called LiveWire. LiveWire JavaScript applications are compiled into bytecode executable files and are similar to CGI programs, executing on the Web server (requiring a LiveWire extension).

Most JavaScript is embedded within HTML files and is downloaded to a browser where it is interpreted and run. Since it is embedded in HTML files, there is no requirement to alter the Web server configuration to handle JavaScript (the requirement for JavaScript support is actually on the browser side). This also means that embedded JavaScript can be placed in DFS and served through DFS Web Secure, subject to DFS ACLs on those HTML files and the directories leading to them.

The execution model for JavaScript application is similar to the one for client side Java applications, shown in Figure 39 on page 94. The only difference is that the JavaScript code is embedded within the HTML document, rather than in a separate file.

5.5.4 Native DCE Applications

Native DCE applications are built with a client/server architecture using remote procedure calls (RPCs) to communicate between client programs and server programs. The application developer can make use of a wide range of DCE services, such as security, naming, and time, within an application that is usually built using the C language. With DCE available for a wide range of operating system platforms, the application developer is also protected from having to worry about the native differences between these machine architectures.

A Note about DCE Application Security

DCE applications can be built with a wide range of security models. DCE provides a number of choices for the application developer in terms of the security that will be applied to each RPC call between the client and the server parts of the application. This involves choosing the RPC protection level.

At the application level, the developer is free to choose the basis on which a server program will allow clients to obtain its services. In many cases a developer will also want to choose the basis on which a client program will check whether it is communicating with a bone fide server program. The developer can make use of information that is contained in the RPC calls to make checks with the DCE security registry or the Cell Directory Service before permitting access. The developer can build an application that makes no security checks and is open to any client access, or the developer can authorize a connection based on some other criteria not directly related to DCE. Writing application ACL managers is an approach sometimes used.

Although the examples in this chapter use a couple of security models that make use of the DCE security, it is not a restriction imposed by the presence of DFS Web Secure.

With the increasing popularity of the Web browser as a client user interface, the question arises of how to set up access to DCE server applications through a Web page. This is particularly relevant to a business with an established set of native DCE applications looking to make those services available to a global audience (whether Internet or intranet based) using popular Web browsers.

There are a number of approaches that can be used to build Web browser access to DCE application servers. Here is a selection:

- Write a CGI program that will launch or access a DCE client application on the Web server system. The DCE client can make RPC calls to a DCE server, and the output will need to be passed back through the calling CGI program and the Web server to be presented on the browser. This approach is shown in Figure 41 on page 99 where the CGI program happens to be placed within DFS. The CGI program could be written in a range of languages, but it must support the ability to call another application, in this case the DCE client (typically compiled C or C++).

  It is not essential for the CGI program to be placed in DFS, but if it is served by DFS Web Secure, there is the advantage of having the browser user's DCE credentials available to the CGI program. Without DFS Web Secure the CGI program will have to establish DCE login credentials if the DCE client program requires them in order to run. Not all DCE clients require a DCE login context in order to run. In fact, there are a wide range of DCE security levels that can be used on both the client-application side and on the server-application side.

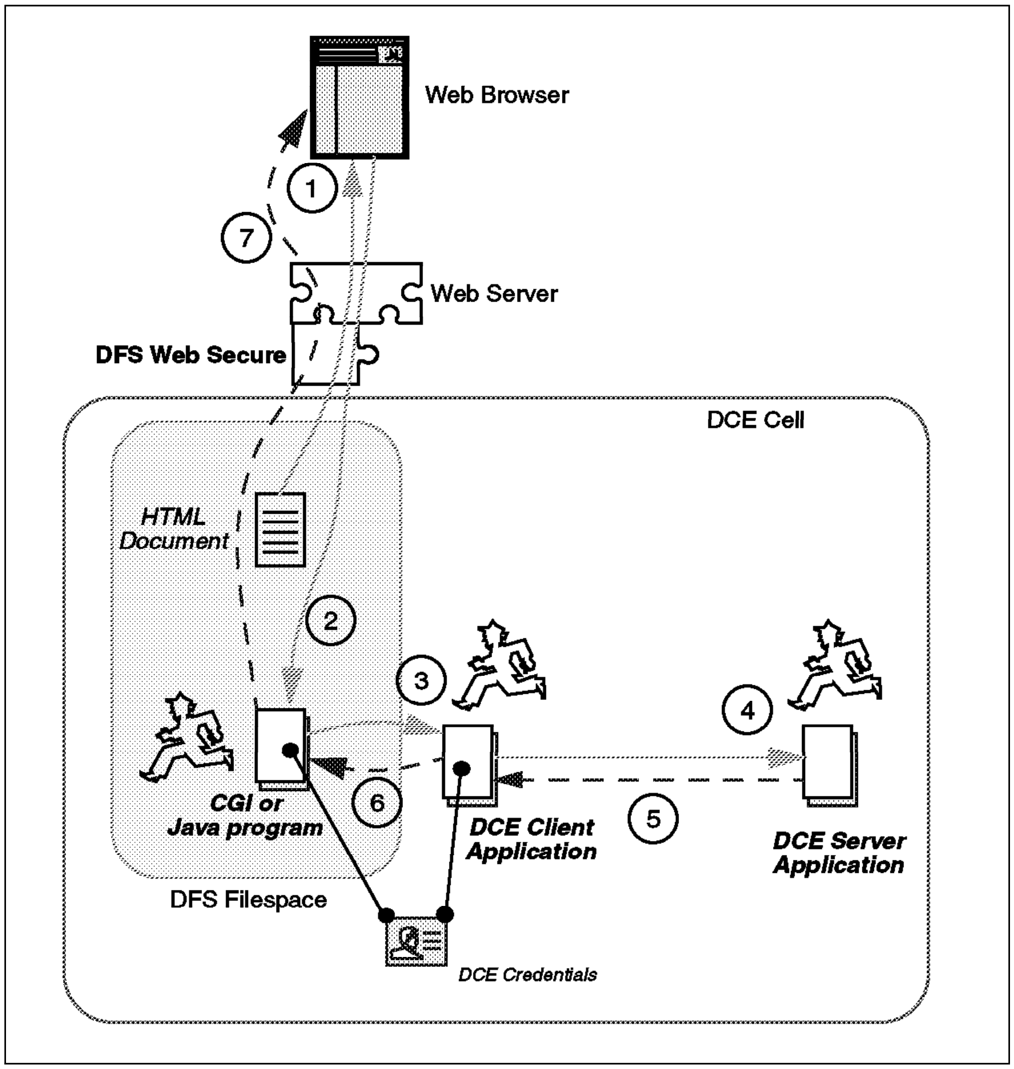


Figure 41. Accessing a Native DCE Application with DFS Web Secure

- Write a hybrid CGI/DCE client program that handles both CGI and includes RPC calls to a DCE server application. This approach is shown in Figure 42 on page 100, and an example of this type written in C is covered in "An Example of a Combined CGI/DCE Client Program" on page 103. Once again placing this hybrid program within DFS and using DFS Web Secure provides a convenient way for the hybrid program to run with the user's DCE credentials. The figure represents a scenario where the CGI/DCE client program is placed in DFS and where DCE login credentials are available to the CGI/DCE client program through DFS Web Secure.

  If the CGI/DCE client program is run from a location outside DFS, then it will have to obtain a login context (DCE credentials) through DCE API calls, if one is required by the security model used in the DCE application.

- Write a Java applet that can be downloaded to the Web browser to run there invoking RPC calls on the DCE server. It is not quite as straightforward as that because a Java applet is restricted in terms of the networks and hosts it is allowed to contact. Actually, a gateway component is required on the Web server that the Java applet is allowed to communicate with, and this gateway will make the RPC calls to the DCE server application. The gateway component will also need to communicate with DCE to establish the required security context on behalf of the Java applet in the browser. This is the basic approach that the DE-Light product from Transarc Corp. supports, providing the required gateway component and tools to help construct the Java applets which will be initiating the RPC requests at the browser. DE-Light and this approach to accessing native DCE applications is covered in detail in Chapter 6, “DE-Light Web Client” on page 109.

- Use a Browser plug-in that can communicate with DCE client services also installed on the browser system to establish DCE credentials and invoke RPC calls on DCE servers. This approach is not investigated any further in this book.
- With Lotus Domino as the Web server and the Distributed Systems Series Link (DSSL) for Lotus Domino toolkit, it is possible to write applications for Lotus Notes/Domino that can access native DCE applications. With the Domino server, these applications can be driven by users from their standard Web browsers. This approach is covered in 7.3, “Distributed Systems Series Link for Lotus Domino” on page 137.

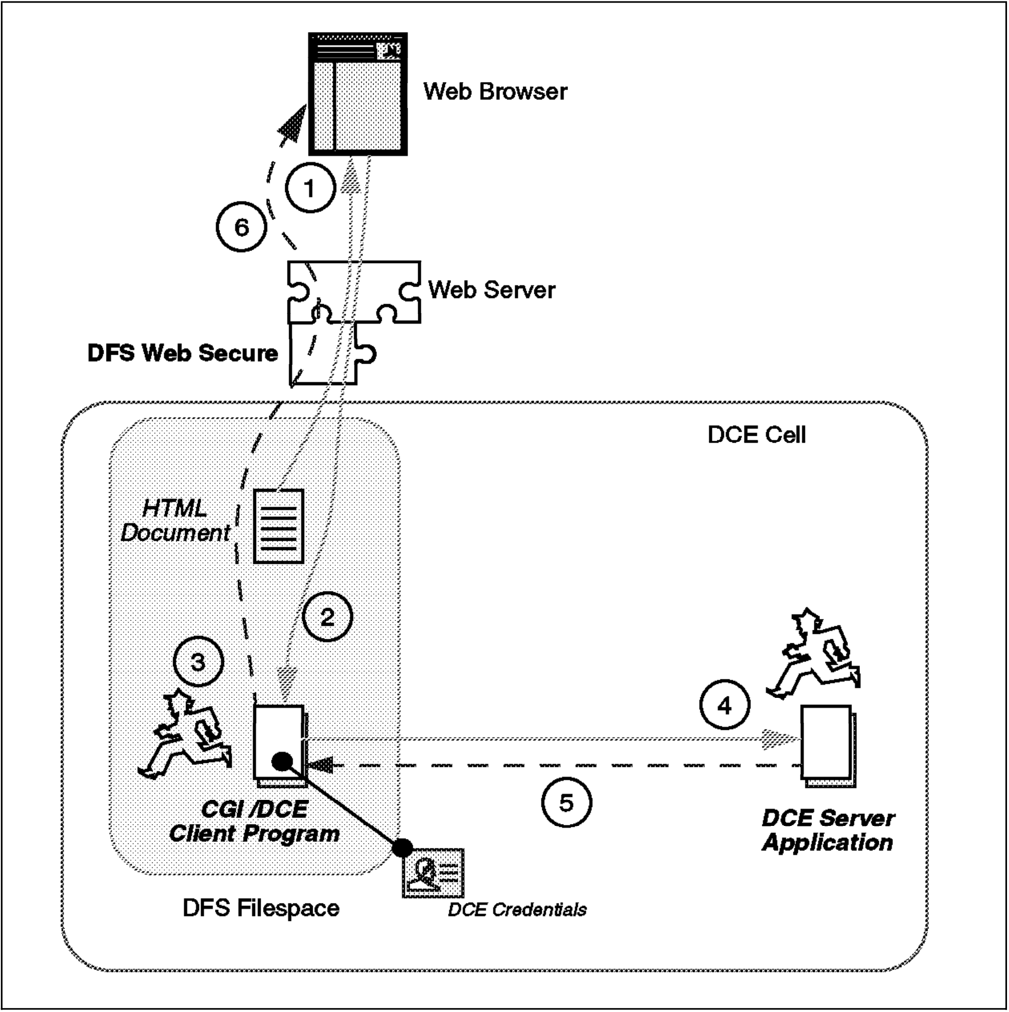


Figure 42. Hybrid CGI /DCE Client Program accessing a Native DCE Application

An Example CGI Script Launching a Native DCE Application: The native DCE application for this example has a simple DCE math server that provides the following interface (shown here in interface definition language) for adding two numbers together.

```
/* Filename: math.idl */

[
uuid(00602160-2F96-1B81-846A-10005A8D5B0E),
version(1.0)
]
interface ServerID
{

/* principal name used by server                                    */
const char  *SERVER_NAME    = "ServerName";

/* server entry in CDS namespace                                    */
const char  *ENTRY_NAME     = "/.:/Servers/SecurityServerID";

/*
*  add operation returns 0 if and only if client is authorized;
*  otherwise, returns -1
*/
long int add(
    [ in ]  handle_t    bh,
    [ in ]  long int    a,
    [ in ]  long int    b,
    [ out]  long        *result);
}
```

An outline of the main functions within the programs for this example is shown in Figure 43

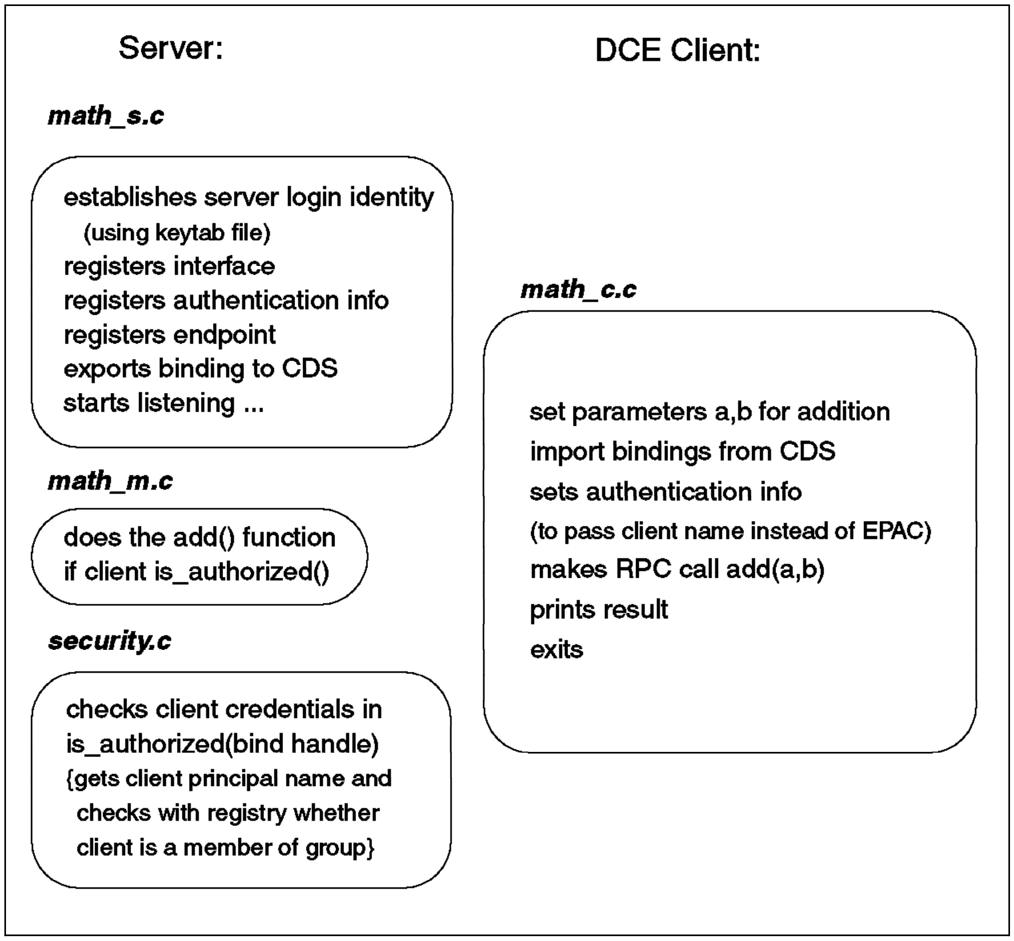

Server:
DCE Client:
math_s.c
establishes server login identity
(using keytab file)
registers interface
registers authentication info
registers endpoint
exports binding to CDS
starts listening ...
math_m.c
does the add() function
if client is_authorized()
security.c
checks client credentials in
is_authorized(bind handle)
{gets client principal name and
checks with registry whether
client is a member of group}
math_c.c
set parameters a,b for addition
import bindings from CDS
sets authentication info
(to pass client name instead of EPAC)
makes RPC call add(a,b)
prints result
exits

Figure 43. Outline of Code in CGI Script Calling DCE Client Example

Here is a CGI program written in Perl script, a popular language for writing CGI programs, which launches the example DCE Client program (math_c). This DCE client program can be started from the command line, takes no arguments and returns output to standard out. This Perl script effectively calls math_c and displays the output from it within the do_work subroutine.

```
#!/:/tools/bin/perl

use CGI::Form;

$query = new CGI::Form;

print $query->header;
print $query->start_html("The calculator results");

# &do_debug($query);

&do_work("/:/home/cgi-programs/math_c");

&print_tail;

print $query->end_html;

sub do_work {
    my($name) = @_;
    print "<H3>Output from math_c</H3><P>\n";
    $res = open(FOO,"$name|");
    while (<FOO>) {
        print $_."<BR>\n";
    }
    close(FOO);
    print "<P>\n";
}

sub do_debug {
    my($query) = @_;
    my(@values,$key);
    print "<H3>The current parameters to this form</H3>";
    foreach $key ($query->param) {
        print "<STRONG>$key</STRONG> -> ";
        @values = $query->param($key);
        print join(", ",@values),"<BR>\n";
    }
    print "<P>\n";
    print "<H3>Environment variables</H3><P>\n";
    foreach $key (sort keys %ENV) {
        print $key, " = ", $ENV{$key}, "<BR>\n";
    }
    print "<P>\n";
}

sub print_tail {
    print <<END;
<HR>
<A HREF="/:/data/web/club/service/calc-pl..html">Back to the form</A>
END
}
```

The do_debug subroutine (which is commented out) allows you to see the full set of parameters passed through CGI and the environment variables that are set.

The DCE client, math_c, is a simple DCE client that binds to a DCE math server and carries out a simple *add* of two numbers with a single RPC call before terminating. This sample DCE math application is used again in the next sample and the full source code is included in Appendix B, "Program Listings" on page 159. In terms of security, math_s checks to see that the client calling it has passed client name information (based on the client's DCE credentials) with the RPC, and it then checks the group memberships of that client principal name in the DCE registry. If the client principal is a member of the good_guys group, then the add() RPC call is completed by the math_s server. This simple DCE client was not specially coded to work within a CGI/Web server environment.

To run the example, the Perl script is placed in the cgi-bin directory (located within DFS) that has been configured in the Web server. The DCE client program, math_c, happens to be in the same directory, but does not have to be there, or even inside DFS. When the CGI program is invoked by the action reference in an HTML form, DFS Web Secure checks if the user has authenticated with DCE before the Perl script can be executed. When authentication has taken place, the Perl script calls math_c, and the DCE responses (to stdout) are passed back to the browser user by the Perl script.

The source code of math_c and a Makefile with details of how to build it are included in Appendix B, "Program Listings" on page 159.

An Example of a Combined CGI/DCE Client Program: This example will use the same math server program. To the sample client program, we add a header file of functions that allow the C client program to handle incoming CGI parameters and send HTML output. An outline of the main functions within the programs for this example is shown in Figure 44. The CGI functions are shown in bold in the figure and are included in the header file cgifns.h. Code listings for the programs listed and cgifns.h are provided in Appendix B, "Program Listings" on page 159.

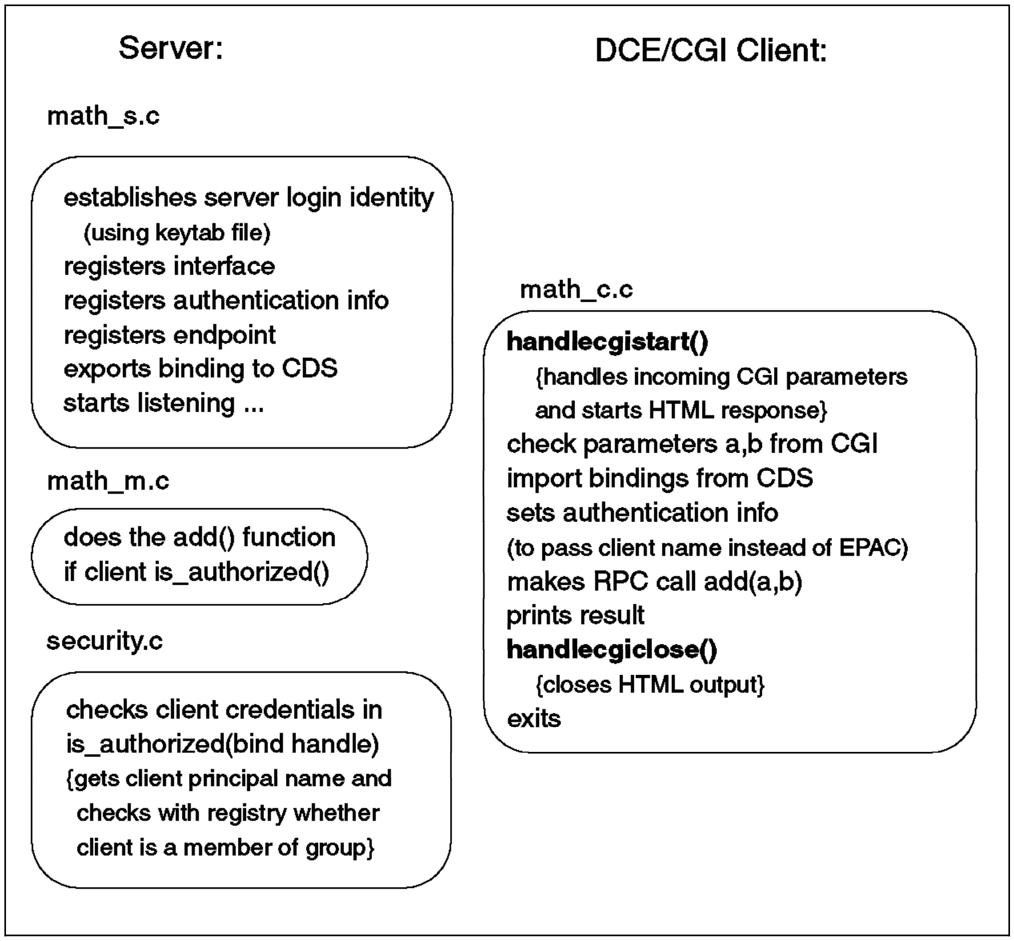

Server:
DCE/CGI Client:
math_s.c
establishes server login identity
(using keytab file)
registers interface
registers authentication info
registers endpoint
exports binding to CDS
starts listening ...
math_m.c
does the add() function
if client is_authorized()
security.c
checks client credentials in
is_authorized(bind handle)
{gets client principal name and
checks with registry whether
client is a member of group}
math_c.c
handlecgistart()
{handles incoming CGI parameters
and starts HTML response}
check parameters a,b from CGI
import bindings from CDS
sets authentication info
(to pass client name instead of EPAC)
makes RPC call add(a,b)
prints result
handlecgiclose()
{closes HTML output}
exits

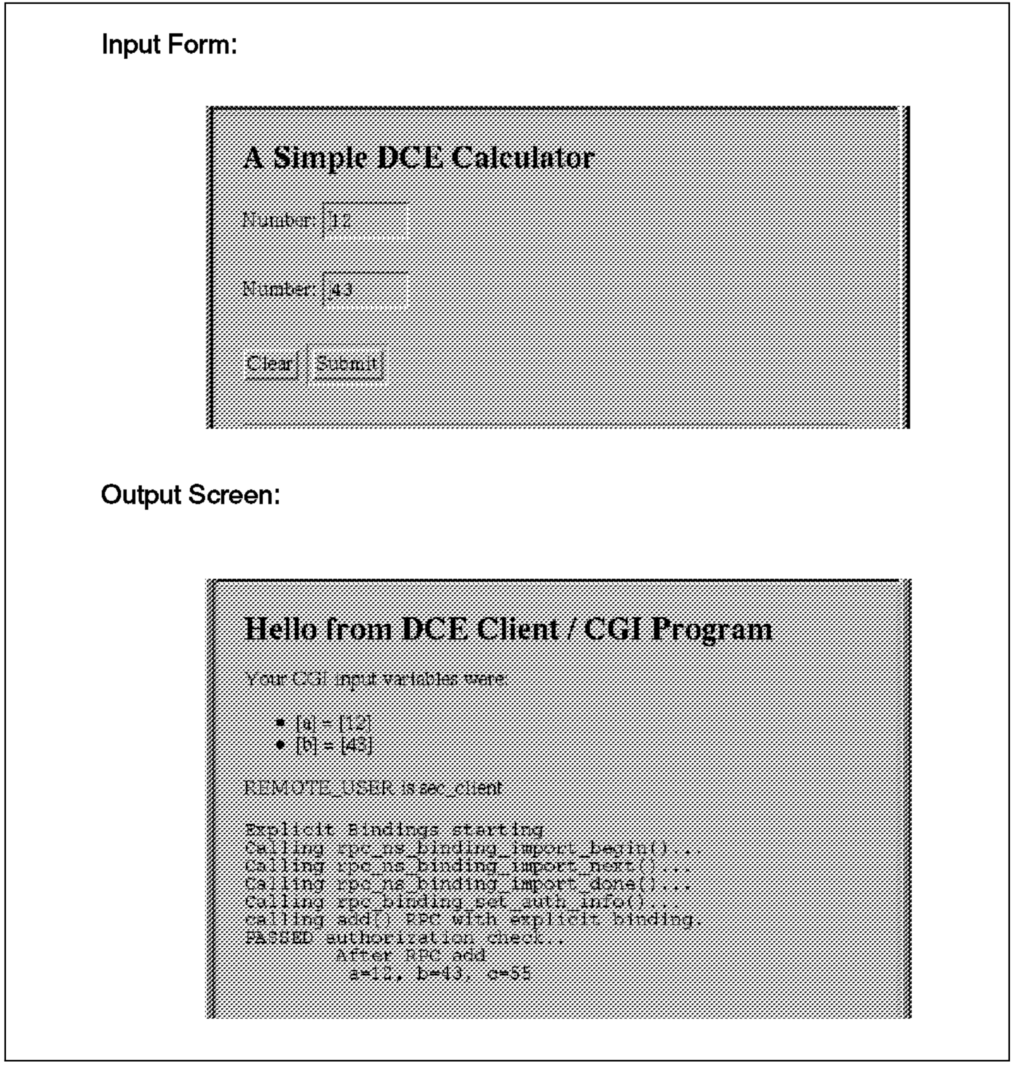
Figure 44. Outline of Code in Combined CGI/DCE Client Example

The highlighted lines show the calls to the CGI functions that are in cgifns.h. There are also a number of C function libraries available on the Internet that can be used when adding CGI functionality to your C programs.

The HTML source of the form that prompts the browser user for two numbers and launches the cgimath_c client program is shown here:

```
<!DOCTYPE HTML PUBLIC ″-//IETF//DTD HTML//EN″>
<html> <head>
<title>A Simple DCE Calculator</title>
</head>

<body>
<h1>A Simple DCE Calculator</h1>

<form action=″/cgi-bin/cgimath_c″ method=″POST″>

<p>
Number: <input type=″TEXT″ name=″a″ size=″6″ MAXLENGTH=″6″> <p>
Number: <input type=″TEXT″ name=″b″ size=″6″ MAXLENGTH=″6″> <p>

<p> <p> <p>
<input type=″RESET″ value=″Clear″>
<input type=″SUBMIT″ value=″Submit″>

</form>

<hr>

</body> </html>
```

A sample input and output from this example, as seen in the browser, is shown in Figure 45 on page 105. The server will only complete the addition if the client has authenticated as a principal which is a member of the group good_guys. The authentication is taken care of here by DFS Web Secure because the path to the CGI program directory is within DFS.



Input Form:

A Simple DCE Calculator

Number: 12

Number: 43

Output Screen:

Hello from DCE Client / CGI Program

Your CGI input variables were:

- [a] = [12]
- [b] = [43]

REMOTE_USER is sec_client

```
Explicit Bindings starting
Calling rpc_ns_binding_import_begin()...
Calling rpc_ns_binding_import_next()...
Calling rpc_ns_binding_import_done()...
Calling rpc_binding_set_auth_info()...
calling add() RPC with explicit binding.
PASSED authorization check..
          After RPC add
           a=12, b=43, c=55
```

Figure 45. Browser Form and Output from Combined CGI/DCE Client Example

The source code of cgimath_c is included in Appendix B, “Program Listings” on page 159.

5.6 Security Considerations

This chapter has covered aspects of opening up a Web site with DFS for business on the Internet or intranet. This site can also include interactive programs through CGI and Java-related technologies. The security considerations for setting up a Web server, covered in 4.3, “Security Considerations on a Web Server” on page 57, need to be extended here to encompass this interactive aspect.

Another key security consideration is how to prevent non-essential network traffic from reaching your Web server and your secure internal network. For its prime role, a Web server needs to be accessible to remote users on wider area networks (whether Internet or intranet) as well being accessible for users on the internal network where its contents will be maintained its contents. Firewalls are used to implement controls on network traffic, and considerations for running DCE and DFS across a firewall are covered in Chapter 8, “Bridging Networks Together: DCE Over Firewalls” on page 143. Installing DFS Web Secure does not make it essential to alter the implementation of firewalls for standard HTTP traffic between the Web server and Web browsers. However, in the scenario of a publicly accessible Web server being placed across a firewall from DCE servers on a secure internal network, some changes will need to be made to a firewall setup to allow DCE traffic to flow between the DCE servers and DFS Web Secure on the Web server.

5.6.1 Screening CGI Programs

It is important to screen CGI programs since they are programs that anyone in the world can run on your system. Accordingly, look out for security holes in scripts. In general the scripts/programs should not trust the user input. In particular, do not put user data in a shell command without verifying the data carefully; otherwise a hacker could drive a virtual truck through this security hole. Here are a few examples:

- Beware the `eval` statement.

  Languages like Perl and shell scripts provide an `eval` command that allows you to construct a string and have the interpreter execute that string. This can be dangerous. Consider this example in a shell script:

  ```
  eval 'echo $QUERY_STRING | awk 'BEGIN{RS="&"} {printf "QS_%sn",$1}' '
  ```

  This clever snippet takes the query string and converts it into a set of variable set commands. Unfortunately, the script can be attacked by sending it a query string that starts with a semicolon (**;**), for example how would `;rm -r /*` be treated?

- Do not trust the client to do anything.

  A well-behaved client will escape any characters that have a special meaning to the shell in a query string and thus avoid problems with your script misinterpreting characters. A mischievous client may use special characters to confuse your script and gain unauthorized access.

  If you use any data from your client to construct a command line for a system call, be sure to place backslashes before any characters that have special meaning to the shell before calling the function. This can be achieved easily with a short function.

5.6.2 Security Summary

Here is an overall summary of security considerations for a Web server providing DFS access:

- Secure the base system. Operating systems and underlying hardware (including the TCP/IP services that it provides) must be protected.
- Secure passwords for key user IDs. Procedures need to be followed for password composition, update intervals, and so forth.
- Secure Network traffic around the Web server with Firewall(s).
- Secure DCE resources in the DCE regsitry and set up DFS ACLs on DFS filespace.
- For encryption security on Web server to browser client traffic, configure Secure Sockets Layer (SSL). A more detailed description about SSL setup can be found in 4.3, “Security Considerations on a Web Server” on page 57.
- Screen the interactive content that will be hosted to execute on the Web server system, particularly CGI scripts.

- Use the monitoring capabilities of your Web server and DFS Web Secure and check the activity on the server.
- Implement and use the threshold features in DFS Web Secure.

Chapter 6. DE-Light Web Client

The DE-Light 2.0 product from Transarc Corporation is a gateway technology connecting lightweight clients to existing enterprise DCE application services. The environment provides a non-DCE RPC mechanism for clients that is interpreted by a DE-Light gateway and passed on to named DCE application servers using the DCE protocol. It is a powerful tool providing the modern desktop client, the Web browser, with direct access to DCE applications through Java applets. The browser remains a lightweight interface, ubiquitous across a range of platforms and across the world.

The environment supports clients developed directly for the command line Java interpreter, the Java AppletViewer (part of the Java Development Kit, see section 6.3.1, “The Java Development Kit” on page 123), and Web browsers through applets. The DE-Light gateway is run as a separate process on the same machine as the Web server (for Java security reasons) and links Java indirectly with DCE. Security is supported between a Web browser applet and the gateway by optionally using the Secure Sockets Layer (SSL) protocol and from the gateway to the DCE application server using standard DCE technology. The DE-Light classes available to Java are simple yet powerful. They provide an interpretation of DCE technology from an object-oriented perspective.

This chapter begins with an overview of the DE-Light Web technology describing the breadth of the technology. We give a simplified guide to preparing the environment for DE-Light, then finish with a complete example using each of the client interface types.

6.1 Introduction to DE-Light Web Client and DE-Light Gateway

The World Wide Web is embracing a new wave of information delivered through dynamic access to live sources of data. In the past most interaction has centered around static information access where the browser provides navigation information but little extra customization. Both CGI and Java Applet technology provide dynamic data. CGI puts the computational load on the Web server, and if generating graphics can put a significant load on the network connection, Java applets transfer the computational burden from the server to each browser client. This is a significant advantage when many browsers are accessing a single Web server.

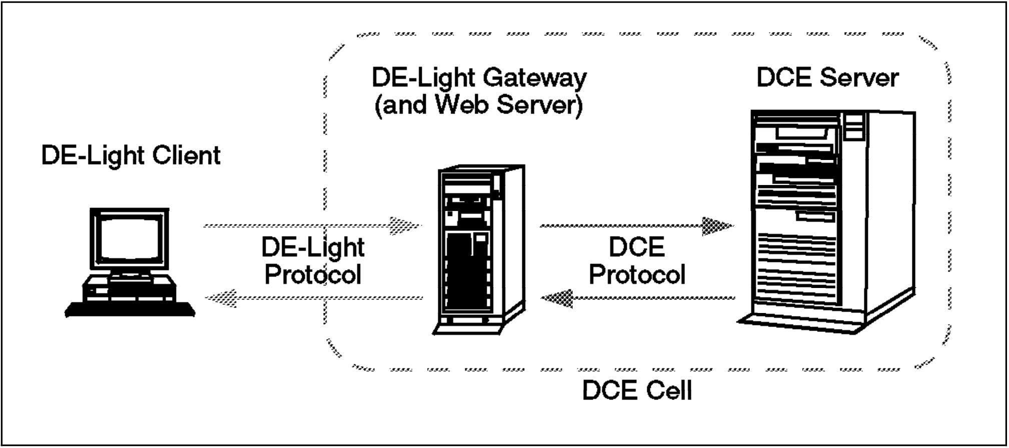


Figure 46. DE-Light Overview

Transarc's DE-Light is a product providing DCE access to clients that either cannot or do not want DCE at the client node. This might be because of limited resources or licensing issues. Figure 46 shows the relationship between the DE-Light components. The client is loaded with DE-Light support code typically comprising less than 100 KB. It uses a dynamic remote procedure call library to request RPC invocations at a DCE server by the DE-Light gateway. The gateway builds DCE bindings, invokes RPC calls, and returns data to the requesting client. The gateway is generic in the sense that it can make calls to arbitrary DCE servers. It does restrict the use of unions, pipes, multi-dimensional arrays and full pointers, but most other DCE attributes and data types are supported. Only the gateway and the DCE application server are DCE-aware and are part of the DCE environment.

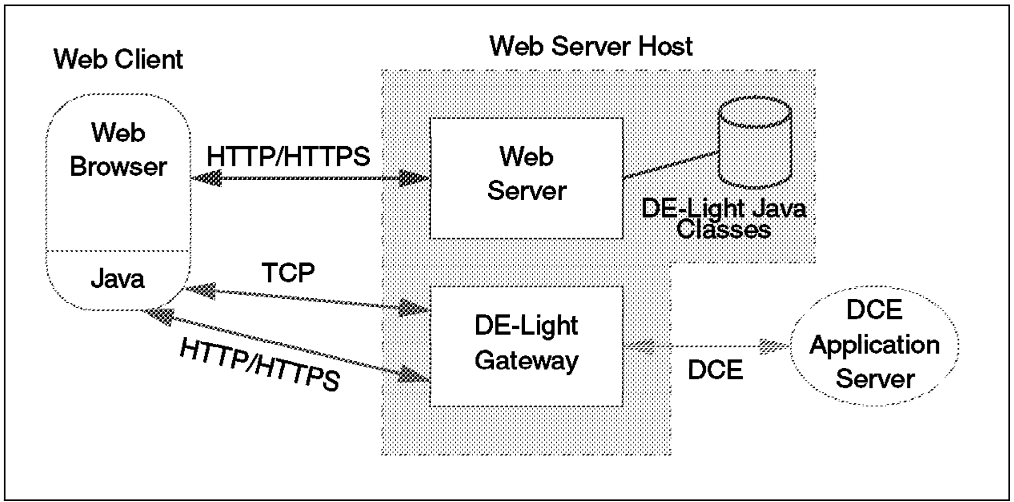


Figure 47. The DE-Light Web Architecture

The DE-Light Web client provides the DE-Light client libraries to Java as Java classes. Java applets make use of the lightweight DCE access facility provided by the DE-Light gateway that is using native Java. These classes are downloaded from the Web server (see Figure 47) in the same way as other Java classes. The security restrictions enforced by Java applets mean the Web server and DE-Light gateway must reside on the same host because network connections can only be set up back to the Web server host.

DE-Light Web can use TCP, HTTP, or HTTP over the Secure Sockets Layer (HTTPS) to communicate with the DE-Light gateway. Beyond the gateway, DCE security is available. See 6.2.2, "Secure Sockets Layer (SSL) Security" on page 114, for further information on configuring security in the DE-Light environment.

DE-Light Web provides an OO-friendly interface to DCE. A connection is established with the gateway and binding instructions given. RPC calls are then constructed and subsequently managed by the gateway with results returned. The Java DE-Light interface is very simple. For DCE access, two classes are provided.

- DrpcConnection

  This class implements a connection to a DE-Light gateway. The class interface is very simple. Table 6 outlines the DCE methods available at a connection.

Table 6. DrpcConnection Class Methods and Constructors

`DrpcConnection()`	**Create a new gateway connection.**
`callRpc()`	**Executes a named RPC.**
`dceLogin()`	**Sets the DCE login context for subsequent RPC calls.**
`dceUsername()`	**Returns the current DCE username.**
`dictionary()`	**A reference to the `DrpcDictionary` associated with the connection. See below.**
`getDceSecurityLevel()`	**Returns the current DCE security level.**
`setDceSecurityLevel(it)`	**Sets the current DCE security level.**
`setSecurity(int)`	**Controls the security between the client and the DE-Light gateway (currently not implemented).**

  Additional methods are available to handle Encina transactions. DE-Light Web also translates DCE return codes into `drpcCommException` exceptions for client to gateway exceptions and to `drpcException` exceptions for DCE exceptions.

- `DrpcDictionary`

  This class provides a drpc dictionary to be used by `DrpcConnection`. It is an extended Java hash table with a few convenience functions such as `put()` and `getint()`. These deal directly with integers rather than with strings.

DE-Light gives more ammunition to the push for the Web browser to be the ubiquitous lightweight client interface. Just as Java ORB (the Object Management Group's Object Request Broker) technology has provided Common Object Request Broker Architecture (CORBA) access through Java applets, DE-Light provides the tools for accessing existing DCE application servers through the web.

A typical scenario would leverage off the key advantages that DCE provides to a distributed environment, namely security, scalability, and distribution. Consider a global transport company that uses a DCE infrastructure to schedule and manage package transportation across a global network of interlinked cells. Typically, dedicated clients are used within the transport company to interact with the various interlinked management packages. Now we want to give customers access to information about their package as it moves about the world toward its destination. Users access an external Web server that provides an ordered list of DE-Light gateways sorted in ascending order of load. The Java applet selects the least loaded, connects and then uses our package ID and personal identification to locate our package in the DCE world. The up-to-date information is available dynamically and directly to our browser. The same Java class libraries could be shared with intranet applications used in a secure mode within the company. We have managed to leverage our software investment in our intranet solutions to provide access to external customers while maintaining security and global access.

We now look at how to prepare the environment for installation of DE-Light Web in an existing DCE environment and then demonstrate its use through a simple Java example.

6.2 Environment Preparation

DE-Light Web builds on the existing DE-Light gateway to provide Java access to DCE application servers. Three areas of preparation must be addressed:

- The *DE-Light gateway*, `drpcgwy`, must be started and any required DCE application server interfaces (IDL files) loaded.
- Java applets at a Web server must be provided with access to the DE-Light Java classes.
- The *Java client* should be initialized with references to both the gateway and the DCE application server being accessed.

Figure 48 shows these components at work as well as the role of the DE-Light administration tool, `drpcadmin`. This is used to load DCE IDL definitions and stop the gateway, among numerous other functions.

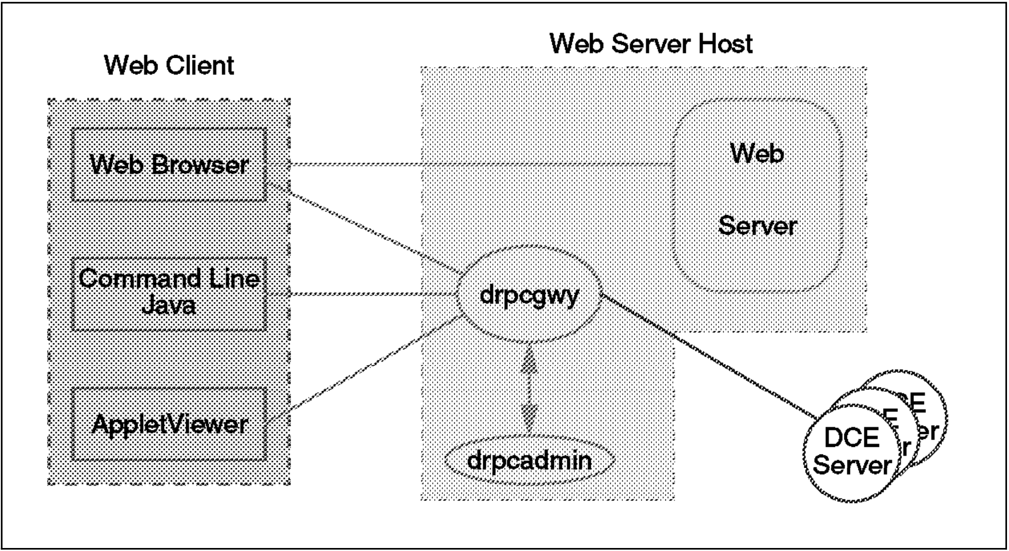


Figure 48. Components in the DE-Light Web Environment

This section is designed as a shortcut description to get a DE-Light gateway off the ground and useful. Full details of the gateway and administration tool can be found in the DE-Light documentation that comes with the product (under AIX, this is /usr/lpp/DE-Light/lib/javadocs/).

6.2.1 DCE Configuration

The DE-Light gateway is able to integrate with DCE security several ways. As a minimum, it requires permission to set up a CDS entry to identify itself to the administrative tool. One approach is to make entry creation available to all, allowing anyone to start a gateway.

```
# dcecp
dcecp> directory create /.:/DE-Light
dcecp> acl modify /.:/DE-Light -change {unauthenticated rwi}
dcecp> acl modify /.:/DE-Light -change {any_other rwi}
dcecp> exit
```

Alternately, the gateway can be given an identity of its own in the form of a DCE principal. This requires setting up the DCE principal, gwy-prin, and a group, gateway.

```
# dcecp
dcecp> group create gateway -inprojlist y
dcecp> user create gwy-prin -group gateway -organization none \
       -password gwy-prin -mypwd -dce-
dcecp> exit
```

We then create a keytab file (gwy-prin.ktf in our example below) to store the gateway principal's password.

```
# rgy_edit
rgy_edit=> ktadd -p gwy-prin -pw gwy-prin -f gwy-prin.ktf
rgy_edit=> ktadd -p gwy-prin -f gwy-prin.ktf -a -r
rgy_edit=> ktdelete -p gwy-prin -f gwy-prin.ktf -v 1
rgy_edit=> exit
```

A gateway administration account should also be created to ensure secure administration of the gateway.

```
# dcecp
dcecp> user create gwy-admin -group gateway -organization none \
       -password gwy-admin -mypwd -dce-
dcecp> exit
```

As an alternative to the open access to the gateway CDS entry just described above, we can restrict the entry to only those principals in the gateway group. The authenticated accounts are given permission to update the CDS entry for a gateway.

```
# dcecp
dcecp> directory create /.:/DE-Light
dcecp> acl modify /.:/DE-Light -add {group gateway rwdti}
dcecp> acl modify /.:/DE-Light -change {unauthenticated tr}
dcecp> acl modify /.:/DE-Light -add {group gateway rwdti} -ic
dcecp> acl modify /.:/DE-Light -change {unauthenticated tr} -ic
dcecp> acl modify /.:/DE-Light -add {group gateway rwdti} -io
dcecp> acl modify /.:/DE-Light -change {unauthenticated tr} -io
dcecp> exit
```

6.2.1.1 Authenticated Administration

At startup, the gateway can be configured to allow any user to perform administration tasks, or it can be limited to a specific principal such as `gwy-admin`. This is specified using command line options as described in Table 7.

Table 7. DE-Light Authenticated Administration

-Z *level*	**Level 0 disables administrative authorization allowing anyone to administer the gateway. Level 1 only allows the principal specified by the `-A` option to administer the gateway.**
-A *principal*	**The sole administrative principal allowed to administer the gateway. Can also be specified with the `ENCINA_ADMIN_PRINCIPAL` environment variable.**

6.2.1.2 Authenticated Users

A DE-Light Java client authenticates with DCE using the login call in the Java DE-Light API.

```
drpc.dceLogin(name,passwd);
```

This associates the given username and password with the named gateway connection, `drpc`. When a DE-Light RPC call is made through the gateway, this identity information will be used by the gateway to obtain DCE credentials and authenticate the gateway as the named client. The DCE application server will see the gateway as the authenticated client.

If the client does not use the `dceLogin` function, then the gateway will, by default, operate in an unauthenticated mode during RPC calls to DCE application servers.

The gateway can be configured to use the credentials of a known DCE principal if no DCE user is provided through the Java API. Table 8 describes the three command line options for the gateway to provide a default gateway identity.

Table 8. DE-Light Gateway Principal

-d	**Default to the named principal if the client is unauthenticated. The Java API provides a method for atuthenticating with DCE as a named user with password.**
-p *principal*	**The default principal for the gateway.**
-k *filename*	**Keytab file for the default principal.**

In this case, clients that do not provide identity information through the `dceLogin` call will be authenticated as the default gateway user.

6.2.2 Secure Sockets Layer (SSL) Security

The DE-Light Web architecture bridges Java environments with enterprise distributed applications defined within the DCE domain. It separates interaction security into two domains.

- The link between the DE-Light gateway and any DCE application server is dealt with from within the *DCE security domain*. The gateway may operate under the identity of a client application or, if unauthenticated, has the option of providing a default identity. The DE-Light API allows the client to specify a category of security between the gateway and server based on standard DCE privacy options (see Table 9 on page 115).

- Security between the client and the gateway can't be handled by DCE as the client is outside the DCE domain. The TCP protocol option for gateway access (see 6.2.3, "Running the DE-Light Gateway" on page 121) provides no encryption of data between client and gateway. The HTTP/HTTPS option is able to provide authentication and encryption based on SSL.

Table 9. DCE Security Options Available to DE-Light Clients

DE-Light Name	DCE Name	Description
DRPC_DCE_PROTECT_DEFAULT	Default	Default DCE security level.
DRPC_DCE_PROTECT_NONE	None	No authentication is performed, no tickets are exchanged, and transmissions are in the clear.
DRPC_DCE_PROTECT_CONNECT	Connect	Protection only at connection establishment.
DRPC_DCE_PROTECT_CALL	Call	Protection at initial RPC request to server.
DRPC_DCE_PROTECT_PACKET	Packet	Ensures all RPCs from a given secure connection are from the same principal.
DRPC_DCE_PROTECT_PKT_INTEG	Packet Integrity	Ensures that no data transmitted between the gateway and server was modified.
DRPC_DCE_PROTECT_PKT_PRIVACY	Packet Privacy	Combines all previous security measures as well as encryption of data.
DRPC_DCE_PROTECT_MAX_VALUE	Packet Privacy	Highest security available. In the current version: DRPC_DCE_PROTECT_PKT_PRIVACY

SSL is currently only supported through an SSL-aware Web browser client such as Netscape Navigator. The DE-Light Java API provides a call to set security on the client/gateway connection based on SSL. The options are outlined in Table 10 on page 116.

Table 10. DE-Light SSL Security Options

DE-Light Name	Description
SEC_NONE	No security.
SEC_BEST	Use the best DE-Light security available. May mean none.
SEC_INHERIT	Inherit DE-Light security based on DCE security used between the gateway and server.
SEC_CIRCUIT_AUTH	Use SSL privacy and integrity checking on connection establishment only.
SEC_PACKET_INTEGRITY	SEC_CIRCUIT_AUTH and integrity checking on requests.
SEC_PACKET_PRIVACY	Use SSL security with cipher 40 bits or greater.
SEC_ENCRYPT	Same as SEC_PACKET_PRIVACY.
SEC_PACKET_PRIVACY_WORLD	SEC_PACKET_PRIVACY with US-exportable ciphers.
SEC_PACKET_PRIVACY_US	Use SSL security with cipher greater than 40 bits.
SEC_PACKET_PRIVACY_US_128	Use SSL security with cipher 128 bits or greater.
SEC_MAX_VALUE	Use highest SSL security defined (in the current version: SEC_PACKET_PRIVACY_US_128).

6.2.2.1 Preparing DE-Light SSL Security

DE-Light uses SSL in much the same way as other Web technology described in 1.3.3, "Secure Transfer Protocols: SSL and S-HTTP" on page 10. SSL is only available between Web browser clients using DE-Light Web and a gateway that has been initialized to support the HTTP/HTTPS protocols (see 6.2.3, "Running the DE-Light Gateway" on page 121).

A client contacts the gateway over the insecure HTTP port and receives in response the public key of the gateway. This key is then used to encrypt or decrypt interaction over the secure HTTPS link. The public key is provided to the client in the form of a certificate that includes the public key, the Distinguished Name of the gateway, the serial number or issue date of the certificate, and the expiration date of the certificate.

A certificate is signed by a Certificate Authority (CA), such as *VeriSign* (see 1.3.4, "Electronic Transactions" on page 12). For testing purposes, DE-Light provides a mechanism to generate a self-signed certificate that is encrypted with the private key of the gateway and provides no security guarantees, but does allow simple testing of SSL mechanisms.

Obtaining a CA-Signed Certificate: A signed certificate can be obtained from many commercial Certificate Authorities, for example VeriSign. A full description of the process for obtaining a certificate is described in the DE-Light documentation:

`/usr/lpp/DE-Light/lib/javadocs/cert_gdref/signed.html`.

It is beyond the scope of this book to reprint this publically available procedure.

Generating and Installing a Self-Signed Certificate: These steps are similar to those performed in the Netscape FastTrack Server Certificate request procedure. First, a certificate request must be generated based on a public/private key pair. DE-Light provides a tool to do this for you.

The following sequence of steps take you through the generation of both the key pair and the associated certificate request. The same process is required for requesting a certificate from an accredited CA or for generating your own self-signed certificate. Keyboard input is shown in bold in the following examples.

```
# /usr/lpp/DE-Light/bin/mkkf
.......................
MKKF Key Manager
Copyright IBM Corp. 1996
All Rights Reserved

Key Ring Menu

 Currently Selected Key Ring:  (none)

N - Create New Key Ring File
O - Open Key Ring File
X - Exit

Enter a command: N
Enter a name for the key ring file, or press ENTER for keyfile.kyr.

Key Ring Menu

 Currently Selected Key Ring:  keyfile.kyr

N - Create New Key Ring File
O - Open Key Ring File
S - Save Key Ring File
A - Save Key Ring as Another File
P - Set Password for Key Ring File
C - Create Stash File for Key Ring File
R - Receive a Certificate into a Key Ring File
W - Work with Keys and Certificates
X - Exit

Enter a command: W
```

We first create the a new *key ring file*. This file stores public key certificates and is maintained by the mkkf utility. We can list keys within this file and see many certificates from Certificate Authorities such as VeriSign and RSA. The key ring file is protected by a password that must be entered to modify or view the key ring file. The SSL infrastructure uses a certificate from the key ring file to initialize security with an SSL-aware client.

```
Key Menu
 Currently Selected Key Ring:  keyfile.kyr
Selected Key Entry:   (none)

L - List/Select a Key To Work With
C - Create a New Key and Certificate Request
I - Import a Key From an Armored Key File
X - Exit This Menu
```

```
Enter a command: C
```

In this case we create a new key pair and certificate.

```
Enter password to use for the key file: ########
Enter the password again for verification: ########
Should the password expire?
Enter Y for yes or N for No: N

Password successfully set.
Press ENTER to continue

Choose Certificate Type Menu
S - Server Certificate
L - Low Assurance
C - Cancel

Enter a command: S
```

A low-assurance certificate is used for testing and is not supported by the DE-Light gateway.

```
Compose Secure Server Certificate Menu

Current Certificate Information
Key Name:  (none)
Key size:  0
Server Name:  (none)
Organization:  (none)
Organizational Unit:  (none)
City/Locality:  (none)
State/Province:  (none)
Postal Code:  (none)
Country:  (none)

M - Modify the Certificate Fields
R - Ready To Create Key and Certificate Request
C - Cancel

Enter a command: M
```

The empty certificate must be modified with correct details for your organization.

```
Enter a name to use for the key entry: Mary User

1:   508
2:   512
3:   768
4:   896
5:   1024
Enter the number corresponding to the key size you want: 5
```

Cryptography Export Restrictions

The United States Commerce Department regulates the export of cryptography technology.

RSA public keys to be exported outside the U.S. and Canada must not exceed 512 bits. Key sizes above this size are generally not available for export. Within the U.S. and Canada, key sizes up to 1024 are available.

There is also a restriction on the export of DES technology requiring no more than 40-bit key technology be exported outside the U.S. and Canada. This has recently been extended to 56-bit if certain key reversal tools are available.

We choose a key size of 1024 bits. A larger key increases security, but also requires more processing for encryption and decryption.

Additional information should be entered to identify the address for the Distinguished Name, including organization, and so on.

```
Compose Secure Server Certificate Menu

Current Certificate Information
Key Name:  Mary User
Key size:  1024
Server Name:  ev1.itsc.austin.ibm.com
Organization:  Widget Co.
Organizational Unit:  Manufacturing
City/Locality:  Anytown
State/Province:  Anywhere
Postal Code:  12345
Country:  US

M - Modify the Certificate Fields
R - Ready To Create Key and Certificate Request
C - Cancel

Enter a command: R
```

We now generate the certificate based upon the information we have provided.

```
Enter file to store the certificate request in: cert.req
Creating Private Key....
Private key was successfully created.
Creating certificate request.....
Certificate request was successfully created
Adding new key to key file.
The new key and certificate request were created successfully.
Press ENTER to continue

Key Menu
 Currently Selected Key Ring:  keyfile.kyr
Selected Key Entry:   Mary User

L - List/Select a Key To Work With
S - Show Information about Selected Key
D - Delete Selected Key
C - Create a New Key and Certificate Request
```

```
I - Import a Key From an Armored Key File
E - Export Selected Key To an Armored Key File
F - Make Selected Key the Default Key for this Key Ring
T - Mark Selected Key as a Trusted Root
R - Create A Certificate Request For Selected Key
X - Exit This Menu

Enter a command: X
```

We now recursively exit the menus and are ready to install a self-certified certificate in the key ring for this gateway.

```
# /usr/lpp/DE-Light/bin/mkkf
........................
MKKF Key Manager
Copyright IBM Corp. 1996
All Rights Reserved

Key Ring Menu

 Currently Selected Key Ring:  (none)

N - Create New Key Ring File
O - Open Key Ring File
X - Exit

Enter a command: O
```

Open the key ring file we created in the previous step.

```
Enter name of file to open or press ENTER to cancel:
keyfile.kyr
Enter the password for  keyfile.kyr:

Key Ring Menu

 Currently Selected Key Ring:  keyfile.kyr

N - Create New Key Ring File
O - Open Key Ring File
S - Save Key Ring File
A - Save Key Ring as Another File
P - Set Password for Key Ring File
C - Create Stash File for Key Ring File
R - Receive a Certificate into a Key Ring File
W - Work with Keys and Certificates
X - Exit

Enter a command: R
```

Receive the certificate into the key ring. This will recognize the certificate as being generated by the same key ring private key and verify that this insecure certificate should be added to the key ring.

```
Enter file name or press ENTER for  Cert.txt.
cert.req
This is a self-signed certificate. Add it to key file? Enter Y for yes or N for No:
Y
Certificate added to key ring.
Press ENTER to continue
```

```
Key Ring Menu

 Currently Selected Key Ring:  keyfile.kyr

N - Create New Key Ring File
O - Open Key Ring File
S - Save Key Ring File
A - Save Key Ring as Another File
P - Set Password for Key Ring File
C - Create Stash File for Key Ring File
R - Receive a Certificate into a Key Ring File
W - Work with Keys and Certificates
X - Exit

Enter a command: X
```

Now we can exit and save the modified key ring file.

Configuring the DE-Light Gateway to use SSL: Two new options to drpcgwy are used to load the DE-Light gateway with SSL awareness. These are outlined in Table 11.

Table 11. DE-Light Gateway SSL Options

-K *pathname*	Location of the SSL key ring file from which the gateway certificate is obtained.
-m *password*	Key ring password as specified in key ring creation. If this option is left out the administrator starting the gateway is prompted for the key ring password.

Recall the SSL support is only available through SSL-aware Web browsers and requires the gateway to be initialized with HTTP/HTTPS support as described in the following section.

6.2.3 Running the DE-Light Gateway

Starting the DE-Light gateway is usually done in two steps. First, the gateway process is started with adequate command line options for proper configuration, and then it is populated with any necessary IDL files. The gateway needs to know the interface definitions of the DCE applications through those IDL files in order to generate correct DCE RPCs.

6.2.3.1 Starting the DE-Light Gateway

The gateway (drpcgwy) is initialized from the command line using a combination of options specifying:

- DCE security requirements (see 6.2.1, "DCE Configuration" on page 113)
- SSL options (see 6.2.2, "Secure Sockets Layer (SSL) Security" on page 114)
- Other management options

The more useful options are summarized in Table 12 on page 122.

Table 12. Useful DE-Light Gateway Options

-n *name*	**The CDS name of the gateway. Used by the administration tool to locate the gateway interface.**
-e *protocol:[endpoint(s)]*	**The contact point for DE-Light clients. It takes one of two forms:** `-e TCP:[1234]` **or** `-e HTTP:[1234,1235]` **where the first is a single TCP connection with associated port and the second a pair of HTTP/HTTPS ports to be used by the gateway. In the second case, both ports must be included for use by secure and insecure parts of the HTTPS protocol.**
-T *class=type:destination*	**This turns on the trace facility built into the gateway. Many classes are supported, but the most useful are either** `error`**,** `audit`**, or** `all`**. Type is** `FILE` **(to the named destination file),** `STREAM` **(to either destination stdout or stderr), or** `AIX` **(to the** `trace` **stream or** `errlog`**). Destination is the required type argument.**

Typical initialization of an authenticated gateway that provides a default DCE principal gwy-prin for unauthenticated clients might look like:

```
# drpcgwy -n /.:/DE-Light/examples/drpcGateway -Z 0
          -p gwy-prin  \
          -k /usr/lpp/DE-Light/local/gate1/gwy-prin.ktf \
          -e "tcp:[4913]" \
          -A gwy-admin -d &
```

6.2.3.2 Populating the Gateway With Known Interfaces

Once the gateway responds with

```
Drpc Gateway is ready.
```

the administrator (gwy-admin in our case) can populate the gateway with DCE IDL files. The gateway requires prior knowledge of interfaces before a client contacts the gateway to use its services to access a DCE application server. These IDL definitions can be loaded at gateway startup by using the -L option or as the gateway is running by using the drpcadmin tool.

```
drpcadmin load /tmp/math.idl
```

Note

The gateway processes IDL files by generating DCE stub files. This requires the gateway run on a host with a DCE development environment installed as well as a C compiler.

Further descriptions of both the drpcadmin tool and drpcgwy are available in the Administrators Reference, available in the release we used in the file /usr/lpp/DE-Light/lib/javadocs/admin_ref/dar-toc.html.

6.3 An Example Using DE-Light Web

This section provides a step-by-step guide to getting a simple client/server application running within a DE-Light Web and DCE environment. It does not exhaustively exercise the functionality of DE-Light Web, but does provide a complete example including each of the Java command line, AppletViewer and Web browser clients.

For details of commands and more complete information, the reader is directed to the DE-Light documentation that comes with the product. In the release we used, the file was `/usr/lpp/DE-Light/lib/javadocs/all-toc.html`.

6.3.1 The Java Development Kit

Java is made available through the Java Development Kit (JDK). The JDK lets you write Java applets and applications that conform to the Java API. The original JDK was provided by Sun Microsystems, Inc. for Windows NT, Windows 95, and x86/SPARC Solaris. Many other platforms are also now supported, including AIX (see `http://ncc.hursley.ibm.com/javainfo/dev_reg.html` for information on obtaining IBM's JDK for AIX).

The Java Development Kit contains several parts.

- A *Java runtime environment* consisting of a Java bytecode interpreter and core Java classes. Java bytecodes are generated by a Java compiler (see below) and provide an intermediate representation of the language which is more compact and efficient than direct Java source code.
- The *Java tools* support development of Java applications. It includes:
  - A Java compiler which transforms Java source code into Java bytecodes to be interpreted by a Java interpreter,
  - A Java command-line debugger,
  - An AppletViewer for testing and running Java applets.
- Several *Java demos* are included, complete with Java source code, which demonstrate applet facilities.

Java interpreters are also supported within Web browsers such as Netscape Navigator and Microsoft Explorer. They interpret Java applet bytecodes that have previously been compiled by a Java compiler.

6.3.2 Starting and Stopping the Gateway

The DE-Light gateway exists to provide an intermediary between a lightweight client and a DCE application server.

```
# LANG=en_US; export LANG
# NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat:\
/usr/lpp/DE-Light/lib/nls/msg/%L/%N; export NLSPATH
# drpcgwy -n /.:/DE-Light/examples/drpcGateway -Z 0 \
        -p gwy-prin \
        -k /usr/lpp/DE-Light/local/gate1/gwy-prin.ktf \
        -K /usr/lpp/DE-Light/local/keyfile.kyr \
        -m delight \
        -e "http:[1423,2423]" \
        -A gwy-admin &
```

We first set the path for message libraries that ensure that they appear in our language of choice. The gateway is initialized with its own principal identity, gwy-prin, but we haven't defined this as a default identity for unauthenticated clients. This is not unintentional. We have given this identity write permission to the gateway CDS entry, preventing unauthorized parties from registering under this name.

An SSL key ring file (keyfile.kyr) along with the password (delight) need to be provided with the command to enable SSL security for SSL-capable Web browsers. Recall that it is possible to leave off the password and have drpcgwy prompt you for this password on startup. Both the key ring and keytab files must be readable by the user starting the gateway process. We do this as root, but it is not necessary.

The HTTP/HTTPS protocols on ports 1423 and 2423 provide transport.

We identify a specific DCE principal responsible for gateway administration. All drpcadmin operations must be performed through this identity, as for example to stop the gateway:

```
# dce_login gwy-admin
Enter Password:
# drpcadmin stop gateway -gateway /.:/DE-Light/examples/drpcGateway
```

The gateway must be stopped through the drpcadmin utility. The CDS entry for the gateway identifies which server is to stop.

6.3.3 Loading the IDL Definition

The gateway can be initialized with server IDL files, but it is easier to load these at runtime using the drpcadmin utility. Since we specified an administrator principal, this must be used when loading IDL files into the gateway. The environment variable identifies which gateway is to be used, or alternatively a -gateway option can be given to drpcadmin.

```
# dce_login gwy-admin
Enter Password:
# DRPC_GATEWAY=/.:/DE-Light/examples/drpcGateway; export DRPC_GATEWAY
# drpcadmin load math.idl
```

IDL files can be replaced in the gateway by first doing a drpcadmin unload and then a drpcadmin load with the new IDL file. The IDL file we use in this example (math.idl) is explained in the next section.

6.3.4 Starting the DCE Application Server

Our DCE application server is very simple. It provides a single RPC call to add two numbers together and provides the result as an output parameter (see Figure 49 on page 125).

```
/*
 * Filename: math.idl
 */

[
uuid(00602160-2F96-1B81-846A-10005A8D5B0E),
version(1.0)
]
interface ServerID
{
/* principal name used by server */
const char  *SERVER_NAME        = "ServerName";

/*
*  add operation returns 0 if and only if client is authorized;
*  otherwise, returns -1
*/
long int add(
        [ in ]  handle_t       bh,
        [ in ]  long int       a,
        [ in ]  long int       b,
        [ out ] long           *result);
}
```

Figure 49. DCE IDL File math.idl

The source code for this DCE application server is given in B.3, “DCE Math Client and Server” on page 167.

```
# ./math_s &
```

The server names itself in the CDS as `/.:/Servers/MathPacServerID`, and this name will be required by DE-Light Web clients to identify the DCE application server they want to use.

6.3.5 The Command Line Java Client

The simplest form of Java client using the DE-Light gateway is invoked directly through the Java interpreter. Source code for this simple application is provided in B.2.2, “Calc Class (Calc.java)” on page 161. The main body for this class takes six arguments.

- A protocol identifier. We are using HTTP/HTTPS with the named ports (1st parameter).
- The CDS identifier for the DCE application server the client wants to communicate with (2nd parameter).
- The name and password for the principal identity that the client is to assume (3rd and 4th parameter).
- The two numbers to add (5th and 6th parameter).

The Java class sets the DCE security to `DRPC_DCE_PROTECT_MAX_VALUE`, which provides maximum protection and privacy. Client-to-gateway security is provided on a best-effort basis, which means maximum SSL security for Web browsers supporting it and no security for the AppletViewer and command line Java.

```
# java Calc ″http:ev1.austin.ibm.com[1423,2423]″
/.:/Servers/SecurityServerID cio ci 1 2
drpc.dceLogin failed: ENC-dlt-0034
Error in call to Calc: java.lang.Exception: Failed to dceLogin: ENC-dlt-0034
java.lang.Exception: Failed to dceLogin: ENC-dlt-0034
        at Calc.identity(Calc.java:92)
        at Calc.main(Calc.java:72)
```

Figure 50. Java Command Line and Calc Login Failure

This simple test can be run on the gateway machine itself or on any client with the Java Development Kit installed. Figure 50 shows the first effort at a command line Java application accessing the math server through the DE-Light gateway. In this case we have provided an invalid password for the cio account. The login failure exception is caught by the Java source code.

```
# java Calc ″http:ev1.austin.ibm.com[1423,2423]″
/.:/Servers/SecurityServerID cio cio 1 2
The Calc Server said: 3
```

Figure 51. Java Command Line and the Calc Class

A successful result is shown in Figure 51. The protocol and address of the server is correct, and the CDS entry leads to successful contact with the math server.

6.3.6 AppletViewer and a Java Applet

The command line Java client demonstrates the simplicity of using Java as an application environment for DCE access. The DE-Light gateway technology provides a similar lightweight client technology for the C language. One unique feature of the Java environment is the ability to create Java applets to access DCE application servers. Java applets can be downloaded into any Java-aware Web browser providing a uniform, multiplatform client interface to DCE application services. The Web browser will automatically request the required DE-Light Java classes needed to interact with the gateway (co-resident with the Web server).

The simplest way to test Java applets is with the AppletViewer application provided with the Java Development Kit. It is the original HotJava HTML browser that first provided a Java-aware Web browser. An applet is encapsulated in some HTML code that has a syntax for specifying the environment that the applet should run in at the browser.

```
<title>Applet Calc</title>

<hr>
<applet code=CalcUI.class width=550 height=150>
<param name=Calc.gateway value=http:ev1.austin.ibm.com[1423,2423]>
<param name=Calc.dceServer value=/.:/Servers/MathPacServerID>
</applet>
<hr>
</html>
```

Figure 52. HTML Document Referencing the Calc Applet

Figure 52 shows the HTML used to hold our applet reference. The browser will request the named class from the Web server where this HTML document was obtained. Note the syntax for providing applet parameters. A complete description of the applet tag syntax can be found in the HTML V3.2 reference, available, for example, from `http://www2.wvitcoe.wvnet.edu/~sbolt/html3`.

The applet class `CalcUI.java` is listed in B.2.3, “Calc User Interface Class (CalcUI.java)” on page 164. It uses the `Calc` class described previously, but provides a graphical user interface (GUI) wrapper to be used within the applet.

The class has three simple parts. The first is an initialization section that is invoked when the class is run as a command line Java script. We will not be using this feature. The second is the initialization of the GUI defining each label and input field. It also obtains the applet parameters and creates an instance of the `Calc` class.

The AppletViewer is simple to invoke, but does require a `CLASSPATH` be defined to direct the viewer to both the applet Java classes as well as the DE-Light classes.

```
# CLASSPATH=.:/usr/lpp/Java/classes:/usr/lpp/DE-Light/lib/javaclasses
# export CLASSPATH
# appletviewer Calc.html
```

Figure 53 shows the AppletViewer window as seen under AIX 4.1 and the Motif window manager. Note the fields to input a login name and password, then the two values to be added. The password is echoed as hash characters.

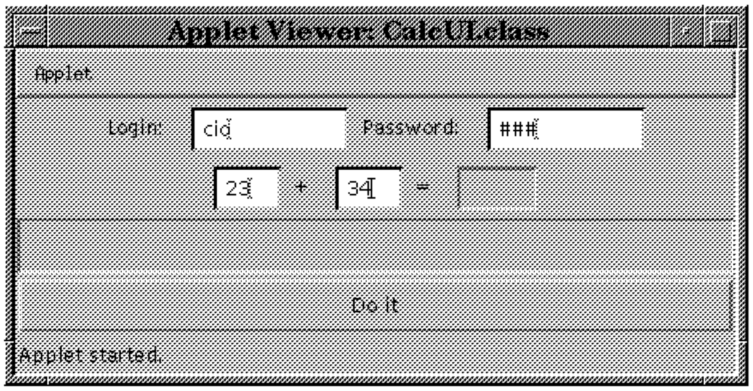
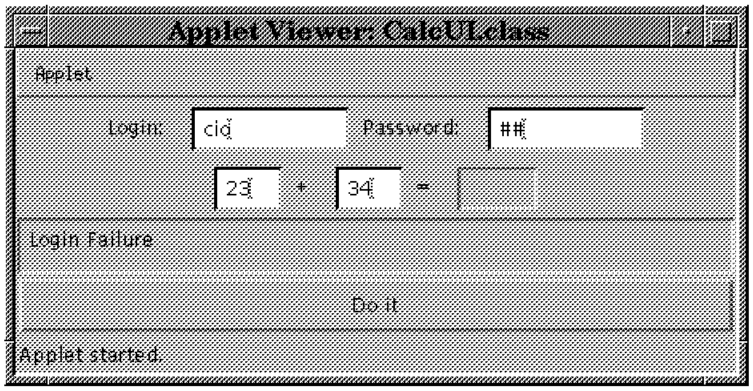


Figure 53. AppletViewer and the Calc Applet

Login failure is shown by an error message in the status window (see Figure 54 on page 128).

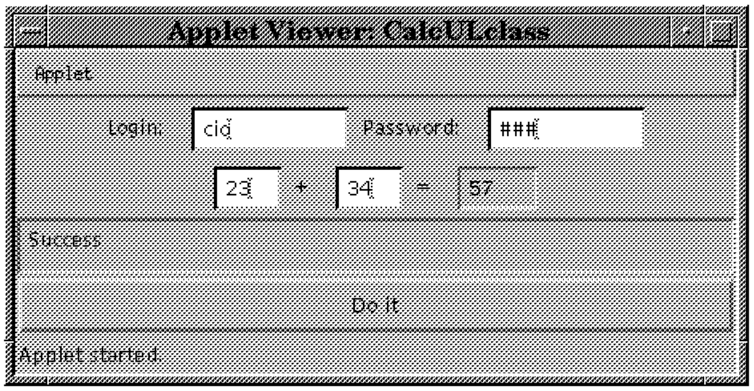


Login Failure

Applet started.

Figure 54. AppletViewer and Calc Applet Failure

Success results in another message in the status window, and the result of the computation is shown in the summation window (see Figure 55).



Success

Applet started.

Figure 55. AppletViewer and Calc Applet Success

The current `CalcUI` class only distinguishes success and login failure. Obviously, there are more potential error conditions that can occur from client-to-gateway-to-server communication. More intensive management of Java exceptions can be added to the `CalcUI` class with minimal effort.

6.3.7 Web Browser Java Applet

Applets accessible from a Web browser provide a powerful component in the client/server computing model. The Web browser is multiplatform, almost uniformly available, and light weight. Applets are downloaded on demand from the Web server, providing the latest software available. In the DE-Light environment they are also distinguished by supporting secure client to gateway communication. The SSL provides both privacy and integrity, which combines with gateway-to-server security to deliver end-to-end security from client to server.

It is important to remember that the DE-Light Gateway and Web server must reside on the same host because of the current Java Applet security restrictions about Applet host connections.

Another important point is the location of the DE-Light classes. These Java classes reside at the Web server and are automatically downloaded to remote Web browsers as needed. Since Java applets are interpreted at the Web browser, they import Java classes that provide the required functionality. Some of these classes will be core Java classes, such as the AWT (Abstract Windowing Toolkit) class or the Applet class, which are available locally to the Web browser because these come with the Java-aware Web browser software. Others will be requested from the Web server providing the Java applet. The Web server will search for these classes in the same location that the applet was found; so a link to the DE-Light Java class library should be included there. For example,

```
ln -s /usr/lpp/DE-Light/lib/javaclasses/COM COM
```

could be done in the applet directory at the Web server.

The Java applet in a Web browser looks similar to the AppletViewer interface (and should since they share the same code). Figure 56 shows the applet interface before calculation begins. These views are from Netscape Navigator running on OS/2 Warp.

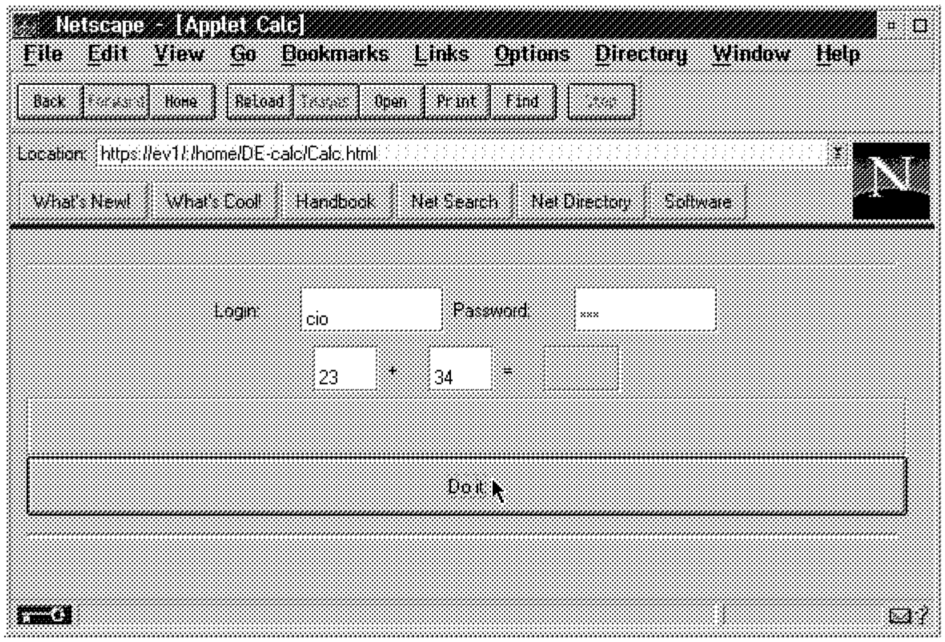

Netscape - [Applet Calc]
File Edit View Go Bookmarks Links Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: https://ev1/:/home/DE-calc/Calc.html
What's New! What's Cool! Handbook Net Search Net Directory Software
Login: cio
Password: ***
23 + 34 =
Do it

Figure 56. Web Browser and the Calc Applet

Login failure is shown in Figure 57 on page 130. Note that each of these screen shots show the browser operating in secure mode (the key in the lower-left part of the screen indicates this). This does not mean that the applet-to-gateway connection is secure through SSL but only that initial access to the HTML document referencing the applet was obtained from a secure Web server (see the https reference in the URL being accessed). The Web browser does not indicate whether there is a secure applet to gateway connection unless some indicator is placed in the user interface for the user. It might be useful to put a similar key symbol in the GUI for instance.

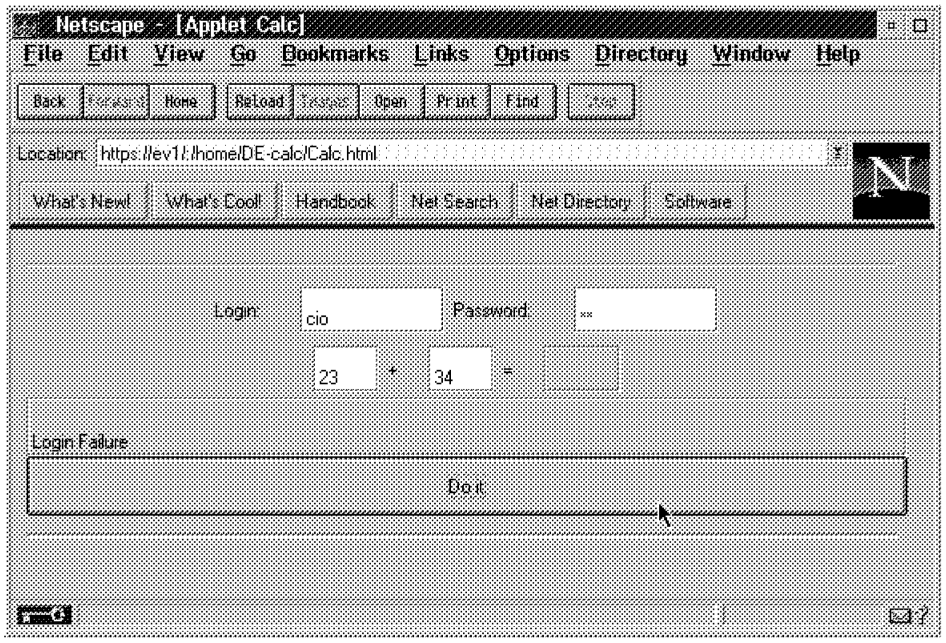

Netscape - [Applet Calc]
File Edit View Go Bookmarks Links Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: https://ev1/:/home/DE-calc/Calc.html
What's New! What's Cool! Handbook Net Search Net Directory Software
Login: cio Password: **
23 + 34 =
Login Failure
Do it

Figure 57. Web Browser and the Calc Applet Login Failure

Success is again similar to the AppletViewer result and is shown in figure Figure 58.

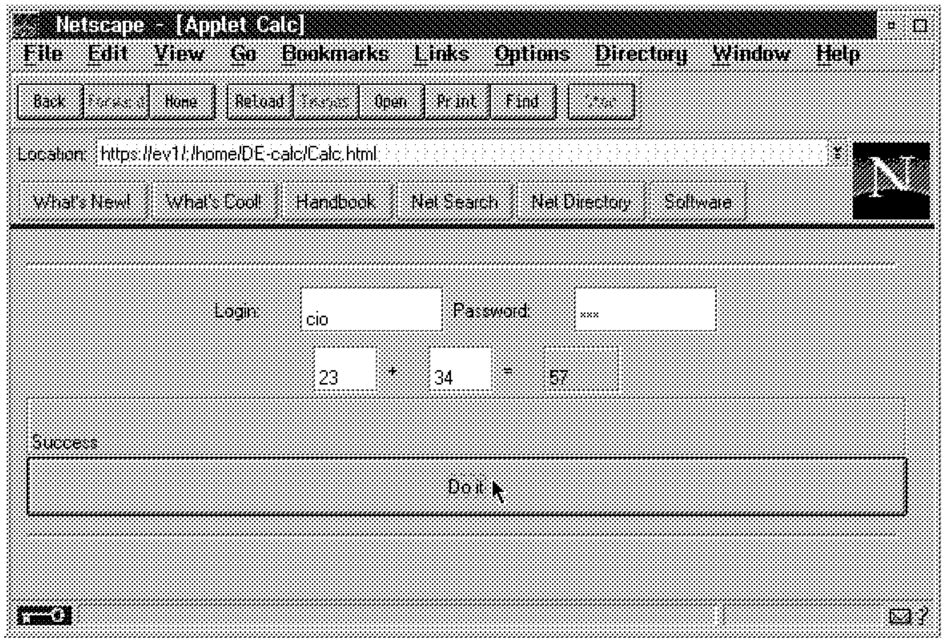

Netscape - [Applet Calc]
File Edit View Go Bookmarks Links Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: https://ev1/:/home/DE-calc/Calc.html
What's New! What's Cool! Handbook Net Search Net Directory Software
Login: cio Password: ***
23 + 34 = 57
Success
Do it

Figure 58. Web Browser and Calc Applet Success

Since these applets were run within an SSL-aware browser and the Java source code attempted to use the best client to gateway security available, we can assume they have been using a secure communication connection. It is possible to verify this by watching the network traffic or by putting the client or server into trace mode (see section 6.4, “Problem Solving” on page 131).

6.4 Problem Solving

Sometimes all does not go as planned when developing Java applications in a DE-Light Web environment. Here we include some tips on avoiding and finding problems.

- The Netscape Navigator browser provides a Java Console for displaying output from your Java Applet to the normal output channels. This is a very useful way to pick up debugging information such as client traces.
- We have previously seen that the gateway can be put into a trace mode (see 6.2.3, "Running the DE-Light Gateway" on page 121). This will provide a running commentary on exactly what the gateway is doing. It is also useful for verifying SSL-compliant communication. The simplest method for doing this is to add `-T all=STREAM:stdout` to the command line starting the gateway.
- Java clients can also be put into trace mode. Before any calls to DE-Light Web services, the following method should be called:

  ```
  SERVICE.SETDEBUGLEVEL(5);
  ```

  It will cause trace information to be output to either the Java Console in a browser or to the standard output device for command line Java.
- When developing Java-based applications for use with DE-Light Web, begin with a command line program invoked through the Java interpreter, progress to the AppletViewer, and finally unleash the Web-browser-based client. This will allow you to fix the easiest problems first and isolate the cause of problems more easily.
- Use the most general exception catching you can in your Java classes. All exceptions should be caught and handled. Failure to do so could result in unexpected behavior by the applet. It is better to be on the safe side, rather than to spend too much time for debugging.
- It is possible to test the HTTP/HTTPS protocol support by the gateway without using real Java clients. If the gateway was started with the following protocol selection,

  ```
  -e http:[1423,2423]
  ```

  use a Web browser and directly connect to the URL

  ```
  http://ev1.austin.ibm.com:1423
  ```

  for the HTTP portion of the protocol, or use

  https://ev1.austin.ibm.com:2423

  for the secure HTTPS side. Both connections should result in a message similar to the following from Netscape Navigator

  ```
  Error 404
  Request is not supported by this server.
  ```

  if all is operating correctly. The HTTPS connection will display the secure connection information windows before displaying the error message. If the gateway is not working correctly, the Web browser will pop up an error window indicating a network error has occurred and the Web browser was unable to contact the server. For example, Netscape Navigator displays an error message with

```
A network error occurred:
  unable to connect to server
The server may be down or unreachable.
Try connecting again later.
```

This is a quick test to see if a problem involves communication to the gateway.

Chapter 7. Alternate Methods for Using DCE on the Web

The Distributed Computing Environment (DCE) is a secure, enterprise-wide middleware environment. Its strengths lie in a robust and scalable architecture that brings together all parts of an organization under a common, secure interaction environment. Interoperability spans from desktop to mainframe, interconnecting all within an enterprise intranet.

The World Wide Web has had an enormous impact on the way we share information both within an organization and more traditionally between organizations. In the past this has often relied upon a relaxed security model, but as more organizations open their intranets to external visitors, more emphasis is being placed on strong security policies and mechanisms. Firewalls (see Chapter 8, “Bridging Networks Together: DCE Over Firewalls” on page 143) have played a part in this revolution. They restrict access to a network by limiting access and providing auditing facilities to track problems.

A key product linking the semi-structured information accessed through the Web with the more formally structured information in the DCE space is DFS Web. It links the secure world of DCE and DFS to the Web Servers using server plug-in technology. Full DCE security semantics are available to the Web browser as mediated by the Web server plug-in. We have already explored using DFS Web security management to allow the direct execution of privileged DCE client applications as CGI scripts. Alternatively, DE-Light provides DCE application server access through Java-based clients using a DCE-aware gateway as an intermediary.

In this chapter we explore several alternative methods for integrating DCE/DFS and the Web. One of the more obvious candidates is to use the local document access methods available through a browser. Combine this with a local DFS runtime, and the client has access to any DFS-based Web document which it has permission to read. Another alternative at the browser side is the use of browser plug-ins to support DCE. We discuss the difficult issues associated with such a plan. Finally, we look at DSSL for Lotus Domino, a toolbox enabling DCE access through the Lotus Domino Server. The simplicity of the LotusScript programming environment is combined with the power of enterprise application access in a simple development environment.

7.1 Native DFS Exploitation on the Browser Side

DCE’s Distributed File System is a robust, secure, and scalable solution to information access both within an enterprise and through secure intercell links to clients external to the enterprise. It abstracts location information from naming, allowing files to be moved, and possibly replicated, without concern to client access. Individuals can offer sets of files to the DCE environment through appropriate access control specification in combination with intercell agreements between cells (or enterprises).

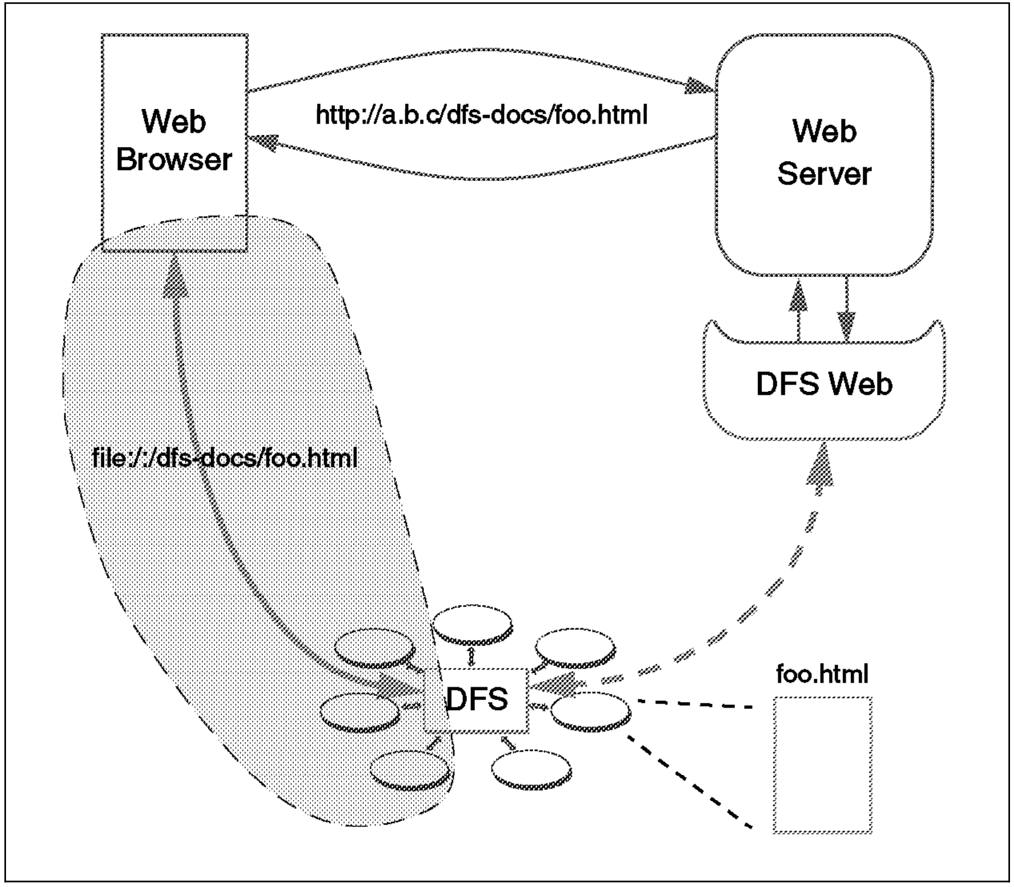


Figure 59. Exploiting DFS Access with the Web Browser

We have already seen these benefits exploited through IBM's DFS Web Secure. This plug-in for a Web server manages file access by the server to a DFS filespace. DFS file access control is correctly obeyed by requiring clients to authenticate with the DFS Web plug-in before accessing documents stored within DFS. The same access controls are used in authenticated Web document access as are used natively within a DFS client/server environment. The Web document space can now be completely distributed providing both robust access through the use of DFS replication and security by way of DCE security. Figure 59 shows this access path from Web browser to Web server, through the DFS Web plug-in, and to the DFS Web file space. This solution builds on an existing enterprise DFS solution, but can be exploited by both the enterprise intranet community as well as a broader Internet community. The browser can be anywhere in the world without knowledge of DCE or DFS.

There is a simpler solution to accessing documents in the DFS filespace if we limit client browsers to the intranet/cell space. We assume that the local browser host includes a DCE and DFS runtime environment. A user authenticates with DCE, then invokes their local browser under this authenticated user environment. If DFS documents are accessed as local files, then a similar DFS document space is available to the user as was accessed through the DFS Web plug-in at the server. Figure 59 shows this alternative access path in the lower-left shaded area.

No Web server is involved in the document retrieval if direct file access is performed. This has the benefit of one less piece of software being maintained, but at the same time, no logging, event triggers, or document access statistics are available. If links within the document use relative naming conventions, such as

```
<A HREF="overview/ovr-toc.html">The Overview</A>
```

instead of absolute references, such as

```
<A HREF="http://a.b.c/overview/ovr-toc.html">The Overview</A>
```

then subsequent document access will stay within the local DFS filespace. Obviously, this solution only makes sense for browsers with access to DCE/DFS and either in the enterprise DCE cell or linked to it through a trusted cell. Security is identical to DFS file access security from any other application and requires no intermediary such as DFS Web to translate.

Direct file access through DFS as described in this section is probably not an adequate solution for large-scale Webs due to its requirements for relative links (or absolute file links) and the lack of tracing and logging facilities. But it might be an ideal solution for department-wide Web exploitation where individuals want to selectively share information with others—even used in addition to other methods, like DFS Web Secure. Provided the document or file links are set up accordingly, this easy can also be used in conjunction with NFS to DFS gateways, which removes the requirement for having local DCE and DFS code on the browser host.

7.2 Browser-Side Plug-Ins

Browser plug-ins (see also 3.1.2.2, "Web Browser" on page 33) are designed to extend the browser by providing additional functionality and compatibility. Plug-ins are available for Web browsers, such as the Netscape Navigator, across a range of application areas, such as:

- Multimedia viewers including audio, video, and support
- Utility functions such as object imbedding and compression/decompression
- Applications such as games and information managers

A complete description of available Netscape Navigator plug-ins is available at:

```
http://home.netscape.com/comprod/products/navigator/version_2.0/plugins/index.html
```

Plug-ins form a very close relationship with the browser. They come in the form of loadable modules that are instantiated as needed. Figure 60 on page 136 illustrates the relationship between browser, plug-in and Web server. The browser recognizes a special HTML tag, `<EMBED> </EMBED>`, which includes a number of attributes. Most important of these is a MIME (Multipurpose Internet Mail Extensions) attribute that specifies the variety of information included in the embedded data. The browser is created knowing a set of available plug-ins and their supported MIME types.

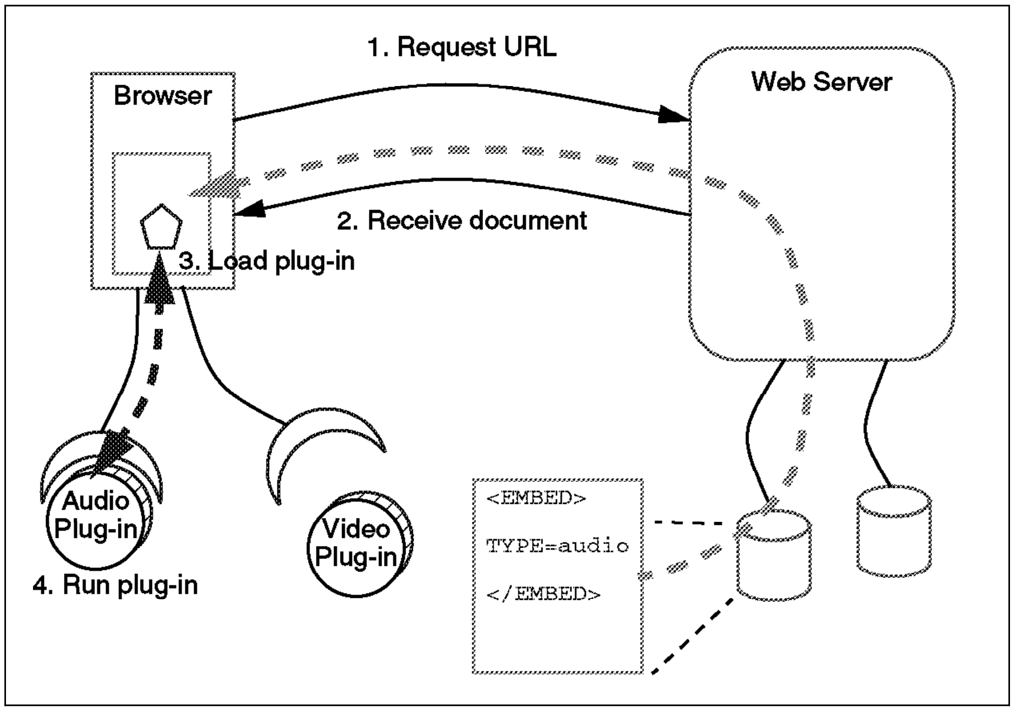


Figure 60. Using Netscape Navigator Plug-Ins

Initially, a plug-in is allocated a piece of screen real estate for which it is responsible. The plug-in and the browser communicate through a well-defined API that brings data to the plug-in and sends it to the browser. Key elements in this relationship are events and streams. Events include mouse events that link into the availability of stream data. Plug-ins must be extremely careful with any synchronous activity as they can easily block not only other plug-ins but also the browser itself. Currently, no plug-in technology is available that integrates DCE or DFS security with a Web browser. A similar browser-side functionality is provided, for example, by Gradient Technology's NetCrusader product. It provides the browser with a local Web server proxy that is able to use an additional DCE-based interaction protocol to provide secure communication within a NetCrusader environment.

The DCE plug-in module concept is searching for alternative methods to link the browser to DCE. The DE-Light product from Transarc Corp. provides one such mechanism. Another alternative would be to implement DCE-compatible Java classes directly. We can rule this option out rather quickly. The current security restrictions applied to Java applets limits how many network connections an applet can make and to whom the connections can be. Direct DCE access would require access to several hosts providing the Cell Directory Service, the Security Service, and the DCE application server itself. In addition, the DCE protocol is complex and large. Java classes supporting the full protocol would be too large for useful network transport.

A plug-in module is a complex piece of code. The C-language API is written with primitive data manipulation in mind, such as video and audio support. It is not something to be undertaken lightly. To provide a generic DCE plug-in library would require a significant piece of work to support the dynamic RPC building as found in the DE-Light gateway. The DCE runtime API does not currently provide support for dynamic specification and execution of RPC calls to interfaces unknown at compile time. An application-specific DCE plug-in module is more feasible, but again would be a large piece of work. It would not only require DCE interaction calls but also a user interface, or at least user interaction, to support the application.

Currently, a DCE plug-in module is not feasible except for application-specific tasks where DCE becomes another part of the plug-in environment requirements.

A look through the current list of plug-ins for the Netscape Navigator can give you a feeling for the effort required to provide plug-in functionality. A list of available plug-ins is available at
http://home.netscape.com/eng/mozilla/3.0/handbook/plugins/pguide.htm

7.3 Distributed Systems Series Link for Lotus Domino

Lotus Domino (Notes) is an enterprise groupware solution bringing together an organization under a common interaction infrastructure. It supports the traditional components of groupware, namely communication, collaboration, and coordination through a number of components including shared databases, electronic mail, scheduling tools, and database access. The Lotus Domino Server extends the Notes environment to include Internet access through Web browser clients. Traditional Notes forms are dynamically translated into HTML and displayed at the browser. Ubiquitous browser technology can now play a full part in the business workflow process.

Distributed Systems Series Link for Lotus Domino (DSSL for Lotus Domino) adds yet another dimension to Domino by providing access to DCE application servers. The users and Domino administrators benefit from the simple LotusScript programming language used by Notes/Domino to develop integrated applications bringing together enterprise DCE solutions with desktop workflow functionality. Full advantage can be taken of DCE security and flexibility while using either Notes client or Web browser technology. In either case the renowned complexity of DCE programming has been encapsulated and presented to the LotusScript programmer in a simple, seamless package.

DSSL for Lotus Domino is available from the IBM Developers Connection Program (also called DEVCON) which offers a variety of tools and documentation on CD-ROM and on the Web at http://www.developer.ibm.com/devcon/.

7.3.1 The Lotus Domino/Notes Architecture

Lotus Domino is a de facto standard in the field of *groupware* technology. It facilitates organizational interaction through a common integration framework. Specifically these facilities cover three important areas.

- Electronic mail providing *communication* both within the intranet and Internet
- Calendaring and scheduling facilities encourage group *collaboration*
- Access to organizational data to *coordinate* business processes

Lotus Domino provides these solutions over corporate-wide, distributed databases and integrates with existing office solutions. The Domino architecture consists of traditional client/server technology deployed as Domino servers and Notes clients (see Figure 61 on page 138).

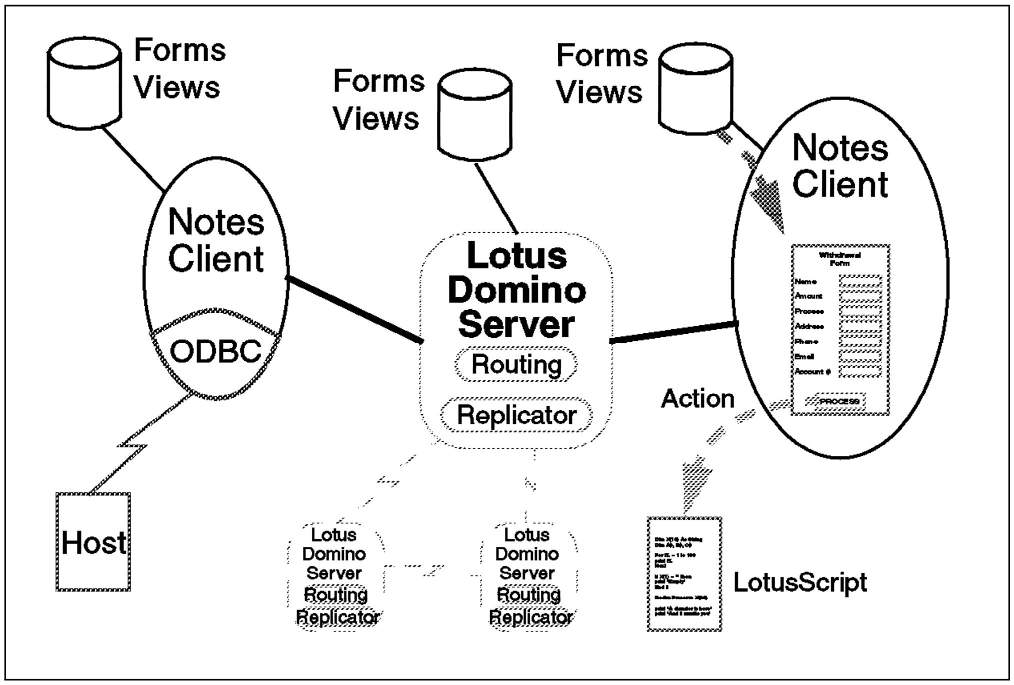


Figure 61. Lotus Domino Architecture

The Domino server coordinates the distribution of information between Notes clients and also to other Domino servers that may contain replicated information. The Domino server has a number of components, such as routing for electronic mail distribution and a replicator for managing data consistency among servers. Lotus Domino data is stored in *databases* that can be thought of as containers for information sharing. Data is entered into these databases through Domino *forms* and examined through *views*. Each performs some data massaging, such as checking of input or restructuring output.

LotusScript is the language through which components of Domino are glued into a useful organizational tool. The language is very similar to Visual Basic and is simple enough to be within reach of almost any Domino user. For example, a few pages of LotusScript can take data from a form, feed it into a shared database, inform a list of interested colleagues by electronic mail, and present any new information back as a view.

Lotus Domino also incorporates facilities to tie into existing information systems such as databases. These facilities are often themselves stored in Lotus databases and downloaded into a Lotus client when needed. For example, database records can be accessed from a LotusScript program through an Open Database Connectivity (ODBC) module.

Lotus Domino Server enhances traditional Lotus Notes services by integrating Internet facilities into the product. In effect, Domino provides a Web server within the Domino server (Figure 63 on page 142 illustrates this service), listening on the same well-known communication port as any other Web server. Domino will dynamically translate Lotus forms and views into HTML form code giving Web browsers access to the power of the Lotus Domino/Notes groupware environment.

The Lotus Domino environment has three important features that make it very applicable to organizations.

- The Domino architecture provides a common *integration* framework for groupware tools that is both powerful and *extensible*.
- The LotusScript programming environment is *simple* to use and understand yet powerful enough to integrate with the diverse set of tools available within the groupware environment. The task of gluing in place a new piece of functionality can be accomplished by frontline workers rather than relying on the skills of experienced programmers.
- Web browsers can access Lotus resources in the same way as a dedicated Lotus client. As with all *browser-based clients*, they benefit from Web features such as remote access, security, and open interworking.

7.3.2 DCE/DSS Integration with Lotus Domino

DSSL for Lotus Domino is a tool used to integrate Lotus Domino, the Distributed Computing Environment (DCE), and DCE applications. It builds on the strength of Lotus Domino to provide a simple yet powerful environment for the development of clients that benefit from the existing DCE application servers and the integrated groupware environment. DSSL for Lotus Domino makes full use of DCE security and directory services while minimizing the traditional complexity of DCE programming. Figure 62 illustrates the architecture integrating DCE and Lotus Domino.

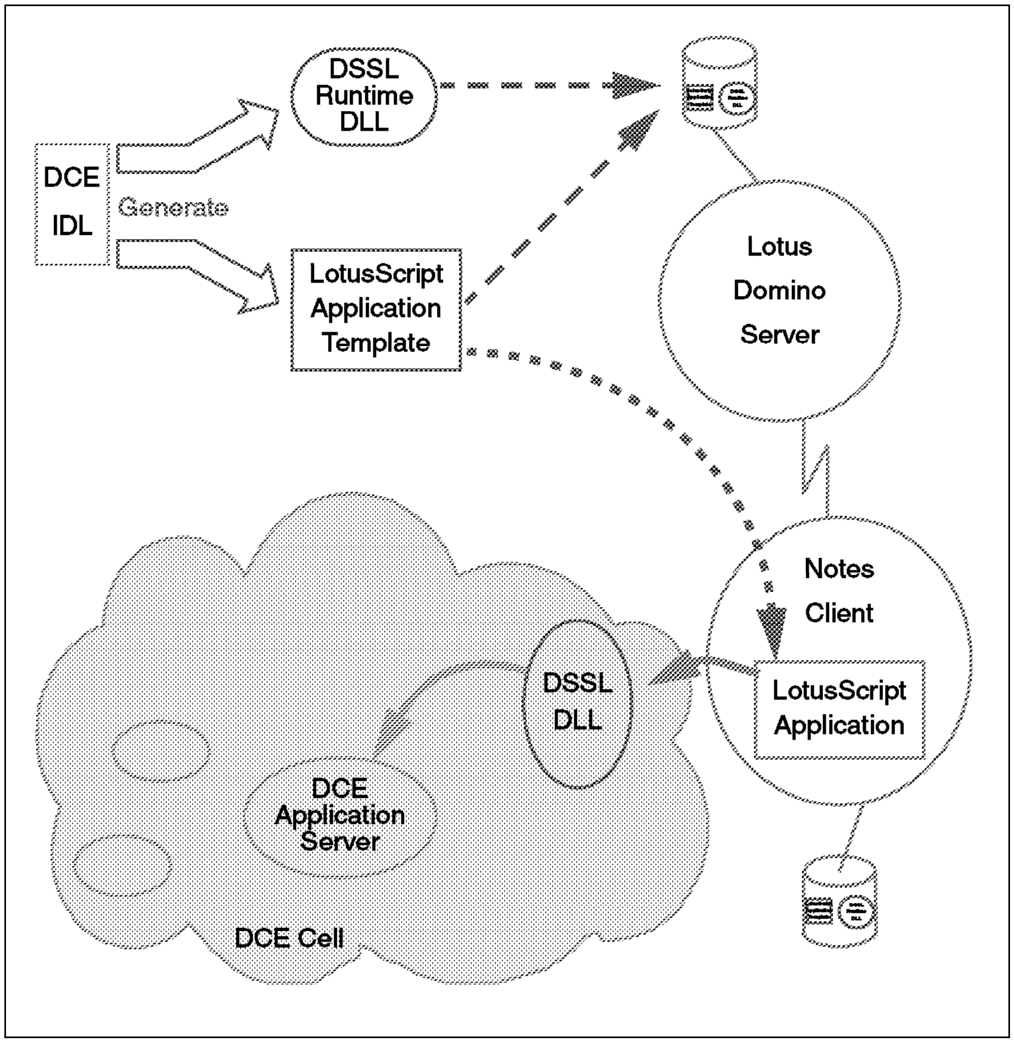


Figure 62. Integrating DCE and Lotus Domino

DSSL for Lotus Domino integrates into both the DCE and Lotus Domino environments by providing the seamless access to DCE facilities that would be expected through LotusScript. The author sees no difference between local LotusScript procedures and those of a remote DCE application server. The development environment is currently available on five platforms.

- OS/2 Warp, IBM DCE Directory and Security Service (DSS), IBM VisualAge C++ and Lotus Domino Release 4.5.
- Windows 95, Gradient DCE Versions 1.0.3, 1.1 and 2.0, Microsoft Visual C++ and Lotus Domino Release 4.5.
- Windows NT, Gradient DCE Version 1.0.3, 1.1, and 2.0, Microsoft Visual C++ and Lotus Domino Release 4.5.
- Windows 95, IBM DCE 1.1, Microsoft Visual C++, Lotus Domino Release 4.5.
- Windows NT, IBM DCE 1.1, Microsoft Visual C++, Lotus Domino Release 4.5.

The runtime environment requires a similar setup, but no compiler is required.

The graphical development tool takes a DCE IDL file and generates a dynamically loadable *runtime support library* (DLL) and a LotusScript *application template* from which the Lotus client can be developed.

- The DLL is compiled from C source code generated by the development tool and standard DCE stub files provided by the native DCE IDL compiler. The DLL provides a gateway for access to the server described by the provided DCE IDL file.
- The application template is a LotusScript application providing a starting point for the LotusScript programmer to develop the required application.

Both the runtime library and application template are loaded into a project and stored in a Domino database. They are made available to the Domino community just as other information is shared within the groupware environment.

7.3.3 Secure DCE Access Through a Web Browser and Lotus Domino

The Lotus Domino Server introduced Internet access to the Domino groupware environment. This extended the Domino realm to include Web browsers as client access points, but also integrates the Web browser as a business activity into the existing workflow support environment within Domino.

The combination of DCE access, a Web browser and Domino provides a powerful organizational integration environment. DCE delivers security and flexibility through an open standard, thus providing interoperability and cross-platform support. The Web provides the equivalent standard solution at the client side with a cross-platform, network-aware interface that is regularly extended by industry. Lotus forms the organizational architecture bringing together the user interface with information and service sources such as databases, DCE and Domino components. LotusScript is the glue for this architecture.

We have already discussed both DCE and the Web as Domino integration components. We now turn our attention to an example of such integration with respect to security. Figure 63 on page 142 tracks access to a DCE application server through a Web browser.

1. The process begins back at the DSSL for Lotus Domino development environment where the same template application code used with the Notes client can be used within the Web browser model. The code checks which of the clients the access is coming from and determines how the parameters to the form are to be obtained. This step is not shown in Figure 63 on page 142.
2. If the Domino form is protected by a Domino access control list (ACL), then the Lotus Domino Server returns a ″401 Error″ or an access restricted warning to the browser asking for a username and password.
3. The username is checked within the Domino name/address database, and a DCE compatible name is obtained.
4. The Domino form is now dynamically translated into HTML and presented to the Web browser just as HTML would be from a dedicated Web server.
5. The HTML form is passed to the Lotus Domino Server agent.
   - The DCE user identity is passed through the DSSL for Lotus Domino runtime (5a) and verified as a valid DCE login based on a known DCE keytab file associated with the name. DSSL for Lotus Domino remembers the credentials associated with the login and to be used with the DCE application server access.
   - The server agent hands control off to the DSSL for Lotus Domino runtime (5b) generated by the DSSL for Lotus Domino development environment.
6. The call is made by the DSSL for Lotus Domino runtime to the DCE application server and the result passes back to the Lotus Domino Server agent, which then displays the results as an appropriate Domino view in HTML.

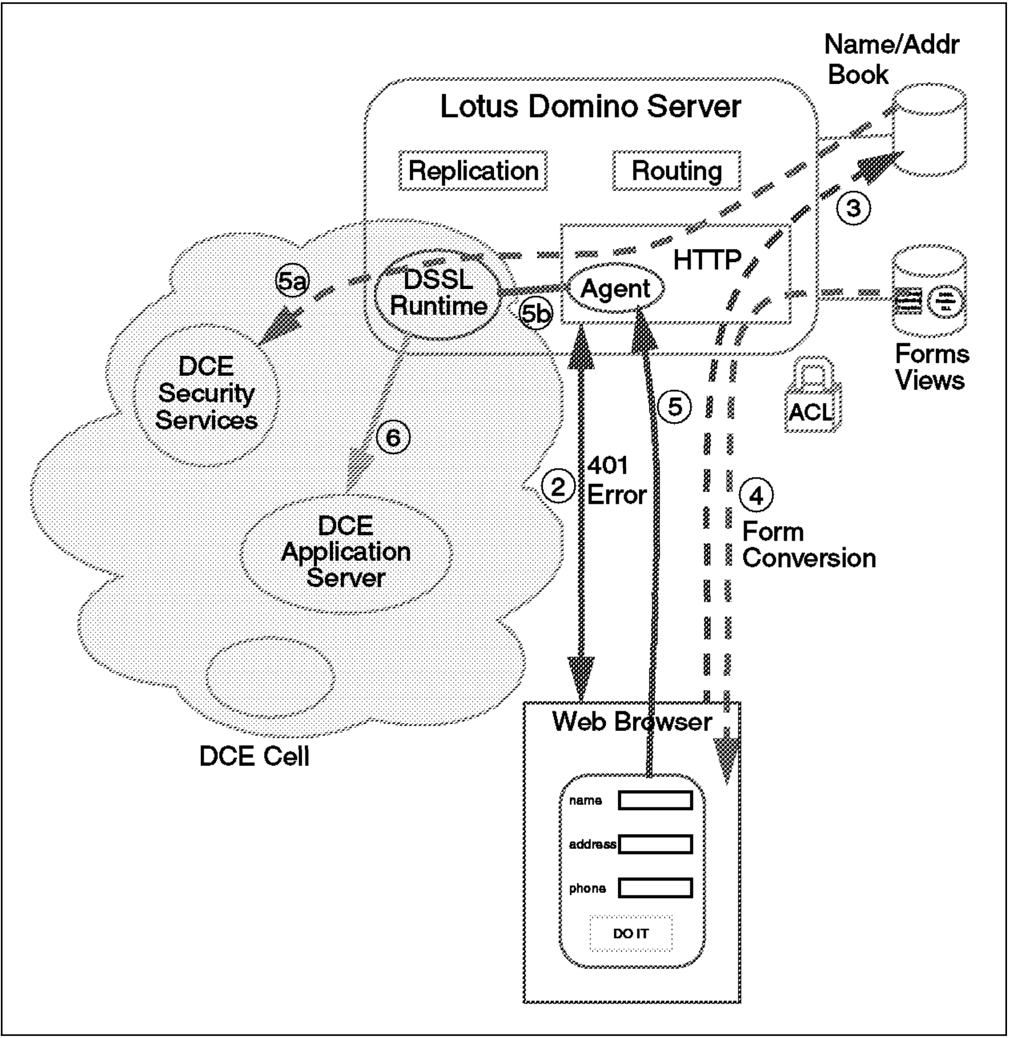


Figure 63. Lotus Domino, DCE and the Internet

DSSL for Lotus Domino enables users to connect from the Internet with a Web browser to access the full range of Domino information and services including transaction services, databases, and existing enterprise DCE applications. Users are spared almost all of the complexity of programming, and instead, the task of creating enterprise applications can fall with those who have an intimate understanding of the business processes and not just a technical perspective. DSSL for Lotus Domino is based on an open standard, thus providing secure and flexible access to an extensible corporate infrastructure. Domino is the glue in this system that enables scalable, manageable, and secure solutions to be built.

Chapter 8. Bridging Networks Together: DCE Over Firewalls

As a company network is bridged to the Internet, or even to an organization intranet, it is important to be able to control and monitor the traffic across these network bridges. The risk of hackers entering a company network from the Internet and wreaking havoc on systems attached to that company network is one obvious reason for setting up such controls on network traffic. A firewall is a vital component for enforcing these controls. In this chapter we look in particular at the requirements for running DCE core services and DCE applications across firewalls.

8.1 Principles of a Firewall

The principal function of a firewall is to control traffic flow between networks. In order for a firewall to be able to achieve this control, it must have the following properties:

- All traffic between the networks must pass through the firewall.
- Rules defining authorized traffic can be defined to the firewall and enforced by it. In other words, any traffic not specifically authorized according to these rules must be blocked by the firewall.
- The firewall itself, if it is running as a package on top of an operating system, should be secured from unauthorized access.

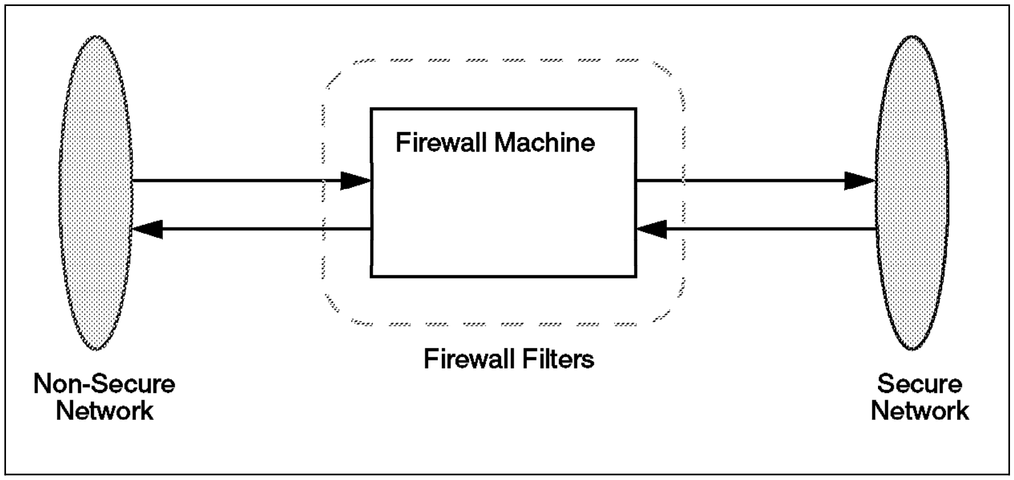


Figure 64. Schematic of a Firewall

If these aspects roughly define a firewall, then how does a firewall actually impose the control that is required? The main techniques that firewalls use are:

- Filter all TCP/IP packets arriving at the firewall based on key information such as source and destination IP addresses, port numbers, ans so forth. Figure 64 shows a schematic of a firewall using filters.
- Run application-level servers, often known as proxy servers.
- Carry out a stateful inspection of the packets arriving at the firewall. This involves taking a detailed look at the packet contents and assessing whether the context information (such as the user, the application and the protocol) is appropriate for the packet to be passed on. This is a very powerful firewall

model that will place corresponding demands in terms of system performance.

Before looking at these techniques in more detail, it is important to note that just installing firewalls is not a sufficient security policy to make it safe to attach a network to the outside world. Firewalls cannot secure traffic that does not pass through them, and it is also vital to secure administrative access to any firewall. This means securing the underlying operating system if the firewall is a package running on top of a standard operating system such as AIX. Implementing firewalls is also not a replacement for sound security measures on the systems comprising the ″secure″ network that is being connected to an external network through the firewall(s).

8.1.1 Packet Filtering

The most commonly used firewall approach is to set up filter rules on an IP router inserted between the secure and insecure networks to govern all the network traffic passing between the two networks. The firewall checks each IP packet against the filter rules before passing (or not passing) it on to its destination. The more attributes the filtering rules can check on the better. Important attributes include:

- Source and destination IP addresses
- The IP protocol
- Source and destination TCP and UDP ports
- The interface where the packet arrived (secure network interface or insecure network interface)

The advantages of packet filtering are fast performance and the fact that the end-users are unaware of the presence of the firewall (provided they are using permitted channels of communication across it) because they can use their standard client programs. The disadvantage is that there are ways for mischevious users to send packets pretending to originate from ″trusted″ IP addresses (IP address spoofing) and pass through filters. Filters will also be unaware of what is happening at an application layer where you may want to impose some further control. Nevertheless, they are a good starting point for implementing security on network connections.

A firewall implementing packet filtering on a router to operate at the network level is sometimes also referred to as a *Screening Router*. Packet filtering can be used to control (pass or deny) DCE traffic through a firewall.

8.1.2 Application (Proxy) Servers

To impose more control on what is happening at the application layer, it may be possible to use an *application* or *proxy server* tool. For a specific application, this means actually breaking the user's access at the firewall system. The proxy application at the firewall may be a more restrictive version that performs some extra verification and logging before carrying out the user's intended connection to the application beyond the firewall. An example is a proxy Telnet server shown in Figure 65 on page 145. Here, a user uses standard telnet to login to the firewall and from there uses Telnet to login to a host on the non-secure network. The security advantage is that the firewall can now have stricter filtering rules to prevent all Telnet traffic passing through the firewall, and information about the source host of the telnet session is not passed onto the non-secure network (where all accesses appear to be originating from the firewall system).

Although the user now has to make a double connection, which takes more time and impacts performance, the user is still able to use a standard Telnet client program. Sometimes with this approach, the particularly secure gateway system that users must log on to before accessing other systems and services is termed a *bastion host* or a *gateway host*.

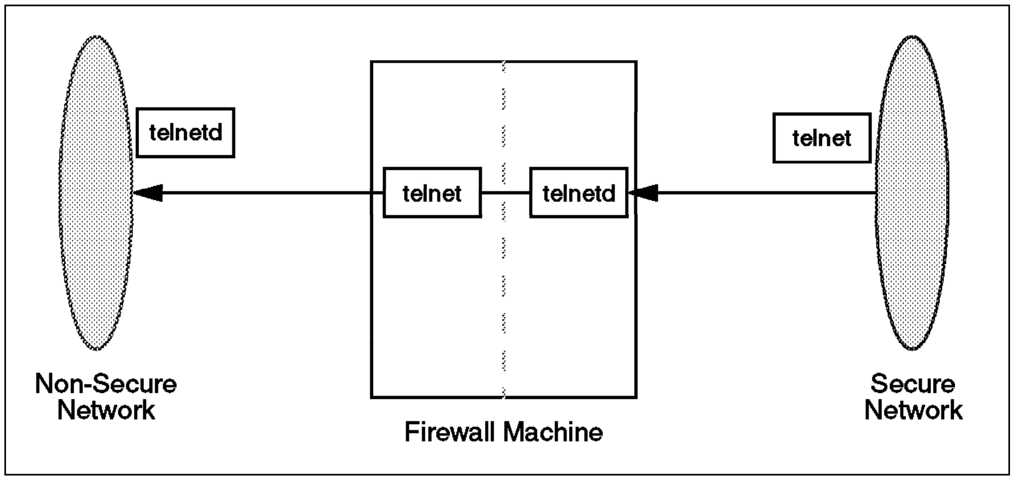


Figure 65. Firewall with Proxy Telnet Server

A proxy server needs to be obtained for each specific TCP/IP-based application. There are several available for applications, such as FTP, HTTP, NNTP, Gopher and X11.

The most common use of a proxy server is for Web connections using HTTP. All Web browsers on the secure network connect to the proxy server, which in turn forwards these requests to the final destinations on the non-secure network. Responses are then routed back to the originator on the secure network. This offers an additional advantage of caching often referenced information in the firewall machine. Besides the necessary configuration on the Web browser, the user is not aware of this double connection.

The DCE environment is made up of several well-known services as well as a potentially wide range of custom client/server applications. The dynamic, and often peer-to-peer nature of the TCP/IP traffic between DCE clients and servers is probably why there is no DCE proxy server available. In a sense the DE-Light Web product from Transarc provides a kind of proxy server for accessing custom DCE applications—where a Java client's connection to a DCE application server is split at the machine that is running the DE-Light gateway (as well as the full DCE client). DE-Light Web is covered as an example in 8.2, "Practical DCE Scenarios" on page 147, and in Chapter 6, "DE-Light Web Client" on page 109.

8.1.3 SOCKS Server

Another application-level approach on firewalls is to use *SOCKS*. In the same way that we saw with proxy servers, the application session is broken at the firewall. With SOCKS, however, there is no need for a special application server on the firewall with the user needing to perform a double connection. On the other hand, the user does have to use a special version of the application client that is SOCKS aware (called a SOCKSified client), and there needs to be a configuration profile in place on the firewall for the generic SOCKS server to allow the user's intended access. A schematic of the SOCKS server is shown in Figure 66 on page 146. SOCKS is generally favored for outbound TCP traffic from the secure network. It is not suitable for DCE client/server communication because it would require special, SOCKSified versions of DCE core services and applications, and there are no SOCKS servers for DCE available.

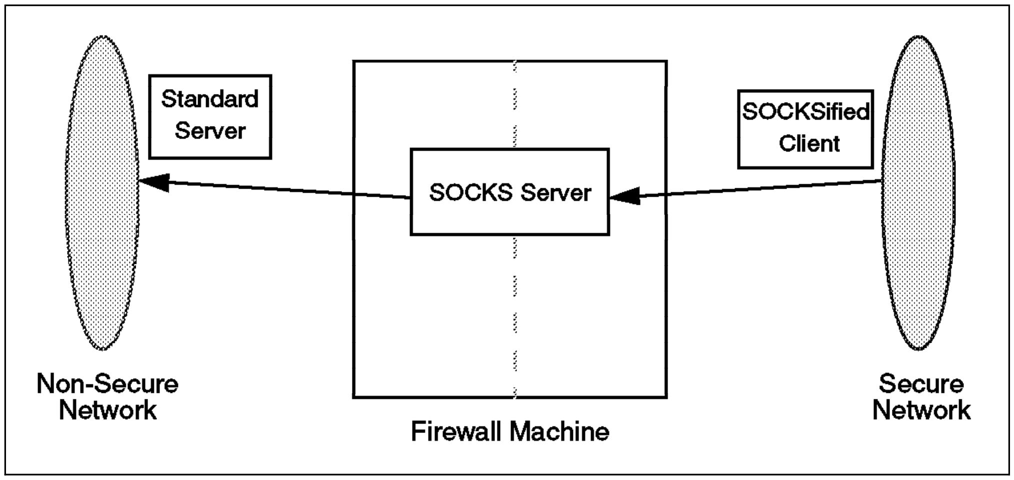


Figure 66. Firewall with SOCKS Server

8.1.4 Demilitarized Zone

Another firewall approach that combines some of the approaches covered above is the *screened subnet* (sometimes referred to as a *demilitarized zone* or *DMZ*), which is depicted in Figure 67. Here, a small network is set up between the secure network and the external network with traffic between the two blocked by filtering. The DMZ, which is accessible from the secure and the external networks, contains machines that perform as gateways for specific applications. Using separate machines for each major application can improve integrity further, if the additional costs can be afforded.

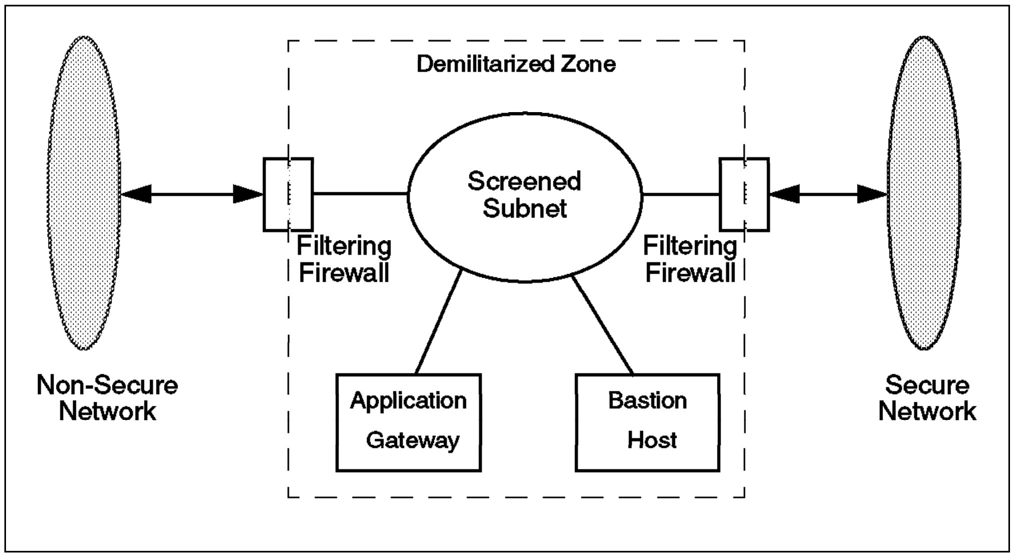


Figure 67. Screened Subnet or Demilitarized Zone (DMZ)

8.1.5 IP Tunnelling

Some firewall products support the setting up of secure IP tunnels. This approach allows the setting up of a private communications channel between two private networks across a non-secure network, such as the Internet. With compatible firewall products at each end of the tunnel, traffic for the full range of IP applications can pass across the Internet and be obscured from eavesdroppers by encryption. This approach could be used to set up secure links across the Internet between the DCE cells of collaborating organizations. This does require a specific setup for each organization and compatible firewall products at each end of the tunnel. For this reason, it is not an approach for allowing widespread Internet client access to DCE services. Some firewall approaches that can be used for this type of access are covered in the following sections of this chapter.

Whichever firewall approach (or combination of them) is adopted, it is very important to enable the logging functions within the firewall(s). A security breach which has passed through the firewall can then be investigated. If attacks occur, it is better to know about them in order to be able to improve the configuration of the security controls—such as firewall filtering rules.

8.2 Practical DCE Scenarios

The following two sections describe two typical environments where firewalls bridge DCE environments together. We look at a more general implementation using packet filtering first, and another example using a DE-Light gateway in a demilitarized zone is described in the second example.

8.2.1 Using Packet Filtering for DCE Traffic

For the first example, we consider how to set up the packet-filtering rules required on a firewall when placing a Web server with DFS Web Secure (serving files in DFS) across a firewall from the DCE cell. The requirements for this example match those of many publicly accessible Web servers that are placed outside firewalls that are protecting companies' internal networks. In this example with DFS Web Secure, we need to have a full DCE/DFS client running outside the firewall and still able to communicate fully with DCE/DFS servers on the internal network across the firewall.

Even a basic DCE client interacting with the DCE core servers in a DCE cell involves TCP/IP traffic over several TCP and UDP port numbers. When DCE application servers are also up and running, the number of ports used by DCE is going to increase further. This means that the filter rules will have to open more than just one or two ports in the way that filter rules are set up for FTP and Telnet. Also, the port numbers used by DCE, apart from port 135, are not well-known ports, since they are dynamically allocated above 1024. Our aim is to open up the minimum number of ports through the firewall to enable the DCE and DFS clients to run.

In order to have some control over the dynamic ports that DCE services use (even for the base services such as a DCE login), it is necessary to set the environment variable `RPC_RESTRICTED_PORTS` before starting DCE services. This needs to be done on all the DCE server systems inside the cell that our DCE client outside the firewall is likely to communicate with. This will be at least the CDS server, the Security server and the DFS servers. Here is an example of how to set this variable:

```
RPC_RESTRICTED_PORTS=ncadg_ip_udp[6500-6550]:ncacn_ip_tcp[5000-5050]
```

This environment variable tells DCE to use only ports 6550-6550 for UDP, and 5000-5050 for TCP protocols, respectively. A convenient way to set this variable for all DCE processes is to put it in the /etc/environment file on AIX. For this example, we used the variable as shown above with a range of fifty ports for each protocol type, although in a larger production DCE environment these range sizes are likely to be too restrictive. A range of a hundred or more ports for each protocol would be more practical.

The IP filtering rules in this example are defined with the following format, which is the one used by the *IBM Internet Connection Secured Network Gateway* firewall product (SNG) for AIX. Other firewall products will allow filter rules to be defined in a similar way using this type of information relating to IP packets.

Rule action	Can be set to either *permit* to process the IP packet further or *deny* to prevent further handling of it as a result of this rule.
Source address definition	Two dotted decimal addresses. First the address and then the network mask.
Destination address definition	Same syntax as for source address definition.
Protocol	The protocol type of the IP packet. This can be *all*, *icmp*, *tcp*, *tcp/ack* or *udp*. The *tcp/ack* matches only TCP packets that have the acknowledgement bit on.
Source port/ICMP type	The first field specifies the type of operation, the second the desired port number (for ICMP packets, it is the ICMP type of message). The operation is an arithmetic operator field, such as: *all*, *eq*, *neq*, *gt*, *ge*, *lt*, and *le*. The operator is applied to the port field; so for example, gt 1035 would only match packets from port 1035 or higher.
Destination port/ICMP code	Same usage as source port above. For ICMP packets, it refers to the ICMP code field.
Adapter	Defines which adapter the packet is flowing through: *secure*, *non-secure* or *both* (does not care which one).
Routing	Defines whether the packet is from or destined for the firewall itself or whether it is from and destined to other machines, in which case the firewall is to act as router. This can be set to *local*, *route* or *both*.
Direction	Defines whether the packet is coming into or going out of the specified adapter. Can be *inbound*, *outbound* or *both* (does not care which way).
Log control	This determines if the packet should be logged by the firewall.

Fragmentation control	This determines whether the rule should match headers, fragments and non-fragmented packets. `yes` matches all, `no` matches only non-fragmented packets and `only` matches only headers and fragments.

The topology of the network for this example is shown in Figure 68 where the DCE/DFS client machine (which is running DFS Web Secure) is connected to the network defined as non-secure to the firewall. Here is an example of some generic filter rules for this network that are to prevent IP address spoofing across this firewall:

```
deny 9.3.1.0 255.255.255.0 0 0 all any 0 any 0 non-secure both inbound
deny 0 0 9.3.1.0 255.255.255.0 all any 0 any 0 non-secure both outbound
```

The first rule stops inbound traffic on the non-secure interface of the firewall that purports to originate from a secure address on the non-secure network (in other words something pretending to be on the secure side). The second rule stops outbound traffic on the non-secure interface that purports to be destined for a secure address on the non-secure network. These filter rules are recommended, and they should be placed near the top of the filter rules file if it is processed sequentially.

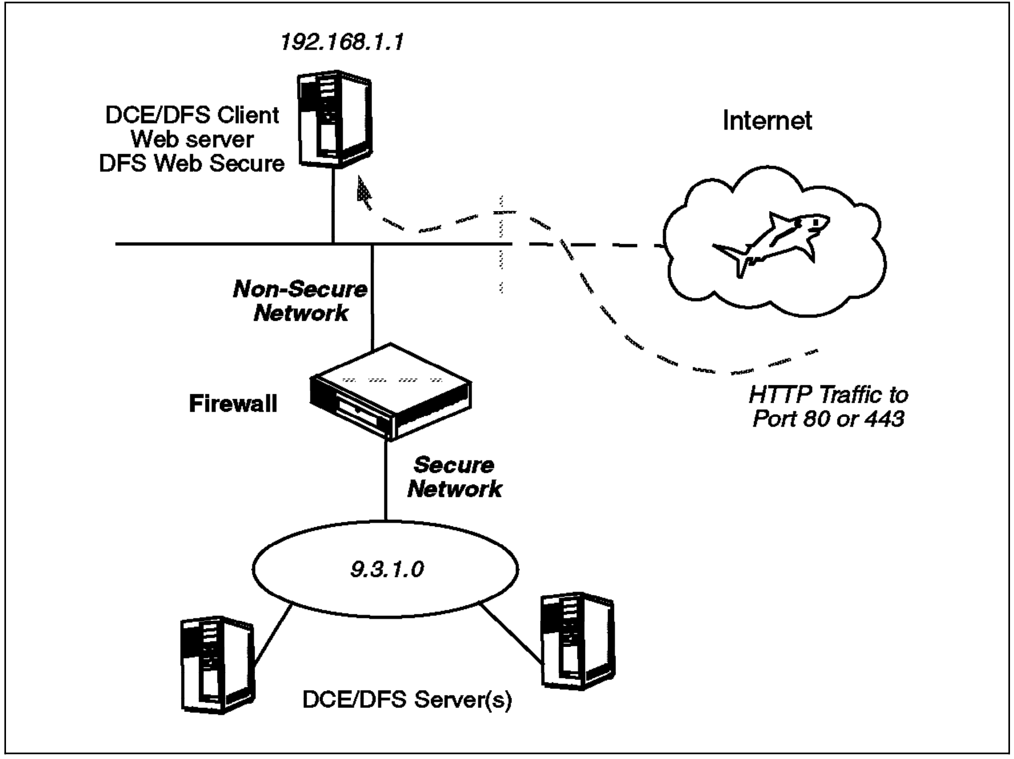


Figure 68. Network Layout for DFS Web Secure over Firewall

The filtering rules to open up the firewall to DCE traffic need to encompass the following lines of TCP/IP communication:

- Both TCP and UDP traffic into port 135 on the DCE client system from the restricted DCE port ranges (set for UDP and TCP in the variable `RPC_RESTRICTED_PORTS`) on all DCE servers on the secure network that the DCE client system needs to access.

- Both TCP and UDP traffic, in both directions, between port 135 on the DCE server systems on the secure network and the restricted DCE port ranges (set in the variable RPC_RESTRICTED_PORTS) on the DCE client system.
- Both TCP and UDP traffic, in both directions, between the respective DCE port ranges on the DCE client system and all DCE servers on the secure network that the DCE client system needs to access.

These filter rules are depicted diagrammatically in Figure 69. It is possible to limit DCE to use only UDP traffic, which will reduce by half the number of filter rules that need to be set up (in effect reducing the opening up of the firewall). DCE/DFS traffic could not be restricted to TCP alone because DFS manages its own communications over UDP. The filter rules for this example do restrict the DCE traffic to only one IP address on the non-secure network—the known IP address 192.168.1.1 of the DCE/DFS client system. The rules could be tightened up further by preparing a separate rule set for each DCE server system and each DFS server system that the client system needs to communicate with. Security would then be tighter, particularly since there are other software services that make use of port 135, and the rules shown here allow the DCE client system to contact any of them on the secure network. Unless specifically changed by various means, all DCE Security and CDS servers should be made accessible from/to the client system with specific filter rules. Also with DFS servers, creating specific per-server filter rules will mean having to set one up for every DFS server that is serving a fileset that the DCE/DFS client on the non-secure network wants to access. As is often the case, tightening security in this way will involve increased administration effort.

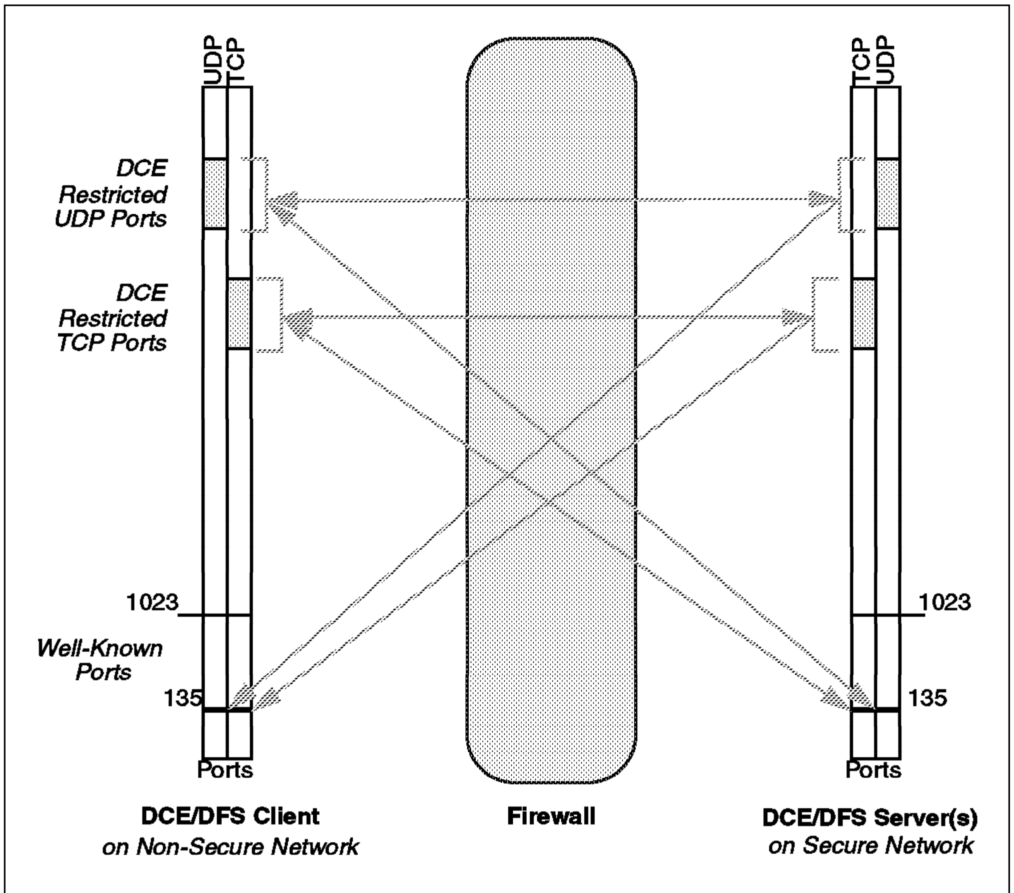


Figure 69. Firewall Filter Rules with DCE/DFS Client on Non-Secure Network

The actual filter rules used for this example are shown in Figure 70 on page 151:

```
# rule block for traffic from DCE servers on secure network into port 135 on the DCE/DFS client system
#
deny   9.3.1.0 255.255.255.0 192.168.1.1 255.255.255.255 udp gt 6550 eq 135 both route both l=y f=n
permit 9.3.1.0 255.255.255.0 192.168.1.1 255.255.255.255 udp gt 6499 eq 135 both route both l=y f=y
deny   9.3.1.0 255.255.255.0 192.168.1.1 255.255.255.255 tcp gt 5050 eq 135 both route both l=y f=y
permit 9.3.1.0 255.255.255.0 192.168.1.1 255.255.255.255 tcp gt 4999 eq 135 both route both l=y f=y
#
# similar rule block for traffic in both directions between DCE restricted ports on DCE client and
# port 135 on the DCE servers on the secure network
#
deny   192.168.1.1 255.255.255.255 9.3.1.0 255.255.255.0 tcp gt 5050 eq 135 both route both l=y f=y
permit 192.168.1.1 255.255.255.255 9.3.1.0 255.255.255.0 tcp gt 4999 eq 135 both route both l=y f=y
deny   192.168.1.1 255.255.255.255 9.3.1.0 255.255.255.0 udp gt 6550 eq 135 both route both l=y f=n
permit 192.168.1.1 255.255.255.255 9.3.1.0 255.255.255.0 udp gt 6499 eq 135 both route both l=y f=y
deny   9.3.1.0 255.255.255.0 192.168.1.1 255.255.255.255 udp eq 135 gt 6550 both route both l=y f=n
permit 9.3.1.0 255.255.255.0 192.168.1.1 255.255.255.255 udp eq 135 gt 6499 both route both l=y f=y
deny   9.3.1.0 255.255.255.0 192.168.1.1 255.255.255.255 tcp eq 135 gt 5050 both route both l=y f=y
permit 9.3.1.0 255.255.255.0 192.168.1.1 255.255.255.255 tcp eq 135 gt 4999 both route both l=y f=y
#
# rules for 2-way traffic between restricted ports on DCE servers and DCE client
#
deny   192.168.1.1 255.255.255.255 9.3.1.0 255.255.255.0 tcp gt 5050 gt 5050 both route both l=y f=y
permit 192.168.1.1 255.255.255.255 9.3.1.0 255.255.255.0 tcp ge 5000 ge 5000 both route both l=y f=y
deny   9.3.1.0 255.255.255.0 192.168.1.1 255.255.255.255 tcp gt 5050 gt 5050 both route both l=y f=y
permit 9.3.1.0 255.255.255.0 192.168.1.1 255.255.255.255 tcp ge 5000 ge 5000 both route both l=y f=y
deny   192.168.1.1 255.255.255.255 9.3.1.0 255.255.255.0 udp gt 6550 gt 6550 both route both l=y f=n
permit 192.168.1.1 255.255.255.255 9.3.1.0 255.255.255.0 udp ge 6500 ge 6500 both route both l=y f=y
deny   9.3.1.0 255.255.255.0 192.168.1.1 255.255.255.255 udp gt 6550 gt 6550 both route both l=y f=n
permit 9.3.1.0 255.255.255.0 192.168.1.1 255.255.255.255 udp ge 6500 ge 6500 both route both l=y f=y
```

Figure 70. Firewall Filter Rules for DCE

Notice how the filter rules to restrict traffic to a range of ports have to come in deny/permit pairs. For each IP packet arriving at the firewall, the information for that packet is compared with each filter rule in sequence, working down the filter rules file. Processing stops when the first match is found; so the deny rule for ports higher than the upper limit of the port range must precede the permit rule that defines the lower limit of the port range.

In this example we are routing the DCE traffic through the firewall, using packet filters to control them. Ideally, we would like to run the firewall as a dual-homed gateway, with all sessions being broken and then reestablished on the other side. Unfortunately, the complex nature of the DCE sessions makes this impractical. The next best thing is to use filters that define the session endpoints very precisely to prevent them being abused by an attacker. That is what we have done in this example.

In certain cases a real environment may use illegal or unroutable addresses on the secure side of the firewall. What this means is that they are IP address ranges that have not been officially assigned, but have simply been chosen by the administrators, either at random or from the special private address ranges defined in RFC1597. The rule is that these addresses should never be routed into the Internet, but in this case we are, in fact, allowing them through the firewall boundary. By having filters that restrict this routing, we are preventing the illegal addresses from being visible anywhere except on the external server, which is acceptable. Note, however, that the external server should only use static routing in this case. If you run a routing protocol on the server, such as RIP (routed daemon), it will advertise the illegal addresses to the Internet backbone.

When using DFS Web Secure on the DCE/DFS client system across the firewall from the DCE servers, we found that the CGI processes that were being started by the Web server were not picking up the RPC_RESTRICTED_PORTS environment variable. Therefore any DCE access was likely to start on a port that was restricted by the firewall filter rules above and fail. This prevents use of the DFS Web Manager and DCE Administration GUI components of DFS Web Secure. This situation can be avoided by configuring the Netscape server to setup the RPC_RESTRICTED_PORTS environment variable in the initialization of each CGI process. This is done by adding this line to the Init section of the Netscape server configuration file, obj.conf, (usually located in the directory /usr/ns-home/httpd-<server_instance_name>/config):

```
Init fn="init-cgi"
    RPC_RESTRICTED_PORTS="ncadg_ip_udp[6500-6550]:ncacn_ip_tcp[5000-5050]"
```

If this configuration change is not carried out but you would like to open up more channels in the firewall to allow this communication to pass through, then the filter rules will have to be changed to match the diagram in Figure 71. This represents a rather open set of channels through the firewall.

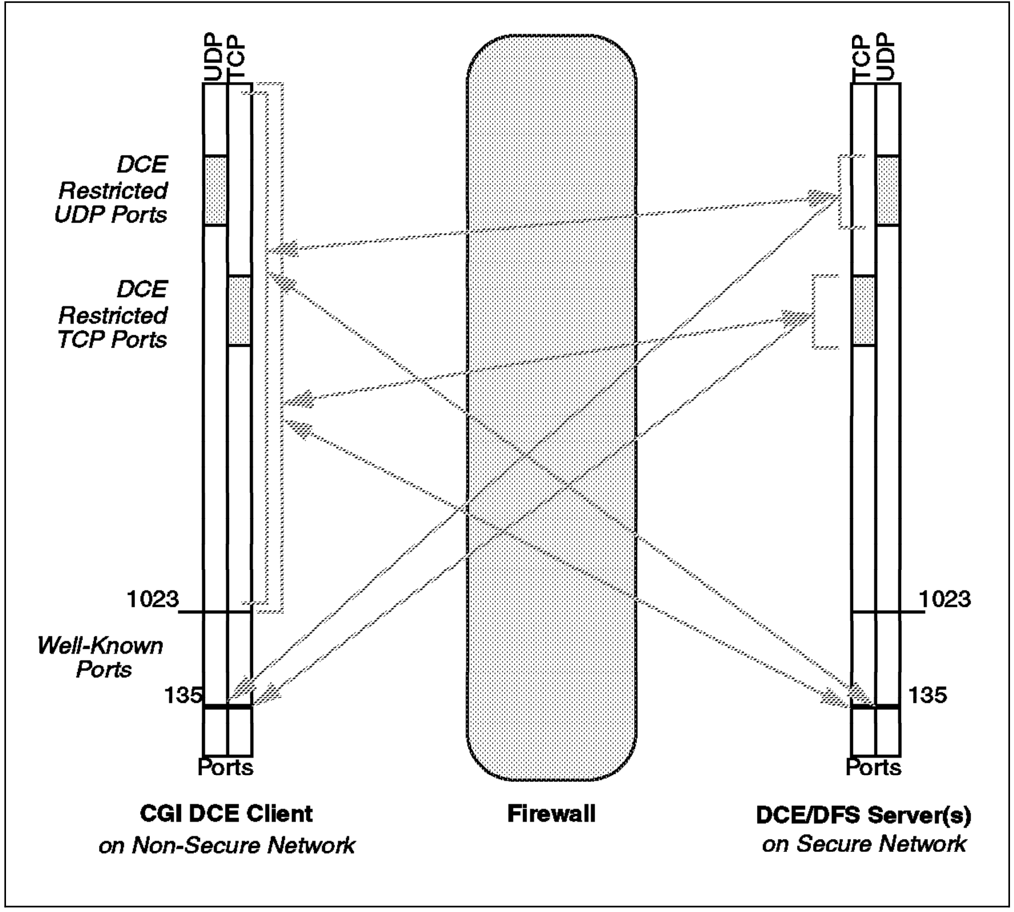


Figure 71. Filter Rules for CGI Programs without RPC_RESTRICTED_PORTS

8.2.2 DE-Light Gateway in a Screened Subnet (DMZ)

In this example we look at the filter rules that are required to allow Java applets in standard Web browsers on the Internet to access DCE applications running on a secure internal network. This capability is provided by using the DE-Light Web product from Transarc Corp., which is covered in detail in Chapter 6, “DE-Light Web Client” on page 109. Here we consider the port traffic involved with DE-Light Web that will be the basis of the filter rules that are required to allow this communication.

There are several steps to the communication involved in this example that are shown in Figure 72 on page 154. The DE-Light gateway is running on a system that is inside a screened subnet as part of a firewall strategy to prevent direct traffic between the secure internal network and the Internet. The communication sequence is:

1. The Web browser contacts the Web server to get a Java applet that can access the DCE application. This requires HTTP traffic across the firewall, usually on ports 80 or 443 (for SSL).
2. The Web server sends the Java applet class files back to the Web browser. This requires return HTTP traffic to be permitted through the firewall from the Web server port (usually 80 or 443).
3. The Java applet (a DE-Light client) runs in the browser and connects to the DE-Light gateway that is running on the same system as the Web server inside the DMZ. The DE-Light gateway can be started on specified port numbers for TCP as well as on specified port numbers for HTTP and HTTPS. The chosen port number(s) is specified among the command line arguments when the DE-Light gateway is started. The filter rules need to allow traffic to come into these chosen ports on the DE-Light gateway system and allow traffic to return from these ports to the DE-Light client (which could be on any dynamic port on the browser’s system).
4. The DE-Light gateway makes the DCE application call on behalf of the DE-Light client, which requires DCE runtime services to be flowing across the firewall between the DMZ and the secure network where the DCE application servers are running. The filter rules to allow DCE traffic in both directions across a firewall are covered in the previous example.

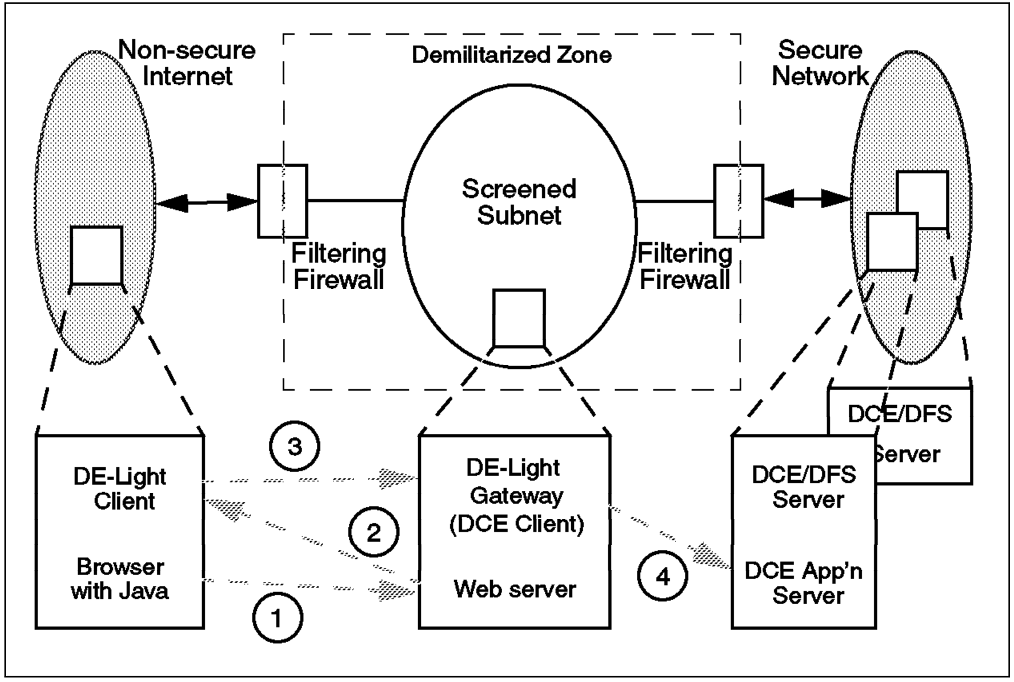


Figure 72. DE-Light Gateway in a Demilitarized Zone

Some sample filter rules for this example environment covering the first three stages of the communication sequence listed above are shown in Figure 73. Sample filter rules for DCE traffic across a firewall (step 4) are shown in the previous example. The sample rules shown in Figure 73 would be set on the external firewall or screening router (between the Internet and the DMZ). The filter rules for step 4 would be set on the firewall between the DMZ and the secure internal network. The firewalls for the DMZ would also be set up to prevent traffic being routed across the DMZ between the Internet and the secure internal network.

```
#
# Rule block to allow Web browser HTTP traffic from anywhere to port 443 (as step 1. above)
# and the return traffic with the TCP acknowledgement bit set on (as step 2. above)
#
permit 0 0 192.168.1.1 255.255.255.255 tcp     gt 1023 eq 443 non-secure route inbound  l=y f=y
permit 0 0 192.168.1.1 255.255.255.255 tcp     gt 1023 eq 443 secure     route outbound l=y f=y
permit 192.168.1.1 255.255.255.255 0 0 tcp/ack eq 443 gt 1023 secure     route inbound  l=y f=y
permit 192.168.1.1 255.255.255.255 0 0 tcp/ack eq 443 gt 1023 non-secure route outbound l=y f=y
#
# Rules for DE-Light Gateway on port 4913 in DMZ , open for access from a client applet anywhere
# (as step 3. above)
#
permit 0 0 192.168.1.1 255.255.255.255 tcp     gt 1023 eq 4913 non-secure route inbound  l=y f=y
permit 0 0 192.168.1.1 255.255.255.255 tcp     gt 1023 eq 4913 secure     route outbound l=y f=y
permit 192.168.1.1 255.255.255.255 0 0 tcp/ack eq 4913 gt 1023 secure     route inbound  l=y f=y
permit 192.168.1.1 255.255.255.255 0 0 tcp/ack eq 4913 gt 1023 non-secure route outbound l=y f=y
```

Figure 73. Sample Filter Rules for DE-Light Example

Appendix A. Other Sources on the Internet

This Appendix lists some sources of additional information on the Internet pertinent to the subject of this book. The provided links and news groups are by far not all available sources of information on the Internet; they represent a small selection which was found to be useful while writing this book. Due to the dynamic nature of the Internet, some of the links provided in this Appendix may already have changed or disapeared at the time you try to access them.

A.1 Links on the World Wide Web

This sections lists some of the unnumbered links on the World Wide Web that provide information about DCE, DFS, Web Security, Firewalls, Products, and other related topics.

A.1.1 DCE and DFS

The IBM DCE home page:

```
http://www.networking.ibm.com/dce/dcehome.html
```

Transarc Corporation's home page and DCE/DFS product information:

```
http://www.transarc.com/
http://www.transarc.com/afs/transarc.com/public/www/Public/ProdServ/Product/
index.html
```

The Open Group's DCE home page:

```
http://www.opengroup.org/tech/dce/
```

OSF DCE Frequently Asked Questions (FAQs):

```
http://www.cis.ohio-state.edu/hypertext/faq/usenet/dce/faq/faq.html
http://www.dstc.edu.au/AU/research_news/dce/dce-faq.html
http://www.mmt.bme.hu/~kiss/docs/dce/contents.html
```

DFS high level overview:

```
http://www.transarc.com/afs/transarc.com/public/www/Public/ProdServ/Product/
DFS/dfsoverview.html
```

Distributed File Systems, a comarison report by the Patricia Seybold Group:

```
http://www.transarc.com/afs/transarc.com/public/www/Public/ProdServ/Product/
DFS/Seybold/dfssey.html
```

DFS performance report by Cray Research, Inc.:

```
http://www.cray.com/PUBLIC/product-info/sw/dce/perf.html
```

IBM employees who have access to the internal Web may also refer to the internal DCE and DFS home page. It contains a number of links to other, internal and external Web links. It is located at:

```
http://w3.austin.ibm.com/dce/
```

A.1.2 Security and the Internet

The World Wide Web Security FAQ:

```
http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html
```

RSA FAQ:

```
http://www.rsa.com/rsalabs/faq/faq_des.html
```

Various:

```
http://home.netscape.com/info/security-doc.html
http://world.std.com/~franl/crypto/rsa-guts.html
http://www.ascom.ch/Web/systec/security/idea.htm
http://www-ns.rutgers.edu/www-security/
http://ganges.cs.tcd.ie/mepeirce/Project/oninternet.html
```

Cybercash:

```
http://www.cybercash.com/
http://www.ecstore.com/cybercash.html
```

A.1.3 Secure Internet Protocols (SSL, S-HTTP, SET)

SSL:

```
http://www.ietf.org/ids.by.wg/tls.html
http://www.consensus.com/security/ssl-talk-faq.html
http://www.interwebinc.com/security/
http://home.netscape.com/newsref/std/SSL.html
http://www.netscape.com/newsref/std/tunneling_ssl.html
```

S-HTTP:

```
http://ds.internic.net/internet-drafts/draft-ietf-wts-shttp-04.txt
http://ds.internic.net/internet-drafts/draft-ietf-wts-shtml-03.txt
http://www.homeport.org/~adam/shttp.html
```

SET:

```
http://www.tamney.com/set.htm
http://www.verifone.com/products/software/icommerce/html/setpaper.html
http://www.mastercard.com/set/
http://www.visa.com/cgi-bin/vee/sf/set/setbus.html
http://www.virtualschool.edu/mon/ElectronicProperty/klamond/Setpmt.htm
```

Netscape and SSL certificates:

```
http://home.netscape.com/newsref/std/ssl_2.0_certificate.html
```

Netscape and RSA Public Key Cryptography:

```
http://home.netscape.com/newsref/ref/rsa.html
```

VeriSign Digital ID Center:

```
http://digitalid.verisign.com/crp_intr.htm
```

A.1.4 Java

Java Developers Kit (JDK):

http://www.javasoft.com/products/jdk/

Netscape JavaScript Guide:

http://home.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html

A.1.5 Web, General

HTML references:

http://www.w3.org/pub/WWW/
http://www2.wvitcoe.wvnet.edu/~sbolt/html3/

The Common Gateway Interface (CGI):

http://hoohoo.ncsa.uiuc.edu/cgi/

CGI programming libraries, mostly for C language:

http://www.camtech.com.au/jemtek/cgi/lib/
http://www.boutell.com/cgic/
http://www.ncsa.uiuc.edu/People/daman/cgi++/
http://www.he.net/~searsbe/ScriptingInC.html
http://www.go2net.com/people/paulp/cgi-security/safe-cgi.txt
http://www.perl.com/perl/faq/perl-cgi-faq.html

A.1.6 Firewalls

Firewall FAQs and related information:

http://sun1000e.pku.edu.cn/on_line/firewall/faq.html
http://www.inf.utfsm.cl/~vparada/faq/firewall
http://www.cis.ohio-state.edu/hypertext/faq/usenet/firewalls-faq/faq.html

A.1.7 Product Related

Various Netscape documentation:

http://home.netscape.com/search/index.html
http://developer.netscape.com/library/documentation/index.html
http://developer.netscape.com/library/one/sdk/enterprise/doc/unixguid/
appconf.htm
http://home.netscape.com/comprod/server_central/config/nsapi.html

Netscape Plug-Ins:

http://home.netscape.com/eng/mozilla/3.0/handbook/plugins/pguide.htm

DE-Light overview and documentation:

http://www.transarc.com/afs/transarc.com/public/www/Public/ProdServ/Product/
DELight/index.html
http://www.transarc.com/afs/transarc.com/public/www/Public/ProdServ/Product/
DELight/delov.html

A.2 News Groups

The following news groups serve as an arena for discussions, questions, answers and your own contributions about DCE in general:

```
comp.soft-sys.dce
comp.security.firewalls
comp.lang.java.*
comp.infosystems.www.authoring.cgi
sci.crypt
```

Appendix B. Program Listings

This Appendix provides the source code listings of the various programs used in the previous chapters of this book.

All sample code provided in this Appendix can be downloaded from the IBM Redbook Web page at http://www.redbooks.ibm.com. From this page, click on the **DOWNLOADS** button and then click on the SG244949 directory in order to access the files. Alternatively, you can download the files using anonymous FTP from www.redbooks.ibm.com. Once connected, the files can be found in the directory redbooks/SG244949.

In the directory mentioned above you can find a single tar file which contains all sample code files of this Appendix in separate subdirectories.

B.1 Recursive DFS ACL Viewing and Editing

The following korn shell script, named rchacl, provides a recursive ACL editing and viewing utility. Please read the instruction and information in the header. The utility is provided as is.

```
#!/usr/bin/ksh
################################################################################
#
# rchacl: A recursive ACL listing and editing utility. Allows to list, modify and
#         delete ACL entries for all files and/or directories beneath a given
#         directory. As with any recursive tools, use only with care!
#
# Modify or delete ACL entries:
#   rchacl <DFS-Dir> {-m|-d} <acl_entry> [-io|-ic|-f|-d]
#
# List ACL entries:
#   rchacl <DFS-Dir> -l [-io|-ic|-f|-d]
#
# Flags: -m  modify specified ACL entries
#        -d  delete specified ACL entries
#        -l  list ACLs
#        -io 'initial object' (directories only)
#        -ic 'initial container' (directories only)
#        -f  files only, do not list/modify/delete directory ACLs
#        -d  directories only, do not list/modify/delete file ACLs
#        acl_entry is an ACL entry, e.g. user:foo:rwx---
#
# Note: The recursive processing does NOT follow symbolic links. If, for
#       example, /: is specified as directory, rchacl will not change/list
#       any ACLs of files or directories beneath it (since /: is a symbolic
#       link to /.../<cell name>/fs).
#
# This tools is provided AS IS, use at your own risk.
#
################################################################################

# Usage message
usage(){
  echo "\nUsage: rchacl <DFS-Dir> {-m|-d} <acl_entry> [-io|-ic|-f|-d]"
  echo "    or rchacl <DFS-Dir> -l [-io|-ic|-f|-d]"
  echo "    or rchacl -?"
  echo "\nExamples: rchacl /:/home/charlie/private -m user:charlie:rwxcid"
  echo "          rchacl /:/home -m user:cell_admin:rwxcid -ic"
  echo "          rchacl /:/home -d group_obj:rwx---\n"
}

# Check number of arguments
if [ $# -lt 1 -o $# -gt 4 ]; then
  echo "\nInvalid number of arguments." >&2
  usage >&2
  exit 1
fi

# Handle usage requested with -?
if [ $1 = -? ]; then
  usage
  exit 0
fi

DFSDIR=$1
OPER=$2
if [ "$OPER" = "-l" ]; then
  WORK_ON="$3"       # May be empty
else
  ACL_ENTRY=$3
  WORK_ON="$4"       # May be empty
fi

# Check if $DFSDIR is a directory
if [ ! -d "$DFSDIR" ]; then
  echo "$DFSDIR is not a directory." >&2
  exit 1
fi

# Process the file and/or directory options
case "$WORK_ON" in
  -ic | -io ) IC_IO=$WORK_ON
              TYPE="-type d"
              ;;
  -f )        IC_IO=
              TYPE="-type f"
              ;;
  -d )        IC_IO=
              TYPE="-type d"
              ;;
  "" )        IC_IO=
              TYPE=
              ;;
  * )         echo "'$WORK_ON' is not a valid parameter." >&2
              exit 1
              ;;
esac

# Recursive ACL processing
case $OPER in
  -l ) # Recursive ACL listing, check parameters first
       find $DFSDIR $TYPE -exec acl_edit {} -l $IC_IO \;
       ;;
  -m ) # Recursive ACL modification
       ACL_ENTRY=$3
       find $DFSDIR $TYPE -exec acl_edit {} -m $ACL_ENTRY $IC_IO \;
       ;;

  -d ) # Recursive ACL deletion
       ACL_ENTRY=$3
       find $DFSDIR $TYPE -exec acl_edit {} -d $ACL_ENTRY $IC_IO \;
       ;;

  * )  usage >&2
       exit 1
       ;;
esac

# Terminate normally
exit 0
```

B.2 Java Classes for Calc

Section 6.3, “An Example Using DE-Light Web” on page 123, uses a simple Java application to access a DCE calculation server. The client software consists of three source files:

- A Makefile responsible for building the Java classes
- The base calculator class, `Calc.java`
- An extension to the applet class providing a graphical user interface to the calculator service, `CalcUI.java`

The code samples provided here are derived from sample code for the DE-Light product from Transarc Corp.

B.2.1 Makefile

```
# ID: $Id: Makefile.aix,v 1.1 1996/11/22 19:24:45 rek Exp $
#
# COMPONENT_NAME: DCE Encina Lightweight Client Example Programs
#
# Modified for use with a calculator example
#
# ORIGINS: Transarc Corp.
#
# (C) COPYRIGHT Transarc Corp. 1996
# All Rights Reserved
# Licensed Materials - Property of Transarc
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with Transarc Corp
#

JAVAC=javac
DELIGHTCLASSES=/usr/lpp/DE-Light/lib/javaclasses
CLASSPATH=/usr/lpp/Java/classes
BASECLASSPATH=.:${DELIGHTCLASSES}:${CLASSPATH}
JAVACSWITCHES=-classpath ${BASECLASSPATH}

CLASSES=Calc.class CalcUI.class
SOURCES=Calc.java CalcUI.java

all: ${CLASSES}

Calc.class: Calc.java
	${JAVAC} ${JAVACSWITCHES} Calc.java

CalcUI.class: CalcUI.java
	${JAVAC} ${JAVACSWITCHES} CalcUI.java

clean:
	rm -f ${CLASSES}
```

B.2.2 Calc Class (Calc.java)

Provides constructor functions for the `Calc` class as well as for the `add` routine doing the real work. `identity` is a separate method since not all class instances will require DCE privileges be obtained.

```
/*
 * ID: $Id: Calc.java,v 1.2 1996/12/23 15:25:00 rek Exp $
 *
 * COMPONENT_NAME: DCE Encina Lightweight Client Example Programs
 *
 * ORIGINS: Transarc Corp.
 *
 * (C) COPYRIGHT Transarc Corp. 1996
 * All Rights Reserved
 * Licensed Materials - Property of Transarc
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with Transarc Corp
 */

/**
 * @author Copyright (c) Transarc Corporation 1996
 * @version 1.1
 *
 * This is a modified version of the Java DE-Light example included
 * with the DE-Light Web distribution.  Calc is a DCE application
 * server which provides a single RPC call.
 *
 * long int add(
 *      [ in ]  handle_t       bh,
 *      [ in ]  long int       a,
 *      [ in ]  long int       b,
 *      [ out ] long           *result);
 *
 * This example may either be run standalone or as part of a calling/derived
 * class.
 */
import java.applet.Applet;
import java.awt.*;
import COM.Transarc.Delight.*;

public class Calc {

  /* DE-Light connection information */
  private DrpcConnection drpc;
  private String dceServerName;

  /**
   * main is called only when this applet is run standalone
   * (i.e. without a browser).
   * @param args     specifies the DE-Light gateway name, DCE server name
   *                 to which the client is to bind, user name,
   *                 password, and the two numbers to add.
   */

  public static void main(String args[]) {

    int reply = 0;

    if( args.length < 4 ) {
      System.out.println("Usage: java Calc" +
                         " gateway-name  dce-server-name" +
                         " [user password] a b");
      System.exit(1);
    }

    try {

      // Construct and bind, specifying the gateway
      // location, server's DCE name, and required security.  We
      // provide a second contructor which doesn't need the security
      // options.
      Calc CalcInterface = new Calc(args[0], args[1],
                              DrpcConnection.DRPC_DCE_PROTECT_MAX_VALUE,
                              DrpcConnection.SEC_BEST);

      int a = 0;
      int b = 0;

      // Ideally we should be checking that these are integers before
      // parsing
      if (args.length == 4) {
        a = Integer.parseInt(args[2]);
        b = Integer.parseInt(args[3]);
      } else if (args.length == 6) {
        a = Integer.parseInt(args[4]);
        b = Integer.parseInt(args[5]);

        // Set up the login context for the new connection
        System.out.println("Setting login context " + args[2]);
      CalcInterface.identity(args[2],args[3]);
    } else {
      System.out.println("Usage: java Calc" +
                         " gateway-name  dce-server-name" +
                         " [user password] a b");
      System.exit(1);
    }

    reply = CalcInterface.add( a, b);

    System.out.println("The Calc Server said: " + String.valueOf(reply) );

  } catch (Exception e) {
    System.out.println( "Error in call to Calc: " + e.toString() );
    e.printStackTrace();
  }
}

public void identity(String name, String passwd) throws Exception {
  // This could fail if the DCE login or password are not correct.
  try {
    drpc.dceLogin(name,passwd);
  } catch (DrpcException e) {
    System.out.println( "drpc.dceLogin failed: " + e.toString() );
    throw(new Exception("Failed to dceLogin: " +
                        e.toString() ));
  }
}


/**
 * This form of the constructor is used if security options should
 * be included in the connection.
 **/
public Calc( String gatewayName, String serverName,
             int DCESec, int Sec){
  try {
    drpc = new DrpcConnection( gatewayName );
    drpc.setDceSecurityLevel(DCESec);
    drpc.setSecurity(Sec);
  } catch (DrpcException e) {
    drpc = null;
    System.out.println("Failed to obtain DrpcConnection: " +
                       e.toString() );
  }
  dceServerName = serverName;
}


/**
 * This form of the constructor is used if no security options are
 * to be set up ober the connection.
 **/
public Calc( String gatewayName,
             String serverName){
  try {
    drpc = new DrpcConnection( gatewayName );
  } catch (DrpcException e) {
    drpc = null;
    System.out.println("Failed to obtain DrpcConnection: " +
                       e.toString() );
  }
  dceServerName = serverName;
}

/**
 * This method wraps the DRPC calls in an interface with a Java
 * equivalent to that specified in the DCE IDL file.
 *
 **/
public int add(int a, int b) throws Exception {
  // store [in] parameters in the default data dictionary
  drpc.dictionary().putInt("a", a );
  drpc.dictionary().putInt("b", b );
```

```
    // Issue the call to the gateway/server
    drpc.callRpc(" server=" + dceServerName +
                 " interface=ServerID rpc=add" +
                 " [return] val [in] a [in] b [out] result");

    // retrieve/return the [out] parameter and return value
    int response = drpc.dictionary().getInt("result");
    int val = drpc.dictionary().getInt("val");

    if ( val < 0 ) {
      throw (new Exception("Bad add result"));
    }
    return response;
  }
};
```

B.2.3 Calc User Interface Class (CalcUI.java)

This is the user interface wrapper for the `Calc` class. It extends the applet class, as do all applets, and provides the expected interface construction and event-handling routines.

```
/*
 * ID: $Id: CalcUI.java,v 1.2 1996/12/23 15:25:00 rek Exp $
 *
 * COMPONENT_NAME: DCE Encina Lightweight Client Example Programs
 *
 * ORIGINS: Transarc Corp.
 *
 * (C) COPYRIGHT Transarc Corp. 1996
 * All Rights Reserved
 * Licensed Materials - Property of Transarc
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with Transarc Corp
 */
import java.applet.Applet;
import java.awt.*;
import COM.Transarc.Delight.*;

/**
 * @author Copyright (c) Transarc Corporation 1996
 * @version 1.1
 *
 * This is the Java / DE-Light user interface driver for the Calc example.
 * It is based upon the DE-Light Web example provided with the distribution.
 *
 * See Calc.java for further information.
 */

public class CalcUI extends Applet {

  /* DE-Light connection information */
  private Calc CalcInterface;

  private String drpcGatewayName;
  private String dceServerName;

  /* Gui objects */

  private TextField resultField;
  private TextField aField;
  private TextField bField;
  private TextField nameField;
  private TextField passwdField;
  private TextField msgField;
  private Button sendCalcingBtn;

  /**
   * The constructor method
   */
  public CalcUI() {
    super();
  }
```

```
/**
 * main is called only when this applet is run standalone
 * (i.e. without a browser).  It instantiates a frame with a
 * Calc applet form in it.
 */
public static void main(String args[]) {
  if( args.length < 2) {
    System.out.println("Usage: java CalcUI drpcGatewaySpec dceServerName");
    System.exit(1);
  }

  // Create a new window for our applet to run within
  Frame f = new Frame("Calc");

  // Create a new applet and initialize its data per command line arguments
  CalcUI CalcerForm = new CalcUI();
  CalcerForm.drpcGatewayName = args[0];
  CalcerForm.dceServerName = args[1];

  // Initialize the applet and display it in the window
  CalcerForm.init();
  f.add("Center", CalcerForm);
  f.pack();
  f.resize( f.preferredSize() );
  f.show();
}

/**
 *  Initialize the applet GUI interface and DRPC connection
 */
public synchronized void init()
{
  DrpcConnection drpc;

  // Set up the Panel with a simple FlowLayout containing
  // a single button and field pair
  setLayout(new GridLayout(4,1));

  Panel loginPanel = new Panel();

  loginPanel.add(new Label("Login:"));
  nameField = new TextField(10);
  loginPanel.add(nameField);
  loginPanel.add(new Label("Password:"));
  passwdField = new TextField(10);
  passwdField.setEchoCharacter('#');
  loginPanel.add(passwdField);

  add(loginPanel);

  Panel addPanel = new Panel();

  aField = new TextField(3);
  addPanel.add(aField);
  addPanel.add(new Label("+"));
  bField = new TextField(3);
  addPanel.add(bField);
  addPanel.add(new Label("="));
  resultField = new TextField(4);
  resultField.setEditable(false);
  addPanel.add(resultField);

  add(addPanel);

  msgField = new TextField(30);
  msgField.setEditable(false);
  add(msgField);

  sendCalcingBtn = new Button("Do it");
  add(sendCalcingBtn);

  // Obtain the gateway name and open a connection
  if( drpcGatewayName == null ) {
    drpcGatewayName = getParameter("Calc.gateway");
    }

    if (drpcGatewayName == null ) {
      throw(new RuntimeException("Gateway not specified: " +
                                 "drpc not initialized"));
    }

    if( dceServerName == null ) {
      // Get the DCE server's name from an applet tag
      dceServerName = getParameter("Calc.dceServer");
    }

    if( dceServerName != null ) {
      // Create a Calc DRPC interface object bound to the gateway
      // and DCE CDS name specified parameters.

      CalcInterface = new Calc(drpcGatewayName, dceServerName,
                               DrpcConnection.DRPC_DCE_PROTECT_MAX_VALUE,
                               DrpcConnection.SEC_BEST);

    } else {
      throw(new RuntimeException("DCE server name not specified: " +
                                 "Calc interface not initialized"));
    }
  }

  /**
   * action() - process user input events
   */
  public synchronized boolean action(Event evt, Object obj) {

    // If it is our button press event, then handle it, otherwise
    // pass it on for others.
    if ( evt.target == sendCalcingBtn ) {

      // Create a container for the response
      int reply = 0;

      try {
        // Notify the user that we are sending the DRPC
        int a = Integer.parseInt(aField.getText());
        int b = Integer.parseInt(bField.getText());

        resultField.setText(" ");
        msgField.setText("                    ");

        String name = nameField.getText();
        String passwd = passwdField.getText();

        // Set the identity
        if ( name.length() != 0 ) {
          try {
            CalcInterface.identity(name,passwd);
          } catch ( Exception e ) {
            msgField.setText("Login Failure");
            System.out.println( "CalcInterface.identity failed:" + e.toString() );
            e.printStackTrace();
            throw (e);
          }
        }

        // Do the DRPC through the DRPC interface class
        try {
          reply = CalcInterface.add(a, b);
        } catch ( Exception e ) {

          // Display the response
          aField.setText(String.valueOf(a));
          bField.setText(String.valueOf(b));
          resultField.setText("???");

          msgField.setText("Security Failure");

          throw (e);
        }
```

```
          // Display the response
          aField.setText(String.valueOf(a));
          bField.setText(String.valueOf(b));
          resultField.setText(String.valueOf(reply));

          msgField.setText("Success");

        } catch ( Exception e ) {
          System.out.println( e.toString() );
          e.printStackTrace();
        }
        return true;
      }
      return false;               // we didn't handle this event
    }
  };
```

B.3 DCE Math Client and Server

We have used a common DCE application server theme in this book. It is a simple math server taking two integers as input and producing the sum as output. Section 6.3, "An Example Using DE-Light Web" on page 123, uses the application server in a security mode based upon PAC (Privilege Attribute Certificate) information. Clients must provide a valid PAC before being validated by the server. Section 5.5.4, "Native DCE Applications" on page 97, also uses a math DCE example, but security in this example is principal-based, not PAC based. The server uses principal group membership to determine access rights.

The following code samples are derived from code used by IBM Global Services in their DCE High Intensity Classes. For further information on these classes, refer to the *Consulting and Services* pages of the IBM DCE Web page at `http://www.networking.ibm.com/dce/dcehome.html`, or contact Ken Pierce in the U.S. at (303) 924-5012.

We have split the code into three sections. The first is shared between both versions of the calculation server. The second contains PAC-specific code and the third principal-specific code.

B.3.1 Environment Preparation

This section briefly describes the steps required to set up the DCE environment for the servers to execute in. Remember to synchronize the keytab file and CDS entry names with those defined in the Makefiles for each server (see code later in this section). The following activities should be performed as `cell_admin`.

Create the group good_guys.

```
dcecp> group create good_guys -inprojlist y
```

Create a client principal, `sec_client`.

```
dcecp> user create sec_client -group good_guys -org none -force
       -password sec_client -mypwd -dce-
```

Create a server principal: ServerName.

```
dcecp> user create ServerName -group seclab_server -org none
       -force -password ServerName -mypwd -dce-
```

Give the server permission to add an entry to the servers container.

```
dcecp> acl modify /.:/Servers -add {user ServerName i}
dcecp> object create /.:/Servers/SecurityServerID
dcecp> acl modify /.:/Servers/SecurityServerID -entry -add {user ServerName rw}
dcecp> object create /.:/Servers/MathServerID
dcecp> acl modify /.:/Servers/MathServerID -entry -add {user ServerName rw}
```

Allow server to test group membership.

```
dcecp> acl modify /.:/sec/group/good_guys -add {user ServerName rt}
dcecp> exit
```

Set up a keytab file for each of the PAC- and principal-based servers (they may share the same keytab file). Your user ID should have write permission for whichever storage location you specify for the keytab file. The location shown here is just an example.

```
dcecp> keytab create -local /.:/hosts/<hostname>/config/keytab/seclab-keytab
       -attr {{storage /opt/dcelocal/etc/keyfile} \
             {data {ServerName plain 1 ServerName}}}
```

where <hostname> must be replaced by your local host name.

B.3.2 Common Code

math.acf: Part of the DCE interface specification that calls for explicit binding handles.

```
/*
* Filename: math.acf
*/

[explicit_handle] interface ServerID
{

}
```

math.idl: The very simple interface definition used by the two DCE application servers.

```
/*
* Filename: math.idl
*/

[
uuid(00602160-2F96-1B81-846A-10005A8D5B0E),
version(1.0)
]
interface ServerID
{

/* principal name used by server                                     */
const char  *SERVER_NAME        = "ServerName";

/*
*  add operation returns 0 if and only if client is authorized;
*  otherwise, returns -1
*/
long int add(
        [ in ]  handle_t       bh,
        [ in ]  long int       a,
        [ in ]  long int       b,
        [ out ] long           *result);
}
```

math_c.c: A simple math client. Performs the usual server binding before conducting a simple RPC call to the server. Note the check of the PAC_SEC macro which chooses between PAC- and principal-based authentication at the server.

The Makefile for each of the two server versions specifies either PAC_SEC for PAC-based security or STG_SEC for principal-based security.

```
/*
* Filename: math_c.c
*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <dce/rpc.h>
#include "math.h"

#define ERRORCK( proc, st ) \
        { if ( st != error_status_ok) \
                { dce_error_inq_text(st, dce_error_string, &error_inq_st); \
                fprintf(stderr, "%s FAILED, returned %d\n\t%s\n", proc, st,dce_error_string); \
                fflush(stderr);\
        }}

void main(argc, argv)
int argc;
char *argv[];
{
        long int                a=40,
                                b=2,
                                c=0;
        rpc_ns_handle_t         ns_handle;      /* namespace binding handle */
        rpc_binding_handle_t    bh;             /* rpc binding handle */
        error_status_t          st,
                                error_inq_st;
        char                    dce_error_string[256];

        printf("Explicit Bindings starting\n");

        printf("Calling rpc_ns_binding_import_begin()...\n");

        rpc_ns_binding_import_begin(
                rpc_c_ns_syntax_default,        /* syntax */
                (unsigned char *)ENTRY_NAME, /* search start point */
                ServerID_v1_0_c_ifspec,         /* interface handle */
                NULL,                           /* object UUID */
                &ns_handle,                     /* binding handle return */
                &st );                          /* status return */
        ERRORCK( "rpc_binding_import_begin", st);

        printf("Calling rpc_ns_binding_import_next()...\n");

        rpc_ns_binding_import_next(
                ns_handle,                      /* ns binding handle            */
                &bh,                            /* binding handle return        */
                &st );                          /* status return                */
        ERRORCK( "rpc_binding_import_next", st);

        printf("Calling rpc_ns_binding_import_done()...\n");

        rpc_ns_binding_import_done(
                &ns_handle,                     /* ns binding handle            */
                &st );                          /* status return                */
        ERRORCK( "rpc_binding_import_done", st);

        printf("Calling rpc_binding_set_auth_info()...\n");
        rpc_binding_set_auth_info(
                bh,                             /* binding handle */
                (unsigned char *)SERVER_NAME, /* server princ. name */
                rpc_c_protect_level_pkt_integ,  /* send checksum */
                rpc_c_authn_dce_secret,         /* use DES encryption */
                NULL,                           /* use login context */
#ifdef PAC_SEC
                rpc_c_authz_dce,                /* send EPAC, NOT NAME */
#else
                rpc_c_authz_name,               /* send name, not PAC */
#endif
                &st);
        ERRORCK( "rpc_binding_set_auth_info",st);
        printf("calling add() RPC with explicit binding.\n");

        if ( 0 == add(bh,a,b,&c))
        {
                printf("PASSED authorization check..\n");
                printf("\tAfter RPC add\n\t a=%d, b=%d, c=%d\n",a,b,c);
        }
        else
        {
                printf("FAILED authorization check..\n");
        }
}
```

math_m.c: This comprises the routine doing the addition work by the server. It calls an authorization check routine that is supplied in separate files for each of the PAC-based and principal-based security schemes.

```
/*
* Filename: math_m.c
*/

#include <dce/rpc.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "math.h"

/*
 * the following subroutine is the one executed when the client calls the
 * server.  All it does is add the two ints and returns the result.
 *
 * IT returns -1 on error, else it returns 0 iff ok.

 */

extern int is_authorized( handle_t bh );

long int add( bh, a, b, result )
handle_t        bh;
long int        a, b, *result;
{
        if ( is_authorized( bh ) )
        {
                *result = a + b;
                printf("User is authorized...\n");
                printf("\tExiting add( %d, %d, %d )\n", a, b, *result);
                return 0;
        }
        else {
                printf("User is not authorized...\n");
                return -1;
        }
}
```

math_s.c: The server code. It establishes a security context for the server, then goes through the process of registering the service being provided. Note how the server catches an interrupt signal and disentangles itself from the endpoint mapper and CDS. The CDS entry used by the server is defined in the Makefile for each server variety.

```
/*
* Filename: math_s.c
*/

#include "math.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <sys/types.h>
#include <dce/rpc.h>
```

```
#include <dce/dce.h>
#include <sec_login.h>


#define ERRORCK( proc, st ) \
        { if ( st != error_status_ok) \
                { dce_error_inq_text(st, dce_error_string, &error_inq_st); \
                fprintf(stderr, "%s FAILED, returned %d\n\t%s\n",\
proc,st,dce_error_string); \
                fflush(stderr);\
                exit(1);\
                 }}

main()
{
        sigset_t                sigset;
        pthread_t               this_thread = pthread_self();

        rpc_binding_vector_p_t  bvec;
        error_status_t          st, error_inq_st;
        ndr_char                *string_binding;
        static char             *protseq = "ncadg_ip_udp";
        char                    dce_error_string[256];
        int                     i;
        char                    EntryName[256];
        sec_login_auth_src_t    auth_src; /* how login context authorized */
        sec_login_handle_t      login_context; /* handle to login context */
        sec_passwd_rec_t        *keydata; /* password record */
boolean32               reset_password;

#ifdef IBMOS2
        pthread_inst_exception_handler();
#endif

        strcpy( EntryName, ENTRY_NAME ); /* Namespace entry declared in IDL */


/*  Acquire network credentials */
        printf("Establish login context for math server:\n");
        printf("\tCalling sec_login_setup_identity()...\n");
        /* Establish server identity */
        if (sec_login_setup_identity(
                (unsigned_char_p_t) SERVER_NAME, /* principal name */
                sec_login_no_flags, /* flags for new network credentials */
                &login_context,                 /* [out] login context */
                &st))  {
           printf("\tServer identity has been successfully established\n");

           /*  Now retrieve key from local server keytab file */
           printf("\tCalling sec_key_mgmt_get_key()...\n");
           sec_key_mgmt_get_key(
                rpc_c_authn_dce_secret,         /* authentication service */
                KEYFILE,
                SERVER_NAME,                    /* principal name */
                sec_c_key_version_none,         /* use latest version of key */
                (void **) &keydata,             /* [out] key data */
                &st);
           ERRORCK ("sec_key_mgmt_get_key" , st);

           /*  Validate identity; if successful, server is logged in */
           printf("\tCalling sec_login_validate_identity()...\n");
           if (sec_login_validate_identity(
                   login_context,               /* login context                */
                   keydata,                     /* key data                     */
                   &reset_password,             /* [out] TRUE ==> pswd expired  */
                   &auth_src,                   /* [out] how login validated    */
                   &st)) {
                ERRORCK ("sec_login_validate_identity" , st);

                /* Free key data acquired by sec_key_mgmt_get_key() */
                printf("\tCalling sec_key_mgmt_free_key()...\n");
                sec_key_mgmt_free_key(
                        keydata,                /* key data - from get_key()    */
                        &st);
                ERRORCK ("sec_key_mgmt_free_key" , st);
                /* Certify the network authentication service */
                printf("\tCalling sec_login_certify_identity()...\n");
                if (!sec_login_certify_identity(
                        login_context,          /* login context                */
                        &st)) {
                           ERRORCK ("sec_login_certify_identity" , st);
                           exit(1);
                }

                /* Set login context to be default context                      */
                printf("\tCalling sec_login_set_context()...\n");
                sec_login_set_context(
                        login_context,          /* login context                */
                        &st);
                ERRORCK ("sec_login_set_context" , st);
                printf("\tServer login context validated and set as default\n");
           } else {
                printf("\tServer login context NOT valid, stopping server\n");
                exit(1);
           }
        } else {
           ERRORCK ("sec_login_setup_identity" , st);
           printf("\tServer identity has not been successfully established\n");
           exit(2);
        }


/*  Standard server setup/initialization                                        */
        printf("Calling rpc_server_use_progseq()...\n");
        rpc_server_use_protseq(
            (ndr_char *) protseq,
            (unsigned32) 5,
            &st);
        ERRORCK ("rpc_server_use_protseq" , st);

        printf("Calling rpc_server_register_if()...\n");
        rpc_server_register_if(
            ServerID_v1_0_s_ifspec,
            (uuid_t *) NULL,
            (rpc_mgr_epv_t) NULL,
            &st);
        ERRORCK ("rpc_server_register_if" , st);

        printf("Calling rpc_server_inq_bindings()...\n");
        rpc_server_inq_bindings (
            &bvec,
            &st);
        ERRORCK ("rpc_server_inq_bindings " , st);

        for (i = 0; i < bvec->count; i++)
        {
                printf("Calling rpc_binding_to_string_binding()...\n");
                rpc_binding_to_string_binding(
                    bvec->binding_h[i],
                    &string_binding,
                    &st);
                ERRORCK ("rpc_binding_to_string_binding" , st);
                printf("\tstatus = %d\t\t%s\n", st, (char *) string_binding);
        }

        printf("Calling rpc_string_free()...\n");
        rpc_string_free(
                &string_binding,
                &st);
        ERRORCK ("rpc_string_free" , st);

/*  REGISTER SERVER AUTHENTICATION INFORMATION WITH RPC RUNTIME */
        printf("Calling rpc_server_register_auth_info()...\n");
        printf("\tServer Prinicpal name is %s\n", SERVER_NAME );
        rpc_server_register_auth_info(
            (unsigned char *)SERVER_NAME, /* server principal name */
            rpc_c_authn_dce_secret,             /* secret key encryption        */
            NULL,                                /* use default key retrieval */
            KEYFILE,
    &st);
ERRORCK( "rpc_server_register_auth_info", st );

printf("Calling rpc_ep_register()...\n");
rpc_ep_register(
    ServerID_v1_0_s_ifspec,
    bvec,
    (uuid_vector_p_t) 0,
    (unsigned_char_p_t) "Authorizations with client Principal names",
    &st);
ERRORCK ("rpc_ep_register" , st);

printf("Calling rpc_ns_binding_export()...\n");
printf("Exporting the following name:\t%s\n",EntryName);
rpc_ns_binding_export(
    rpc_c_ns_syntax_dce,
    (unsigned char *)EntryName,
    ServerID_v1_0_s_ifspec,
    bvec,
    (uuid_vector_t *) NULL,
    &st);
ERRORCK ("rpc_ns_binding_export" , st);
printf("rpc_ns_binding_export returned %d\n", st);

/* baggage to handle ctrl-C */
sigemptyset(&sigset);
sigaddset(&sigset, SIGINT);
sigaddset(&sigset, SIGTERM);
if (pthread_signal_to_cancel_np(&sigset, &this_thread) != 0)
{
        printf("pthread_signal_to_cancel_np failed\n");
        exit(1);
}


TRY {
        printf("listening...\n");
        rpc_server_listen(
            (unsigned32) 5,
            &st);
        ERRORCK ("rpc_server_listen" , st);
}
CATCH_ALL {
        printf("unregistring interface\n");
        printf("Calling rpc_server_unregister_if()...\n");
        rpc_server_unregister_if(
            ServerID_v1_0_s_ifspec,
            (uuid_t *) NULL,
            &st);
        ERRORCK ("rpc_server_unregister_if" , st);

        printf("Calling rpc_ns_binding_unexport()...\n");
        rpc_ns_binding_unexport(
            rpc_c_ns_syntax_dce,
            (unsigned char *)EntryName,
            ServerID_v1_0_s_ifspec,
            (uuid_vector_t *) NULL,
            &st);
        ERRORCK ("rpc_ns_binding_unexport" , st);

        printf("Calling rpc_ep_unregister()...\n");
        rpc_ep_unregister(
            ServerID_v1_0_s_ifspec,
            bvec,
            (uuid_vector_p_t) 0,
            &st);
        ERRORCK ("rpc_ep_unregister" , st);

        printf("Calling rpc_binding_vector_free()...\n");
        rpc_binding_vector_free(
            &bvec,
            &st);
        ERRORCK ("rpc_binding_vector_free" , st);

        exit (0);
        }
        ENDTRY;
}
```

B.3.3 PAC-Specific Code

This code is specific to the PAC-based server used in section 6.3, "An Example Using DE-Light Web" on page 123. It comprises a Makefile and security module.

Makefile: This Makefile is mostly unremarkable. Note the defintion of three macros in the CFLAGS macro. KEYFILE is the location of the keytab file for this server. This should already be set up from the preparation section described previously. ENTRY_NAME is the CDS name under which the server will register itself. A similar set of macros will be defined in the principal-based security example (see section B.3.4, "Principal-Specific Code" on page 177).

Note that the client for the PAC-based security example is not used because clients are generated using Java (see section 6.3, "An Example Using DE-Light Web" on page 123).

```
#/************************************************************************/
#/* Module Name: Makefile                                                */
#/************************************************************************/
RM       = rm -f

CC       = /usr/bin/xlc_r4
CFLAGS   = -DPAC_SEC \
           -DKEYFILE=\"FILE:/opt/dcelocal/etc/keyfile\" \
           -DENTRY_NAME=\"/.:/Servers/MathServerID\" \
           -I. -I/usr/include/dce
LIBS     = -ldce -ldcepthreads
IDL      = idl

FNAME    = math
CNAME    = $(FNAME)_c
CSTUB    = $(FNAME)_cstub

SNAME    = $(FNAME)_s
SSTUB    = $(FNAME)_sstub

MNAME    = $(FNAME)_m

ALL      = $(CNAME) $(SNAME)
IDLNAME  = $(FNAME).idl
ACLNAME  = $(FNAME).acl

COBJ     = $(CNAME).o $(CSTUB).o
SOBJ     = $(SNAME).o $(MNAME).o $(SSTUB).o security.o

NEWFILES = $(CSTUB).c $(SSTUB).c $(FNAME).h

#/************************************************************************/
#/************************************************************************/
all:  echoall  $(ALL)

echoall:
        @echo "######## Making all in `pwd`..."


#/************************************************************************/
#/* Remove executables and object code                                   */
#/************************************************************************/
aix os2:
        cp Makefile.$@ Makefile

clean:
        @echo "######## Making $@ in `pwd`..."
        -touch $(IDLNAME)
        -$(RM)  core *¬ *.o $(ALL) $(NEWFILES)
```

```
#/***********************************************************************/
#/* Set up the namespace entries                                        */
#/***********************************************************************/
setup:
        setup.sh

#/***********************************************************************/
#/* Run the test shell script                                           */
#/***********************************************************************/
test:
        test.sh

#/***********************************************************************/
#/* Remove the namespace entries                                        */
#/***********************************************************************/
squeaky:
        cleanup.sh

#/***********************************************************************/
#       Rules for making the c code.  This depends on the c main
# program, the include file, and the c stub.
#/***********************************************************************/
$(CNAME).o: $(CNAME).c $(FNAME).h

#/***********************************************************************/
#       Rules for making the s code.  This depends on the s main
# program, the include file, and the s stub.
#/***********************************************************************/
$(SNAME).o: $(SNAME).c $(FNAME).h


#/***********************************************************************/
#       This links the c and s programs.
#/***********************************************************************/
$(CNAME): $(COBJ)
        $(CC) $(CFLAGS) $(COBJ) $(LIBS) -o $@

$(SNAME): $(SOBJ)
        $(CC) $(CFLAGS) $(SOBJ) $(LIBS) -o $@

#/***********************************************************************/
#   this creates the _stub.o files and the math.h file from the math.idl
#   file.
#/***********************************************************************/
$(FNAME).h $(SSTUB).o $(CSTUB).o: $(IDLNAME) $(ACFNAME)
        $(IDL) $(IDLNAME) -I. -I$(INC)
```

security.c: This is the PAC-specific security-checking routine. It checks for appropriate authentication and authorization. In this case authorization simply checks for a valid PAC (that is, the client was authenticated with DCE when issuing the request).

```
/*
* Filename: security.c
*/

#include "math.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <sys/types.h>
#include <dce/rpc.h>
#include <dce/id_epac.h>
#include <dce/binding.h>


#define ERRORCK( proc, st ) \
        { if ( st != error_status_ok) \
                { dce_error_inq_text(st, dce_error_string, &error_inq_st); \
                fprintf(stderr, "%s FAILED, returned %d\n\t%s\n", proc, st,dce_error_string); \
                fflush(stderr);\
                return(0);\
                 }}
```

```
/*
*  This routine checks the binding handle for the appropriate authorization
*  It returns TRUE if and only if the user is authorized; else it returns FALSE.
*/

int is_authorized( handle_t bh )
{
        rpc_authz_cred_handle_t credentials;    /* authn and authz info handle  */
        unsigned_char_p_t       server_principal_name;  /* server name          */
        unsigned32              authn_level,    /* authentication level         */
                                authn_svc;      /* authentication service       */
        unsigned32              authz_svc;      /* type of credentials          */
        boolean32               authorized = TRUE;
        char                    dce_error_string[256];
        error_status_t          error_inq_st, st;

/*
*  Retrieve client authentication and authorization specifications;
*  these will be used to determine if client is authorized to services
*  provided by server.
*/
        printf("\n\nCalling rpc_binding_inq_auth_caller()...\n");
        rpc_binding_inq_auth_caller(
                bh,                     /* binding handle                       */
                &credentials,           /* [out] client's epac or name          */
                &server_principal_name, /* [out] server prin name spec by client*/
                &authn_level,           /* [out] protection level spec by client*/
                &authn_svc,             /* [out] authn service spec by client   */
                &authz_svc,             /* [out] authz info sent by client      */
                &st);
        ERRORCK("rpc_binding_inq_auth_client", st );

/*
*  Validate client's authentication and authorization specifications
*/

        printf("Checking client authentication --\n");
        printf("\tClient specified server name of: %s\n", server_principal_name);

        /* Shared-secret key authentication required                            */
        if (authn_svc == rpc_c_authn_dce_secret)
                printf("\tClient selected proper authentication service\n");
        else {
                printf("\tClient selected wrong authentication service\n");
                authorized = FALSE;
        }

        /* Protection level of encrypted check sums required                    */
        if (authn_level == rpc_c_protect_level_pkt_integ)
                printf("\tClient selected proper protection level\n");
        else {
                printf("\tClient selected wrong protection level\n");
                authorized = FALSE;
        }

        /* EPAC, not name, required credentials                                 */
        if (authz_svc == rpc_c_authz_dce)
                printf("\tClient selected proper authorization service\n");
        else {
                printf("\tClient selected wrong authorization service\n");
                authorized = FALSE;
        }

/*
*  This completes the initial phase of client authorization.  If the server
*  provides an acl manager, authorization would continue by validating the
*  clients credentials (epac) against appropriate access control lists.
*/

        if (authorized)
                return 1;               /* client is AUTHORIZED                 */
        else
                return 0;               /* client NOT AUTHORIZED                */
}
```

B.3.4 Principal-Specific Code

This code is specific to the server with Principal Name-based authorization-checking, which is used in section 5.5.4, "Native DCE Applications" on page 97. It comprises a Makefile and a security module. There are two client programs that are created. math_c is a plain client that can be run from the command line and takes no input parameters. In the first example in 5.5.4, "Native DCE Applications" on page 97, it is called by a CGI script written in Perl. The code listing of math_c.c was shown in B.3.2, "Common Code" on page 168 **.**

cgimath_c is a DCE client that includes functions to handle parameter input from a Web server through CGI. The CGI-handling functions are included in a header file, cgifns.h**,** which is listed in this section along with cgimath_c.c.

Makefile

```
#/*************************************************************************/
#/*  Makefile                                                             */
#/*************************************************************************/
RM       = rm -f

CC       = /usr/bin/xlc_r4
CFLAGS   = -DSTG_SEC \
           -DKEYFILE=\"FILE:/opt/dcelocal/etc/keyfile\" \
           -DENTRY_NAME=\"/.:/Servers/SecurityServerID\" \
           -I. -I/usr/include/dce
LIBS     = -ldce -ldcepthreads -lm
IDL      = idl

FNAME    = math
CNAME    = $(FNAME)_c
CGINAME  = cgimath_c
CSTUB    = $(FNAME)_cstub

SNAME    = $(FNAME)_s
SSTUB    = $(FNAME)_sstub

MNAME    = $(FNAME)_m

ALL      = $(CNAME) $(SNAME) $(CGINAME)
IDLNAME  = $(FNAME).idl
ACLNAME  = $(FNAME).acl

COBJ     = $(CNAME).o $(CSTUB).o
CGIOBJ   = $(CGINAME).o $(CSTUB).o
SOBJ     = $(SNAME).o $(MNAME).o $(SSTUB).o security.o

NEWFILES = $(CSTUB).c $(SSTUB).c $(FNAME).h

#/*************************************************************************/
#/*************************************************************************/
all:  echoall  $(ALL)

echoall:
        @echo "## Making all in `pwd`..."


#/*************************************************************************/
#/* Remove executables and object code                                    */
#/*************************************************************************/
aix os2:
        cp Makefile.$@ Makefile

clean:
        @echo "## Making $@ in `pwd`..."
        -touch $(IDLNAME)
        -$(RM)  core *.o *¬ $(ALL) $(NEWFILES)

#/*************************************************************************/
#/* Set up the namespace entries                                          */
```

```
#/************************************************************************/
setup:
        setup.sh

#/************************************************************************/
#/* Run the test shell script                                            */
#/************************************************************************/
test:
        test.sh

#/************************************************************************/
#/* Remove the namespace entries                                         */
#/************************************************************************/
squeaky:
        cleanup.sh

#/************************************************************************/
#       Rules for making the client code.  This depends on the c main
# program, the include file, and the c stub.
#/************************************************************************/
$(CNAME).o: $(CNAME).c $(FNAME).h

$(CGINAME).o: $(CGINAME).c $(FNAME).h cgifns.h

#/************************************************************************/
#       Rules for making the server code.  This depends on the s main
# program, the include file, and the s stub.
#/************************************************************************/
$(SNAME).o: $(SNAME).c $(FNAME).h

#/************************************************************************/
#       This links the client and server programs.
#/************************************************************************/
$(CGINAME): $(CGIOBJ)
        $(CC) $(CFLAGS) $(CGIOBJ) $(LIBS) -o $@

$(CNAME): $(COBJ)
        $(CC) $(CFLAGS) $(COBJ) $(LIBS) -o $@

$(SNAME): $(SOBJ)
        $(CC) $(CFLAGS) $(SOBJ) $(LIBS) -o $@

#/************************************************************************/
#   this creates the _stub.o files and the math.h file from the math.idl
#   file.
#/************************************************************************/
$(FNAME).h $(SSTUB).o $(CSTUB).o: $(IDLNAME) $(ACFNAME)
        $(IDL) $(IDLNAME) -I. -I$(INC)
```

cgifns.h: This header file contains functions to handle input coming through the Common Gateway Interface (CGI). Parameters from a Web page will be contained in environment variables that this code can process when it is invoked by the Web server. The parameters from a Web page are placed in an array of strings (cgiparams) by the `getcgiparams` function. The `handlecgistart` and `handlecgiclose` functions can be used to start and close output from a program in HTML format. These functions are used in `cgimath_c` to make it a combined DCE client and a CGI program.

```
/************************************************************************/
/**  file: cgifns.h                                                    **/
/**         To add basic CGI programming functionality to C programs.   **/
/**     You can either paste the whole functions into your program,     **/
/**     or you can import this header file and use the functions.       **/
/**                                                                    **/
/************************************************************************/


#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char **cgiparams ;           /* array of strings to hold cgi parameters  */
```

```
/**                                                                      **/
/**     The hex2char() and unescape_url() routines were lifted directly **/
/**     from NCSA's sample program util.c, packaged with their HTTPD.   **/
/**                                                                      **/
/** Convert a two-char hex string into the char it represents **/
char hex2char(char *what) {
   register char digit;

   digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A')+10 : (what[0] - '0'));
   digit *= 16;
   digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 : (what[1] - '0'));
   return(digit);
}

/** Reduce any %xx escape sequences to the characters they represent **/
void unescapeurl(char *url) {
    register int i,j;

    for(i=0,j=0; url[j]; ++i,++j) {
        if((url[i] = url[j]) == '%') {
            url[i] = hex2char(&url[j+1]) ;
            j+= 2 ;
        }
    }
    url[i] = '\0' ;
}

/*************************************************************************/
/** Fn to read CGI input and place all name/value pairs into a list. **/
/** Return list containing name1, value1, name2, value2, ..., NULL   **/
/*************************************************************************/
char **getcgiparams() {
    register int i ;
    char *request_method ;
    int content_length;
    char *cgiinput ;
    char **pairlist ;
    int paircount ;
    char *nvpair ;
    char *eqpos ;

    /** Depending on the request method, read all CGI input into cgiinput **/
    /** (really should produce HTML error messages, instead of exit()ing) **/
    request_method= getenv("REQUEST_METHOD") ;
    if (!strcmp(request_method, "GET")) {
        cgiinput= strdup(getenv("QUERY_STRING")) ;
    }
    else if (!strcmp(request_method, "POST")) {
        /* use strcmpi() if on Windows */
        if ( strcasecmp(getenv("CONTENT_TYPE"), "application/x-www-form-urlencoded")) {
            printf("getcgiparams(): Unsupported Content-Type.\n") ;
            exit(1) ;
        }
        if ( !(content_length = atoi(getenv("CONTENT_LENGTH"))) ) {
            printf("getcgiparams(): No Content-Length was sent with the POST request.\n") ;
            exit(1) ;
        }
        if ( !(cgiinput= (char *) malloc(content_length+1)) ) {
            printf("getcgiparams(): Could not malloc for cgiinput.\n") ;
            exit(1) ;
        }
        if (!fread(cgiinput, content_length, 1, stdin)) {
            printf("Couldn't read CGI input from STDIN.\n") ;
            exit(1) ;
        }
        cgiinput[content_length]='\0' ;
    }
    else {
        printf("getcgiparams(): unsupported REQUEST_METHOD\n") ;
        exit(1) ;
    }

    /** Change all plusses back to spaces **/
    for(i=0; cgiinput[i]; i++) if(cgiinput[i] == '+') cgiinput[i] = ' ' ;
    /** First, split input on "&" to extract the name-value pairs **/
    pairlist= (char **) malloc(256*sizeof(char **)) ;
    paircount= 0 ;
    nvpair= strtok(cgiinput, "&") ;
    while (nvpair) {
        pairlist[paircount++]= strdup(nvpair) ;
        if (!(paircount%256))
            pairlist= (char **) realloc(pairlist,(paircount+256)*sizeof(char **)) ;
        nvpair= strtok(NULL, "&") ;
    }
    pairlist[paircount]= 0 ;    /* terminate the list with NULL */

    /** Then, from the list of pairs, extract the names and values **/
    cgiparams= (char **) malloc((paircount*2+1)*sizeof(char **)) ;
    for (i= 0; i<paircount; i++) {
        if (eqpos=strchr(pairlist[i], '=')) {
            *eqpos= '\0' ;
            unescapeurl(cgiparams[i*2+1]= strdup(eqpos+1)) ;
        } else {
            unescapeurl(cgiparams[i*2+1]= strdup("")) ;
        }
        unescapeurl(cgiparams[i*2]= strdup(pairlist[i])) ;
    }
    cgiparams[paircount*2]= 0 ;   /* terminate the list with NULL */

    /** Free anything that needs to be freed **/
    free(cgiinput) ;
    for (i=0; pairlist[i]; i++) free(pairlist[i]) ;
    free(pairlist) ;

    /** Return the list of name-value strings **/
    return cgiparams ;
}

/** end of the getcgiparams() function **/

/**   **/
void handlecgistart() {
    char *user_name ;
    int i ;

    /** First, get the CGI variables into a list of strings         **/
    cgiparams= getcgiparams() ;

    /** Print the CGI response header, required for all HTML output.        **/
    /** Note the extra \n, to send the blank line after the content header. **/
    printf("Content-type: text/html\n\n") ;

    /** Finally, print out the complete HTML response page.         **/
    printf("<html>\n") ;
    printf("<head><title>CGI Results</title></head>\n") ;
    printf("<body>\n") ;
    printf("<h1>Hello from DCE Client / CGI Program</h1>\n") ;
    printf("Your CGI input variables were:\n") ;
    printf("<ul>\n") ;

    /** Print the CGI variables sent by the user.  Note the list of **/
    /**   variables alternates names and values, and ends in NULL.  **/
    for (i=0; cgiparams[i]; i+= 2)
        printf("<li>[%s] = [%s]\n", cgiparams[i], cgiparams[i+1]) ;

    printf("</ul>\n") ;

    user_name = getenv("REMOTE_USER");
    if (user_name == NULL)
         printf("<p>There is no REMOTE_USER variable set.</p>\n");
    else {
         printf("<p>REMOTE_USER is %s</p>\n",user_name);
         free(user_name) ;
    }

    printf("<pre>");    /* Start preformatted output of stdout */
}

void handlecgiclose() {
    int i;

    printf("</pre>");    /* End preformatted output of stdout */

    printf("</body>\n") ;
    printf("</html>\n") ;

    /** Free anything that needs to be freed **/
    for (i=0; cgiparams[i]; i++) free(cgiparams[i]) ;
    free(cgiparams) ;

    return; /* exit(0) ; */
}
```

cgimath_c.c: This is the combined DCE client and CGI program. It is very similar to math_c.c, but uses functions from cgifns.h to collect CGI input parameters and send its output as HTML. The parameters arriving through CGI are checked to see if they are valid numbers, and if they are, it uses them in the RPC call to the math server.

```
/*
* Filename: cgimath_c.c
*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <dce/rpc.h>
#include "math.h"
#include "cgifns.h"
#include <ctype.h>

#define ERRORCK( proc, st ) \
        { if ( st != error_status_ok) \
                { dce_error_inq_text(st, dce_error_string, &error_inq_st); \
                fprintf(stderr, "%s FAILED, returned %d\n\t%s\n", proc, st,dce_error_string); \
                fflush(stderr);\
        }}

void main(argc, argv)
int argc;
char *argv[];
{
        long int                a=0,
                                b=0,
                                c=0;
        int                     i,j;
        boolean32               isnumber = TRUE;
        rpc_ns_handle_t         ns_handle;      /* namespace binding handle     */
        rpc_binding_handle_t    bh;             /* rpc binding handle           */
        error_status_t          st,
                                error_inq_st;
        char                    dce_error_string[256];

        handlecgistart();                       /*  CGI handling                 */
                                                /*  check all parameters from CGI */
        for(i=0; cgiparams[i]; i+=2) {
                    if (strcmp(cgiparams[i],"a") == 0) {
                                                /* check for numeric parameter from cgi  */
             for(j=0; cgiparams[i+1][j]; ++j)  {
                 if(!isdigit(cgiparams[i+1][j]))  isnumber = FALSE;
             }
             if (isnumber)  a = atoi(cgiparams[i+1]);
                     isnumber = TRUE;
          }
                     if (strcmp(cgiparams[i],"b") == 0) {
                                               /* check for numeric parameter from cgi  */
             for(j=0; cgiparams[i+1][j]; ++j)  {
                 if(!isdigit(cgiparams[i+1][j]))  isnumber = FALSE;
             }
             if (isnumber)  b = atoi(cgiparams[i+1]);
                     isnumber = TRUE;
          }
        }


        printf("Explicit Bindings starting\n");

        printf("Calling rpc_ns_binding_import_begin()...\n");

        rpc_ns_binding_import_begin(
                rpc_c_ns_syntax_default,       /* syntax                     */
                (unsigned char *)ENTRY_NAME,   /* search start point         */
                ServerID_v1_0_c_ifspec,        /* interface handle           */
                NULL,                          /* object UUID                */
                &ns_handle,                    /* binding handle return      */
                &st );                         /* status return              */
        ERRORCK( "rpc_binding_import_begin", st);

        printf("Calling rpc_ns_binding_import_next()...\n");

        rpc_ns_binding_import_next(
                ns_handle,                     /* ns binding handle          */
                &bh,                           /* binding handle return      */
                &st );                         /* status return              */
        ERRORCK( "rpc_binding_import_next", st);

        printf("Calling rpc_ns_binding_import_done()...\n");

        rpc_ns_binding_import_done(
                &ns_handle,                    /* ns binding handle          */
                &st );                         /* status return              */
        ERRORCK( "rpc_binding_import_done", st);

        printf("Calling rpc_binding_set_auth_info()...\n");
        rpc_binding_set_auth_info(
                bh,                            /* binding handle             */
                (unsigned char *)SERVER_NAME,  /* server princ. name         */
                rpc_c_protect_level_pkt_integ, /* send checksum              */
                rpc_c_authn_dce_secret,        /* use DES encryption         */
                NULL,                          /* use login context for info */
                rpc_c_authz_name,              /* send name, not PAC         */
                &st);
        ERRORCK( "rpc_binding_set_auth_info",st);

        printf("calling add() RPC with explicit binding.\n");

        if ( 0 == add(bh,a,b,&c))
        {
                printf("PASSED authorization check..\n");
                printf("<p>\tAfter RPC add\n\t a=%d, b=%d, c=%d\n",a,b,c);
        }
        else
        {
                printf("FAILED authorization check..\n");
        }

        handlecgiclose();                     /* close off CGI output        */

}
```

security.c: This is the security-checking routine that uses Client Principal name information to determine if a client request can be authorized. In this case the authorization check is passed if the client principal is a member of the group good_guys. This code takes the Client Principal name from information contained with the RPC. By default, RPC contains EPAC information and not Client Principal name; so in this example, the client code had to set up the RPC call in the correct way.

```
/*
* Filename: security.c
*/

#include "math.h"
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
#include <pthread.h>
#include <sys/types.h>
#include <dce/rpc.h>
#include <dce/id_epac.h>
#include <dce/binding.h>
#include <dce/pgo.h>
#include <dce/secidmap.h>


#define ERRORCK( proc, st ) \
        { if ( st != error_status_ok) \
                { dce_error_inq_text(st, dce_error_string, &error_inq_st); \
                fprintf(stderr, "%s FAILED, returned %d\n\t%s\n", proc, st,dce_error_string); \
                fflush(stderr);\
                return(0);\
                 }}

/*
*  This routine checks the binding handle for the appropriate authorization
*  It returns TRUE if and only if the user is authorized; else it returns FALSE.
*/

int is_authorized( handle_t bh )
{
        rpc_authz_cred_handle_t credentials;    /* authn and authz info handle  */
        sec_cred_pa_handle_t    priv_attrs;     /* privilege attributes handle  */
        sec_rgy_handle_t        rgy_context;    /* registry context             */
        unsigned32              authz_svc;      /* type of credentials          */
        boolean32               authorized = TRUE;
        char                    dce_error_string[256];
        unsigned_char_t         group_name[] = "good_guys", /* example group */
                                local_client_princ_name[sec_rgy_name_t_size + 1];
        unsigned_char_p_t       global_client_princ_name; /*client principal name */
        error_status_t          error_inq_st, st;

/*
*  Retrieve client authentication and authorization specifications;
*  these will be used to determine if client is authorized to services
*  provided by server.
*/
        printf("\n\nCalling rpc_binding_inq_auth_caller()...\n");
        rpc_binding_inq_auth_caller(
                bh,                     /* binding handle                       */
                &credentials,           /* [out] epac or name for client        */
                NULL,                   /* [out] don't need server name         */
                NULL,                   /* [out] don't need client authn level  */
                NULL,                   /* [out] don't need client authn service*/
                &authz_svc,             /* [out] authz info sent by client      */
                &st);
        ERRORCK("rpc_binding_inq_auth_client", st );

        if ( authz_svc != rpc_c_authz_name )
        {
                printf("Client did not send name, authz_svc != name\n");
                return 0;               /* AUTHORIZATION FAILURE, EXIT!!!!      */
        }

        /*
        *  Display the principal ID of the client
        */
        printf("Display the principal ID of the client.\n");
        printf("\tCalling sec_cred_get_client_princ_name()...\n");
        sec_cred_get_client_princ_name(
                credentials,            /* credentials from binding handle      */
                &global_client_princ_name, /* [out] client name in global format*/
                &st);
        ERRORCK("sec_cred_get_client_princ_name" , st );
        printf("\tClient, %s, requested authorization\n", (char *) global_client_princ_name );


        /*
        *  Use group membership as an example of determining client's
        *  authorization to services provided
        */
        printf("Check authorization -- use group membership as an example:\n");
        printf("\tBind to registry site for cell --\n");
        printf("\tCall sec_rgy_site_open()...\n");
        sec_rgy_site_open(
                NULL,                   /* use registry for local cell          */
                &rgy_context,           /* [out] handle to registry             */
                &st);
        ERRORCK("sec_rgy_site_open" , st);

        printf("\tTranslate global name to cell-relative principal name --\n");
        printf("\tCall sec_id_parse_name()...\n");
        sec_id_parse_name(
                rgy_context,            /* registry handle                      */
                global_client_princ_name, /* principal name in global format    */
                NULL,                   /* [out] cell name       t              */
                NULL,                   /* [out] cell uuid                      */
                local_client_princ_name,  /* [out] cell-relative principal name */
                NULL,                   /* [out] principal uuid                 */
                &st);
        ERRORCK("sec_id_parse_name" , st );

        printf("\tCheck group membership --\n");
        printf("\tCall sec_rgy_pgo_is_member()...\n");
        authorized = sec_rgy_pgo_is_member(
                        rgy_context,    /* registry handle              */
                        sec_rgy_domain_group, /* test group membership  */
                        group_name, /* group name -- see declarations   */
                        local_client_princ_name, /* client princ name   */
                        &st);
        ERRORCK("sec_rgy_pgo_is_member" , st );

        printf("\tCall sec_rgy_site_close()...\n");
        sec_rgy_site_close(
                rgy_context,
                &st);
        ERRORCK("sec_rgy_site_close" , st );

        if (authorized)
                return 1;               /* client is group member & AUTHORIZED  */
        else
                return 0;               /* client not group member & NOT AUTHORIZED */
}
```

Appendix C. Special Notices

This publication is intended to help solution architects, planners and system administrators to understand and implement security features on their intranet and on the Internet based on technology provided by the Distributed Computing Environment (DCE). The information in this publication is not intended as the specification of any programming interfaces that are provided by IBM DFS Web, Transarc Corporation's DE-Light Web client, or any other products mentioned. See the PUBLICATIONS section of the IBM Programming Announcement for the IBM products, or contact the vendors for non-IBM products for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

Appendix D. Related Publications

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

D.1 International Technical Support Organization Publications

For information on ordering these ITSO publications, see “How To Get ITSO Redbooks” on page 189.

- *Administering IBM DCE and DFS Version 2.1 for AIX (and OS/2 Clients)*, SG24-4714
- *Understanding OSF DCE 1.1 for AIX and OS/2, SG24-4616*
- *The Distributed File System (DFS) for AIX/6000*, GG24-4255
- *IBM DSS and DCE Cross-Platform Guide*, SG24-2543
- *Safe Surfing: How to Build a Secure World Wide Web Connection*, SG24-4564
- *Building a Firewall with the IBM Internet Connection Secured Network Gateway*, SG24-2577

D.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	**Subscription Number**	**Collection Kit Number**
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RISC System/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RISC System/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

D.3 Other Publications

These publications are also relevant as further information sources:

- *Firewalls and Internet Security: Repelling the Wily Hacker*, William R. Cheswick & Steven M. Bellovin, Addison-Wesley, ISBN 0-201-63357-4, IBM Form No. SR28-5580
- *Internet Security SECRETS*, John Vacca, IDG Books Worldwide, ISBN 1-56884-457-3
- *Netscape FastTrack Server Administrator’s Guide*, shipped with the product

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL `http://www.redbooks.ibm.com`.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type `GOPHER.WTSCPOK.ITSO.IBM.COM`
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  `http://w3.itso.ibm.com/redbooks`

- **IBM Direct Publications Catalog on the World Wide Web**

  `http://www.elink.ibmlink.ibm.com/pbl/pbl`

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 415 855 43 29 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com

- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	**Order Number**	**Quantity**

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number

- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

List of Abbreviations

ACL	Access Control List
APA	all points addressable
API	Application Programming Interface
CA	Certificate Authority
CDS	Cell Directory Service
CGI	Common Gateway Interface
DCE	Distributed Computing Environment
DES	Data Encryption Standard
DFS	Distributed File System
DLL	Dynamic Link Library
DMZ	Demilitarized Zone
DNS	Domain Name Service
DSS	Directory and Security Server
DSSL	Distributed Systems Series Link
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPD	HTTP Daemon
IBM	International Business Machines Corporation
IDL	Interface Definition Language
ITSO	International Technical Support Organization
JDK	Java Development Kit
MIME	Multi-Purpose Internet Mail Extension
NSA	National Security Administration
ODBC	Open Database Connectivity
OSF	Open Software Foundation
PGP	Pretty Good Privacy
PROFS	Professional Office System
RIP	Routing Information Protocol
RPC	Remote Procedure Call
SET	Secure Electronic Transactions
S-HTTP	Secure Hypertext Transfer Protocol
SNG	Secured Network Gateway
SSL	Secure Sockets Layer
TRPC	Transactional RPC
WWW	World Wide Web

Index

ITSO Redbook Evaluation

Security on the Web Using DCE Technology
SG24-4949-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redeval@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____________

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:
____

____

____

____

What other redbooks would you like to see published?
____

____

____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**
____

____

____

____

____

SG24-4949-00