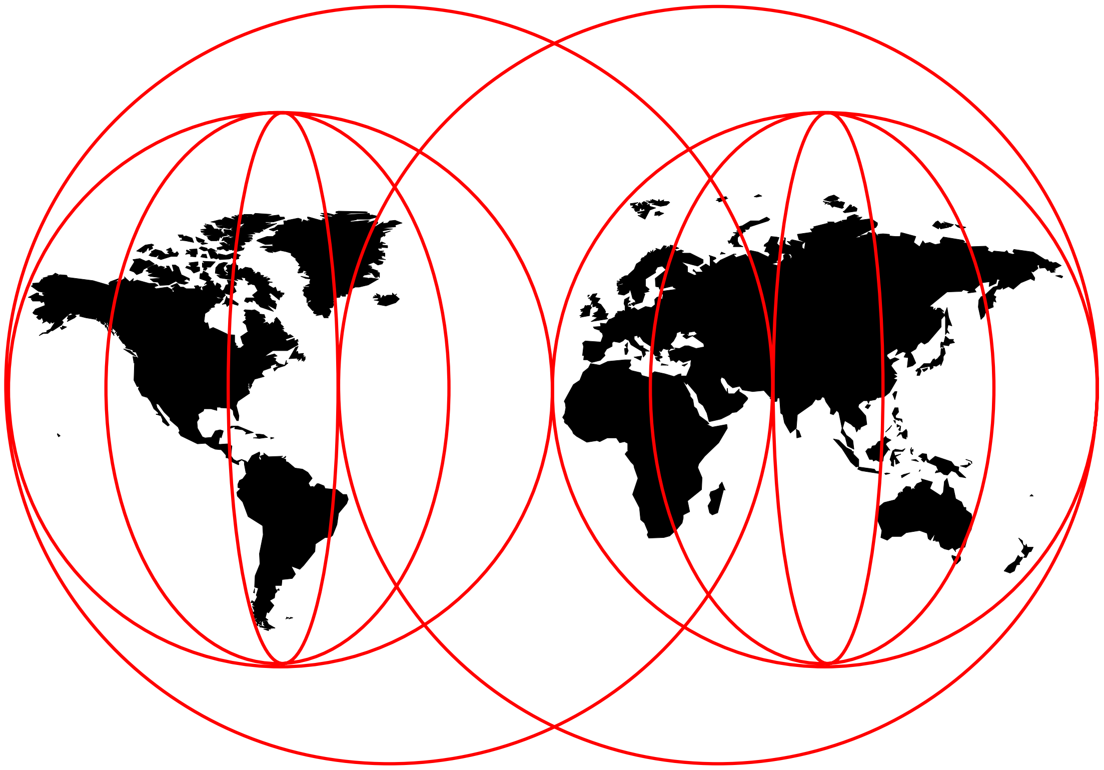
Internet Security in the Network Computing Framework

Marco Pistoia, Kenji Kojima, Narayan Raghu



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-5220-00

International Technical Support Organization SG24-5220-00

Internet Security in the Network Computing Framework

September 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix A, “Special Notices” on page 501.

First Edition (September 1998)

This edition applies to:

- Version 4.6.2.2 of Lotus Domino Go Webserver, North American Edition
- Version 1.0 of IBM WebSphere Application Server
- Version 5.0 of IBM DB2 Universal Database
- Version 3.2 of IBM eNetwork Firewall

for use with the AIX and NT Operating Systems.

Warning

This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. It is recommended that, when the product becomes generally available, you destroy all copies of this version of the book that you have in your possession.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

Contents

Figures

Tables

Preface

This redbook provides an overview of the security component of the Network Computing Framework, as well as specific implementation examples to help explain how to build a secure Network Computing Framework application environment. For example, there are scenarios that show how to configure the various components of the Framework to implement a secure NCF three-tier application. In addition, there are examples that show multiple firewall configurations to protect client/server communication while using the IIOP protocol.

This redbook also describes the architecture of the new Java 1.2 security model and provides many examples of applications and applets, written in Java 1.2, to show you how you can implement the new security features. It also gets very specific about the security measures offered by Lotus Domino Go Webserver 4.6.2.2 and IBM WebSphere Application Server 1.0. Several examples are provided to give you a better understanding of the technologies involved.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the System Management and Networking ITSO Center, Raleigh.

Marco Pistoia is an International Technical Support representative at the Systems Management and Networking ITSO Center, Raleigh. He holds a degree with honors in Pure Mathematics from the University of Rome and a master in Computer Science. His responsibilities at the ITSO are related to all areas of the IBM Network Computing Framework and Internet security. Before joining the ITSO, he was a System Engineer in IBM Italy. He received an Outstanding Technical Achievement Award in 1996.

Kenji Kojima is a network specialist in IBM Japan. He holds a degree in Material Engineering from the University of Tsukuba. He has four years of experience in the Internet security field. He has been with IBM for eight years. His areas of expertise include high performance and availability in AIX. He received a Marketing Excellent Award in 1995.

Narayan Raghu is a Software Engineer in IBM Global Services India Ltd. He has one and a half years of experience in Internet technologies. He holds a degree in Electronics and Communication Engineering from the University of Mysore, India. His areas of expertise include Internet security and e-Commerce. He has worked on IBM India's first large e-Commerce project involving Online Brokerage, and has given several talks on Internet Security and e-Commerce in IBM India.

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

Barry D. Nusbaum
IBM, Systems Management and Networking ITSO Center, Raleigh

Michael H. Conner, Anthony J. Nadalin, Julianne Yarsa
IBM Austin

Ernest L. Evans, Kenneth J. McCauley
IBM Raleigh

David K. Jackson
IBM, Advanced Technology Migration Team, Security

Andreas Weinfurter
IBM Austria

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 511 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

For Internet users	`http://www.redbooks.ibm.com/`
For IBM Intranet users	`http://w3.itso.ibm.com/`

- Send us a note at the following address:

  `redbook@us.ibm.com`

Part 1. NCF Security Components

Chapter 1. An Overview of NCF Security

IBM Network Computing Framework (NCF) for e-business is designed to help developers to build Web-based applications quickly and easily. Through a unified programming model based on Java and JavaBeans, it provides the underlying services needed by network applications. Developers can leverage all of the opportunities coming from the World Wide Web.

IBM NCF is based on a logical, three-tier model, consisting of the following parts:

1. Client

   A client in an NCF environment has the role of requesting the applications and presenting the information, through a Graphical User Interface (GUI). NCF clients are usually Web browsers installed on a personal computer or a network computer. It is required that a Web browser playing the role of a client in an NCF environment is Java-enabled.

2. Web Server

   A Web server in an NCF environment contains business logic and processes that control reading and writing of data. A client machine usually sends a request to the Web server. The Web server processes this request and sends a response back to the client. This response can be static, if it does not depend on the client's input, or dynamic, if it is dependent on the client's input or personalized according with the client's history.

3. Application Server

   An application server in an NCF environment provides data storage and transactional applications used by the Web server processes. A Web server usually sends a dynamic response to a client after interacting with an application server.

The tiers are connected through a set of industry-standard protocols, services and software connectors.

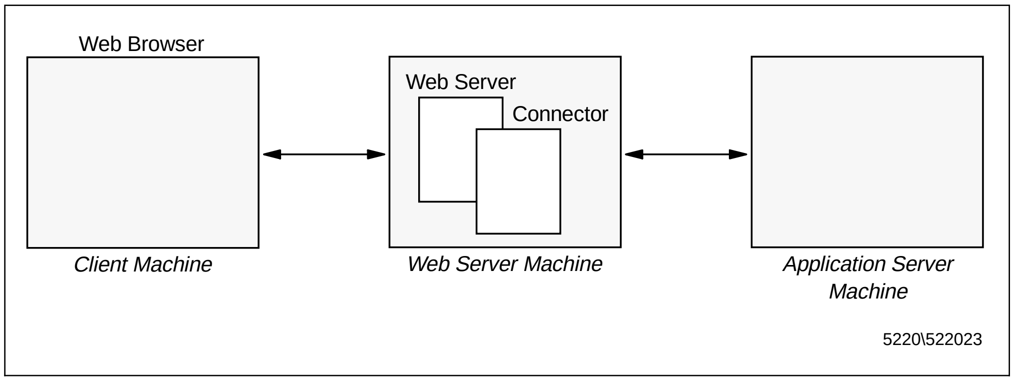


Figure 1. The NCF Logical Three-Tier Model

There are also applications that are completely based on a client/server interaction and do not require the presence of an application server. These also can be considered NCF applications, based on a logical two-tier model.

The rapid growth of Internet technologies has modified the nature of enterprise networks. Today enterprise networks can pass through non-secure networks, and users in non-office environments can reach the secure intranet of their companies even through non-secure channels, without the need of building high-cost private communication lines.

Of course these modifications have carried security risks, the number number of which is increasing. The first security requirement for a network should be to protect private and sensitive resources from the access of non-authorized entities. Moreover, in an e-business environment, security problems are even higher since data, which can also be private and sensitive, flows over non-secure channels. In this case the risk of non-authorized entities accessing private resources on the network is certainly greater.

1.1 NCF Security Architecture

Security architecture in an NCF environment is based on a Public Key Infrastructure (PKI), which makes use of certificate-based authentication and encryption. NCF enforces security on the single components of the Framework and also ensures that communication between the NCF components is secure.

The NCF security architecture is based on the following security components, as shown in Figure 2 on page 5.

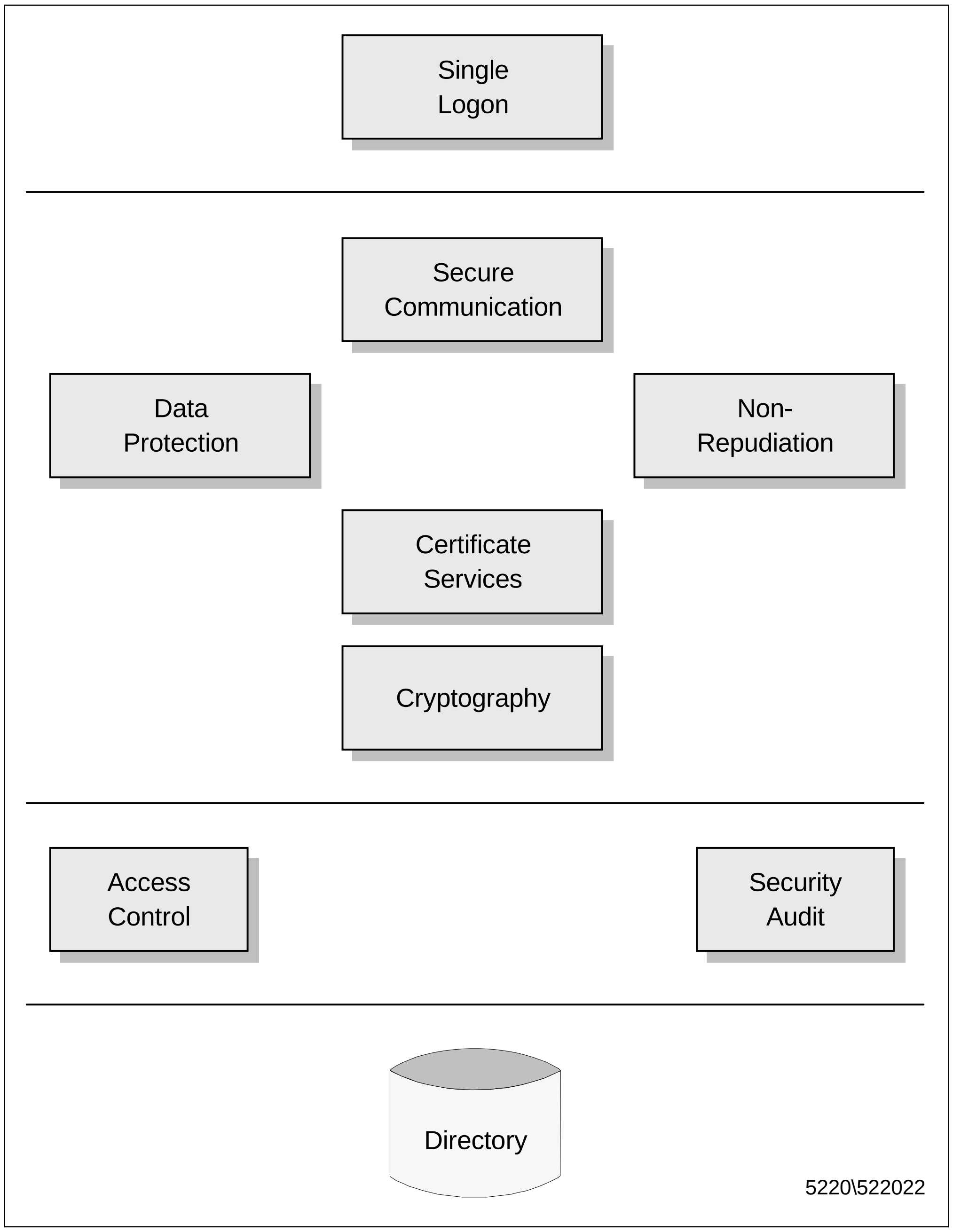


Figure 2. Components of the NCF Security Architecture

1. Single Logon

   In an NCF environment, the user should be able to access all the private resources he or she is authorized to access after a single logon, to a single account with a single password. After that, it is the NCF security architecture that transparently manages all subsequent logons.

2. Secure Communications

   In order to protect user and application data flowing over untrusted network links against intruder attacks, the NCF security architecture supports two secure communication protocols:

   a. The Secure Socket Layer (SSL) protocol grants session-level security by authenticating server and client and by encrypting the communication. The two sides must agree to a session key that will be used to encrypt all the data exchanged.

   b. The Internet Protocol Security (IPSec), also known as Virtual Private Network (VPN) or tunneling support, grants network-layer security by providing a secure channel for data across a network. IPSec must be configured at a firewall on both sides of the network and the two sides must agree to the keys to use as well as the degree of security. The degree of security can be based on an authentication header (including a checksum), encryption or both.

3. Access Control and Auditing

   The NCF security architecture supports resource access control and security auditing in support of businesses' authorization and accountability policies. NCF security policy information is stored and managed in the Lightweight Directory Service Access (LDAP) directory service, which defines a reasonably simple mechanism for Internet clients to query and manage an arbitrary database of hierarchical attribute/value pairs over a TCP/IP communication (you might want to see the Request For Comments RFC1777 at http://info.internet.isi.edu/in-notes/rfc/files/rfc1777.txt).

4. Data Protection and Non-Repudiation

   NCF security architecture provides data protection and non-repudiation services which support protection of stored data and store-and-forward traffic such as e-mail. The NCF security architecture's e-mail protection implements the S/MIME standard on top of its basic data protection services.

5. Certificate Services

   NCF security architecture supports a series of certificate service components:

   - A Certification Authority (CA), for certificate issuance and revocation
   - A Registration Authority (RA), for administration of certificate issuance and revocation
   - An LDAP directory service for certificate and Certificate Revocation List (CRL) distribution and policy management

6. Cryptography

   The NCF security architecture uses cryptographic services that make it possible:

   - To authenticate both sides of a communication
   - To ascertain whether the data was corrupted in transit
   - To modify the contents of a communication in such a way that only the recipient can decipher and read the original message.

For those who are not familiar with cryptography, we recommend the IBM redbook *A Comprehensive Guide to Virtual Private Networks, Vol. I*, SG24-5201, Chapter 2, "A Short Introduction to Cryptography" or the RSA Web site http://www.rsa.com.

1.1.1 NCF and Java Security

Java is the foundation for the NCF architecture, since it is the only programming language needed to access all the NCF components. The Java environment offers several security features, which have been largely enhanced with the new Java 1.2 security model.

The NCF security architecture uses the security features offered by the Java environment:

- The Java 1.2 security API, based on the package java.security and its subpackages:
  - java.security.acl
  - java.security.cert
  - java.security.interfaces
  - java.security.spec
- The Java security interfaces for cryptography, based upon the Java Cryptography Extension (JCE) package 1.2, which provides a framework for encryption and key negotiation.
- The Java 1.2 access control model based upon the Java Naming and Directory Interface (JNDI), which provides Java applications with a unified interface to multiple, heterogeneous naming and directory services in the enterprise. Using this industry-standard, directory-enabled applications become powerful and portable.

Secure communications are provided by SSL in the Java environment.

1.2 Directory Architecture

Even if directory services have been used for a long time, the explosion of distributed and Internet-based computing has generated a proliferation of directory services within organizations. A common situation is that organization often define an employee in several directories - for e-mail systems, for networks, for applications, etc. This way several different repositories, access controls and management interfaces must be created and handled for each directory. An immediate consequence is that high maintenance costs are generated for these organizations.

The NCF networking infrastructure addresses these issues with a four-part, standards-based directory architecture. NCF provides the following key elements of a directory-based enterprise network:

1. APIs and Protocols

   The NCF client interfaces to directory services are based on the industry standard LDAP and JNDI specifications. These APIs and protocols enable applications on any platform or any device to communicate with a network-based LDAP repository.

2. Common Schema

   NCF's common schema enables applications to share the same objects, so that the information for a person or resource is not created, stored and maintained in multiple places across the network. Since an LDAP repository can provide a common schema for key objects such as users, groups, roles and network policies, it can provide also the shared storage for multiple network computing-based applications and allow for common/consistent application access to these objects.

3. Meta-Directory

   NCF defines a set of meta-directory functions for synchronization of information between network repositories in order to ensure interoperability with non-LDAP based repositories. These meta-directory functions provide the ability to map data that is stored in different directories and define the rules by which this data is copied or synchronized between directories.

4. LDAP Repository

   The NCF LDAP repository provides the actual storage and retrieval mechanisms for information about people and resources in the network.

Chapter 2. The New Java 1.2 Security Model

Java programming language is one of the fastest-growing technologies in use on the Internet today. Java is an object-oriented, operating system-independent programming language developed and distributed by JavaSoft, a Sun Microsystems subsidiary.

In this chapter, we talk about the Java Development Kit (JDK) 1.2 security model, to show you the most important security features that it offers. As of the writing of this chapter, JDK 1.2 was still in beta (we used beta 3 and 4 on a Windows NT Server 4.0 platform), and you might find some minor changes from what is mentioned here to what might finally be released. Moreover, at the time of this project, no Web browser, of course, was able to support Java 1.2, so you will see that all the examples of applets that we show run inside the Applet Viewer that comes with JDK 1.2. However, this does not affect any of the security implications that we are going to discuss, and the same applet running on the Applet Viewer should run in a Java 1.2-enabled Web browser, with the same security implications.

2.1 The Need for Java Security

From its inception, Java has shown that it was designed for the net. Java brought about, for the first time on a large scale, the concept of dynamic loading of code from a source outside the system. Though this is very powerful, and adds several features to the system using it, it is also a grave security threat. There could be several risks associated with loading and running remote code. The remote code could steal memory, or CPU time; it could throw in a virus; it could read files on a local system and transmit them to another machine, etc. However, Java is not just for applets any more. Developers now use Java to build stand-alone, enterprise-class applications to enable disparate clients, such as workstations, PCs or Java-based network computers to access legacy databases and share applications across the network. It looks immediately clear, then, that unlike other programming languages and systems, security mechanisms must be an integral part of Java.

Java was designed to offer the following basic security measures:

- Language design features - such as legal type conversions only, no pointer arithmetic, bounds checking on arrays - provide strong memory protection.
- A *sandbox* mechanism controls what a Java program is permitted to do.
- Encryption and digital signatures are used by code owners to embed their certificate into Java classes. In this way, the end user can ascertain who the owner of the code is and whether the class file was altered after having been signed by the owner's certificate.

Java security builds upon three fundamental aspects of the Java Runtime Environment:

- ByteCode Verifier

  The ByteCode Verifier ensures proper formatting of downloaded code. It verifies that the byte code does not violate the type safety restrictions of the Java Virtual Machine (JVM), that internal stacks cannot over/underflow, and that the byte code instructions will have correct typed parameters.

- SecurityManager

  The SecurityManager performs run-time access controls on attempts to perform file I/O, network I/O, create a new ClassLoader, manipulate Threads/ThreadGroups, start processes on the underlying operating system, terminate the JVM, load non-Java libraries (native code) into the JVM, perform certain types of windowing system operations and load certain types of classes into the JVM. For example, the Java applet sandbox, which severely constrains downloaded, untrusted applets to a limited set of functions that are considered to be relatively safe, is a function of the SecurityManager.

- ClassLoader

  The ClassLoader determines how and when Java programs can load codes, and ensures that system-level components within the run-time environment are not replaced.

Java security functionalities, even if built and designed in the language itself, have been changing their features over time, and their evolution has been dependent on the major JDK releases that have been developed until now: 1.0, 1.1 and 1.2.

2.2 The Evolution of the Java Security Model

It must be noted that most of the commercial deployments in the early years of Java was in browsers. Hence, the focus of security in Java has been greatly on applets, and how to protect the client machine from malicious applets.

JDK 1.0 addressed this problem by running all non-local code inside a sandbox. All local code (applications and applets) was by default trusted, and all non-local code, by default was untrusted.

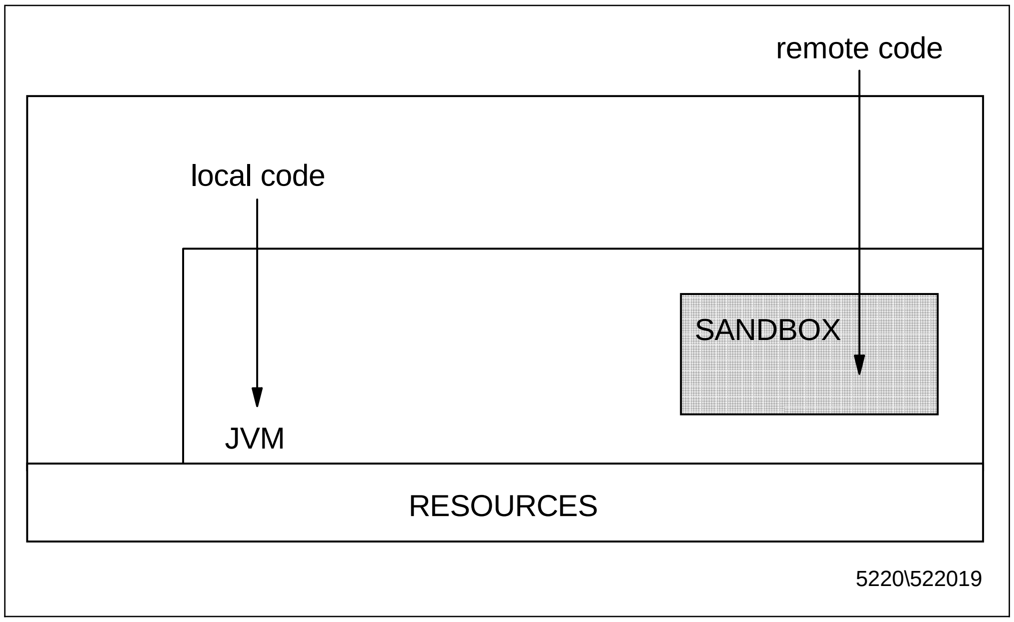


Figure 3. The JDK 1.0 Security Model

Remote applets, though a powerful concept, were shackled by having to run inside a sandbox, and by not being able to perform several operations. They could not read local files and could not write to the disk. They had absolutely no access to the system resources. Moreover they could establish a network connection only with their servicing Web server. This heavily restricted the use of remote applets for all but cosmetic functions to decorate a Web page.

This limitation was solved in JDK 1.1, when signed remote applets were permitted access to several of the system resources that were off limits for those applets without signatures on them. Of course, the client machine had to be informed that certain signatures were trusted, and certain others not. This same policy was actually applied not only to remote applets, but also to other remotely loaded code like, say, remote servlets (see 4.3, “Servlet Sandbox” on page 161). In general, signed remote code was given access to all the system resources, while unsigned remote code was constrained by the Java sandbox. The local code still had complete access to all the system resources.

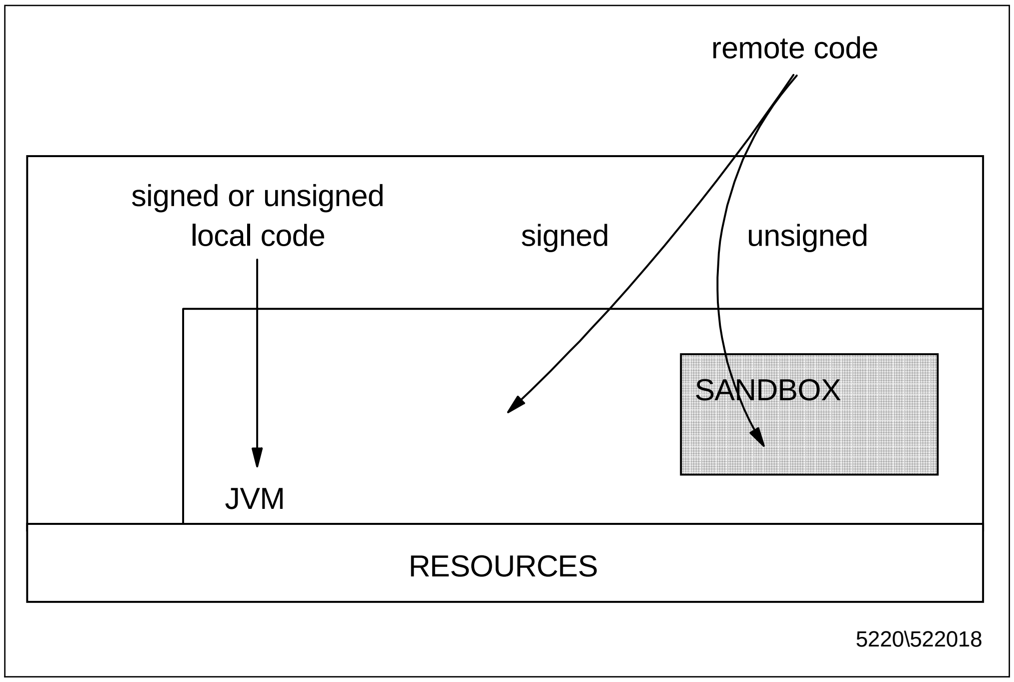


Figure 4. The JDK 1.1 Security Model

Though this opened up interesting possibilities, the system was still rather crude, with all local Java applications enjoying full access to the system resources and all remotely loaded code running inside a sandbox, unless signed by a trusted entity.

This scenario changes in JDK 1.2, where the concept of signed code has been extended to local code as well. With the new security model, all code, whether loaded remotely or locally, signed or unsigned, will get access to system resources based on what is mentioned in a *policy file*. Now two local codes no longer have the same access to system resources if one of them is signed, and the other is not. The whole thing can be specified in the policy file as to what permission you wish to grant to code residing in which code source, or what permission you wish to grant to code signed by whom. This enables you to download and install applications from the Web, and run them by granting them permissions for *only* those things they claim is necessary. This will eliminate codes that have a hidden agenda - such as letting you play a nice game while sending your credit card information or your passwords to a particular server at the same time.

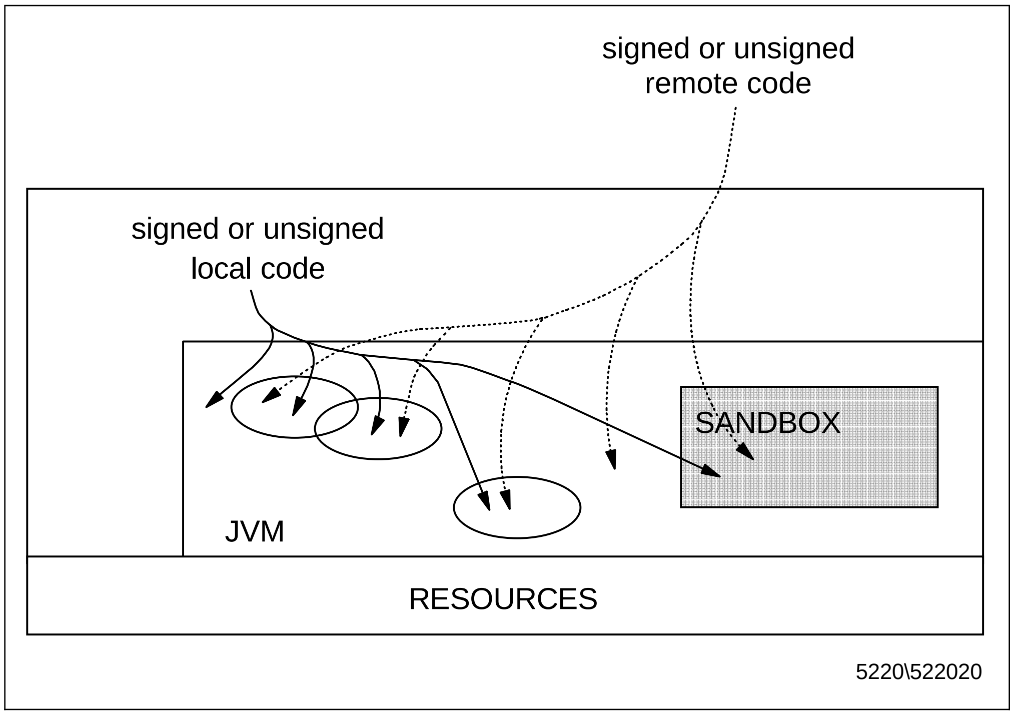


Figure 5. The JDK 1.2 Security Model

Notice, however, that even if the default policy implementation is file-based, application developers can implement their own Policy subclass, providing an implementation of the abstract methods in the java.security.Policy class. There could be multiple instances of the Policy class, even if only one is in effect at any time. The currently installed Policy object can be obtained by calling the getPolicy() method in the Policy class. Codes with permission to reset the policy can change the currently installed Policy object by calling the setPolicy() method in the Policy class. We will discuss this again in 2.4.2.2, “java.policy” on page 26.

Consider now an interesting scenario. You download a little tic-tac-toe program from the Web. It is signed, and you are sure that it will not crash your system, and so you run it. This code simply reads your address book, and sends all the e-mail addresses you have to the database of the nearest junk mailer. Though not very malicious, this is something we all would like to avoid. This is a very likely situation, since more and more software is just being brought off the net, and this trend is likely to continue for a long time. This might lead to fly-by-night software vendors, some of whom might come up with very innovative software, but some of whom you cannot really trust.

As of now, we do not have an option to restrict access to a code to do *only* certain things. You either install the software, or you make do without it.

However, if you are running Java 1.2-enabled software, you could instruct the JVM, through modifications in the policy file, that code loaded from a particular URL (local or remote) and signed by a particular entity is restricted to specific local resources. For example, you may specify in the policy file that the code in question may read files in one particular directory and can do nothing else - cannot open sockets, cannot write or delete any files, etc. This is the fine-grained control mechanism offered by JDK 1.2.

When using JDK 1.2, you can have full control over what each of your programs and applications is permitted to do - this was never possible till now. Similarly, you could spell out the exact things an applet coming from a particular URL can do, or what any programs (applets, applications, servlets) signed by a particular entity can do. Further, if you are talking about multi-user systems, the system administrator could have a default system policy, and each of the users of the system could have their own policy, that permits them to do certain set of things only.

The following table shows the evolution of the security functionalities found in the major JDK releases.

Table 1. Evolution of the JDK Security Functionalities

	JDK 1.0	**JDK 1.1**	**JDK 1.2**
Local unsigned applet resource access	Unconstrained	Unconstrained	Policy based
Local signed applet resource access	Not available	Unconstrained	Policy based
Remote unsigned applet resource access	Constrained by the Java sandbox	Constrained by the Java sandbox	Policy based
Remote signed applet resource access	Not available	Unconstrained	Policy based
Application resource access	Unconstrained	Unconstrained	Policy based
Lexical scoping of privilege modification	Not available	Not available	Stack annotation based with `doPrivileged()`
Cryptographic services for data confidentiality/integrity	Not available	Java Cryptographic Extensions 1.1	Java Cryptographic Extensions 1.2
Digital signature services for code signing	Not available	Java Cryptographic Architecture DSA signature	Java Cryptographic Architecture DSA signature

2.2.1 Java Security API

As you can see in the above table, cryptographic services for data confidentiality and integrity and digital signature services for code signing have appeared in the scene only with JDK 1.1. The Java Security API was built around the java.security package and its subpackages java.security.acl and java.security.interfaces. The first release included primarily cryptography functionalities, which could be incorporated into Java-based applications. The cryptography framework in the Java Security API is designed so that a new algorithm can be added later on without much difficulty and can be used in the same fashion as existing algorithms. For example, even if Digital Signature Algorithm (DSA) is the only built-in algorithm in this release, it is possible to use software from providers to help generate RSA signatures and keypairs for encryption.

The first release for Java security available in JDK 1.1 includes APIs for digital signatures, message digests, key management and Access Control Lists (ACLs). APIs for data encryption and other functionalities, together with their implementations, are released separately in the Java Cryptographic Extensions (JCE) as an add-on package to JDK, in accordance with United States export control regulations. The JCE APIs include block and stream cipher, symmetric and asymmetric encryption and support for multiple mode of operation and multiple encryption.

In JDK 1.2 two new subpackages have been added to the java.security package, and they are java.security.cert and java.security.spec. These packages offer more features to deal with X.509 certificates and to create Certificate Revocation Lists (CRLs) and Certificate Signing Request (CSRs). In particular, java.security.Certificate, that in JDK 1.1 was an interface of abstract methods for managing an identity certificate, is entirely deprecated in JDK 1.2, that offers the entire package java.security.cert to handle certificates. Moreover, the package java.security.cert adds X.509v3 support to certificates.

X.509 Certificates

X.509 is one of the most common formats for signed certificates. It is largely used by JavaSoft, VeriSign, IBM and many other companies for signing e-mail messages, authenticating program code and certifying many other types of data. In its simplest form, an X.509 certificate contains the following data:

1. Version of the certificate format
2. Certificate serial number
3. Identifier of the signature algorithm:
    a. Algorithm ID
    b. Parameters passed to the algorithm
4. Name of the signer of the certificate
5. Period of validity:
    a. Begin date
    b. End date
6. Name of the certified entity
7. Public key of the certified identity:
    a. Algorithm ID
    b. Parameters passed to the algorithm
    c. Public key value
8. Signature (hash code of all the preceding fields, encoded with the signer's private key).

Thus the signer guarantees that a given entity has a particular public key. For more information on X.509 certificates, you can see http://www.ietf.cnri.reston.va.us/ids.by.wg/X.509.html.

As you can see in Table 1 on page 13, another security feature introduced by JDK 1.2 is the lexical scoping of privilege modification, which is a technique enforcing the *least privileged mode*. In other words, this technique permits you to enable only the piece of code that needs the privilege. You could add all the sensitive code in one place, and define that part of the code as privileged, by calling the doPrivileged() method.

2.3 JDK 1.2 Security Architecture

JDK 1.2 introduces a number of new security features which make it easier to enforce access control of protected resources. In earlier versions of Java, JVM resource access was enforced by the sandbox security model, which was a function of the SecurityManager. Extensions were usually limited to features implemented by the platform provider (such as, for example, Web browsers and Web servers). The new JDK 1.2 permission model is much more flexible and even permits application-defined resources to be added to the access control system. Java programs now have the ability to define access restrictions on sensitive resources without requiring the writing of a new SecurityManager or modifying the underlying platform. This means that applets downloaded into a Java-enabled Web browser, or servlets downloaded into a Java-enabled Web server, can add resource access controls to a JVM without having to modify the underlying browser or server implementation.

One of the notable features of the new security model is that most of the access control implementation is contained in the Java security subsystem. Typically Java programs (applications, applets, servlets), components (beans) and libraries (packages) do not need to control any access control code. When a program wants to add protected resources to the JVM, a method call can be added that will check whether the restricted operation is permissible.

The JDK 1.2 access control subsystem introduces new concepts. The first is CodeSource, which is the combination of a codeBase URL and a signer. A codeBase URL represents the location from which the code is loaded. The signer is the entity that signed the code originating from that location. A signer is represented by a set of public keys that should be used to verify the signed code. Each public key is represented as a java.security.PublicKey object, and each URL as a java.net.URL object. The CodeSource is the basis for many permission and access control decisions.

The second concept is the security policy file. The policy file contains a number of `grant` entries which describe the permissions granted to a particular CodeSource. A `grant` entry may contain one or more permissions. A permission is the right to access or use a protected resource. Lastly, a ProtectionDomain is an aggregation of a CodeSource and the permissions granted for the CodeSource as specified in the policy database. Each class file loaded into the JVM via a ClassLoader is assigned to a ProtectionDomain, as determined by the class' CodeSource.

2.3.1 Loading Java Programs

We have already said that the three legs of JVM security are the ByteCode Verifier, the SecurityManager and the ClassLoader. Prior to JDK 1.2, each application had to run its own subclasses of SecurityManager and ClassLoader.

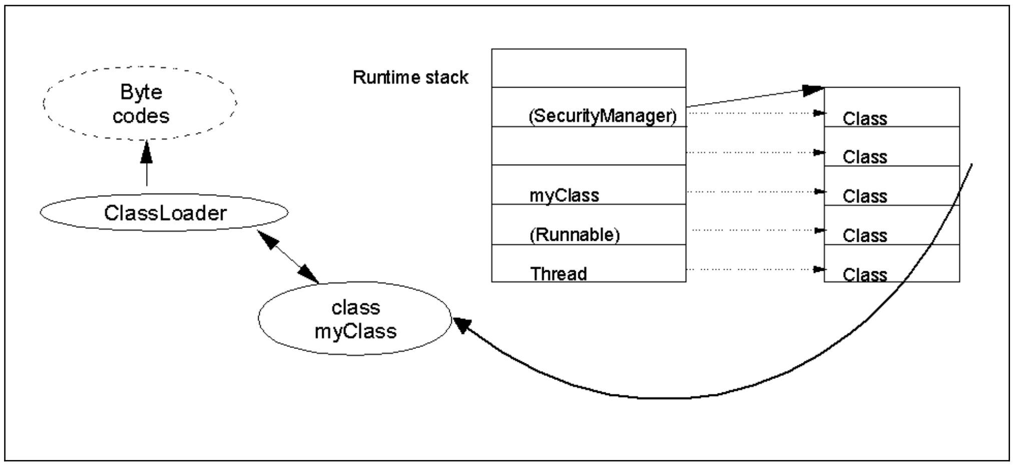


Figure 6. Basic Object Relationships - JDK 1.0 and 1.1

JDK 1.2 simplified the development process by creating a subclass of ClassLoader, called SecureClassLoader. SecurityManager no longer is abstract and can be instantiated and subclassed. Most of its methods now make calls to methods in class AccessController, which provides the access control function in the JDK 1.2. Since most of the SecurityManager methods call AccessController, this greatly simplifies the writing of new SecurityManager subclasses.

The following figure shows the basic object relationships in JDK 1.2, which we explain next.

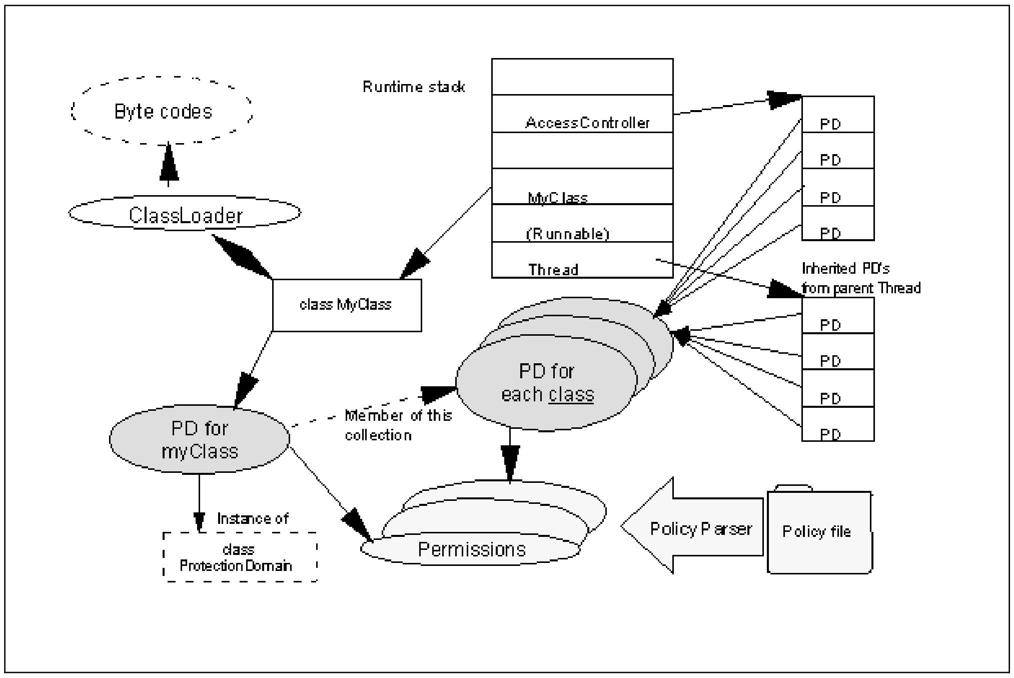


Figure 7. Basic Object Relationships - JDK 1.2

To automatically invoke the new security subsystem, a Java application is started from a native operating system's command line with a special argument, to indicate that the new access control features should be used. The Java runtime creates an instance of the SecureClassLoader, which in turn is used to locate and load the application's class file. A subclass of SecurityManager is created and installed in the Java runtime. The application's main() method is then called with the command line arguments.

The purpose of the change in the Java runtime for starting Java applications is two fold. First, a simple SecurityManager is installed in the system that uses the new Java security access control subsystem. Second, a SecureClassLoader is used to safely and correctly load classes into the Java runtime.

SecureClassLoader has several important purposes. The first is to make sure that searching for the classes is done in the correct order. When the JVM needs a class, JVM first looks for files referenced by the JVM's classpath to see if it is available. Files in the JVM's classpath are intended to be the completely trusted classes that are part of the Java runtime. For example, all the code shipped with the JVM is included in the JVM's classpath, and is therefore considered trusted code. If not found in the JVM's classpath, then an application defined location can be searched (for example, a Web server via a URL request). Finally, code may be part of an application classpath that points to classes that are available in the host file system, but are not part of the JVM's classpath. Classes in the application classpath are typically located on the host's system disk drive (the workstation or the Personal Computer), but the classes are not part of the JVM's fully trusted runtime classes.

Java Classpath in JDK 1.2

As you noticed, in JDK 1.2 classpath does not necessarily indicates trusted (system) code, as it did in JDK 1.0 and JDK 1.1. Instead, there is a java.sys.class.path variable that defines system code locations. There are also a java.class.path property and a CLASSPATH environment variable, but they are used to find non-trusted (application) classes:

- The property java.class.path is used by an application to specify the application's search path of URLs for loading application classes and resources.
- The CLASSPATH environment variable specifies the default value of the property java.class.path. If CLASSPATH is not set, then the default value for java.class.path is set to the current directory.

When loading classes and resources, the ClassLoader will search in the following order:

1. The system classpath (specified by the property java.sys.class.path)
2. The installed extensions (an *extension* is a group of Java packets that implement an API extending the Java platform, as for example JavaServlet, Java3D, JavaManagement, etc.)
3. The application classpath (specified by the property java.class.path).

The option `-classpath` of the `java` command is now shorthand for setting the java.class.path property. Formerly this option was used in JDK 1.0 and 1.1 to override the search path for system classes, but in the new `java` command there is no longer any need to set the system classpath.

The second important purpose of the SecureClassLoader is to create and set the ProtectionDomain information for classes loaded into the JVM. When the SecureClassLoader loads a class into the JVM, the codebase URL and the digital certificate used to sign the class file (if present) are used to create a CodeSource. The CodeSource is used to locate (or instantiate) the ProtectionDomain for the class. The ProtectionDomain contains the permissions that have been granted to the class. Once the class file has been loaded into the JVM, SecureClassLoader assigns the appropriate ProtectionDomain to the class. This ProtectionDomain information, and in particular the permissions in the ProtectionDomain, is used in determining access control during runtime.

Once a Java program starts to run, the SecureClassLoader assists the JVM in loading other classes required to run the program. These classes are also assigned the appropriate ProtectionDomain's based on their CodeSource.

2.3.2 Runtime Access Controls

At various points during a Java program's execution, access to protected resources is requested. This includes network I/O attempts, local file I/O, attempts to create a new ClassLoader or access a program defined resource. To verify whether the running program is allowed to perform the operation, the library routine makes a call to the method SecurityManager.checkPermission(). This method takes a Permission object as argument and determines whether or not it is granted to the current Thread. Each Thread in the JVM contains a number of stack *frames*. Simply stated, these frames contain the method instance variables for each method called in the current Thread. The method checkPermission() walks back to the current Thread's stack frames, getting the ProtectionDomain for each of the classes on the Thread's stack. As each ProtectionDomain in the Thread stack is located, the permission to check is compared to the Permission objects contained in ProtectionDomain. For each stack frame, if the checked permission matches one of the Permission objects in the ProtectionDomain, testing of the permissions continues with the ProtectionDomain of the next stack frame (class) on the stack. This testing repeats until the end of the stack is reached. That is, all of the classes in the Thread have the permission to perform the operation. Thus, the access control check succeeds, typically meaning that the requested operation is able to proceed. If the checked permission is not granted to all classes on the stack (there is no appropriate Permission object in all of the class' ProtectionDomain objects), then a SecurityException is thrown, and access to the resource is denied.

A wrinkle in the above scenario is when a class has a set of permissions, and does not care who its callers may be. For example, a Java bean may be installed on a desktop computer needing to read files from the local disk drive. The bean's class' ProtectionDomain has a permission to read these local files. However, the program loaded from a Web server that calls the bean has a ProtectionDomain that does not have local file read permission. Normally, if the bean were called by the program loaded from the Web server, the bean would be denied access to the files on the local disk drive because the program from the Web server does not have a local file read permission. However, if the bean calls AccessController.doPrivileged(), an annotation is made on the Thread's stack frame indicating that when the checkPermission() method searches for ProtectionDomains, the search stops at this stack frame. The bean may make any number of method calls, but when the checkPermission() method is called on another permission object, the search back through the stack frames to find ProtectionDomain objects stops at this stack frame. Based on the above scenario, the ProtectionDomain objects for the bean will be checked, but the ProtectionDomain objects for the program from the Web server are not checked since the search stopped at the stack frame for the bean. Therefore, the file read operation will succeed.

A subtle aspect of the above doPrivileged() operation is that programs creating new Threads would lose ProtectionDomain information when a new Thread is created. That is, each new Thread creates a new run-time stack. The classes on the stack of the parent Thread are not present in the new Thread. Important ProtectionDomain information is no longer available when a checkPermission() operation is performed. This would give new Threads more permissions than the Threads that created them. To get around this apparent loss of security information, the ProtectionDomains of the parent Thread are attached to (inherited by) a child Thread when it is created. So, unless a doPrivileged() operation is performed in the child Thread, the parent Thread's ProtectionDomain objects are also checked during a checkPermission() operation.

2.3.3 How To Write Privileged Code in Java 1.2

We just mentioned in 2.3.2, “Runtime Access Controls” on page 18 that it is possible to mark Java code as being privileged, by calling the doPrivileged() method for the AccessController class. This class, in the beta 3 release of JDK 1.2, that we largely used in this project, did not define a doPrivileged() method for marking a code segment as privileged. Instead, it implemented two methods, beginPrivileged() and endPrivileged(), that encapsulated the privileged code. These two methods were deprecated in the later beta 4 release and replaced by the doPrivileged() method. In this section, we show you some examples of how this security feature can be implemented.

The normal use of marking a segment of code as privileged is as follows. If you do not need to return any value from the privileged block, you can write a piece of code similar to the following:

```
normal code

class SensitiveAction implements PrivilegedAction
{
  public Object run()
  {
    // privileged code goes here, for example:
    System.loadLibrary("awt");
    return null;  // nothing to return
  }
}

...  // normal code

someMethod()
{
  ...  // normal code

  AccessController.doPrivileged(new SensitiveAction());

  ...  // normal code
}

...  // normal code
```

Figure 8. Privileged Code Returning a null Value

What you did with the above lines of code is to define a new class, named SensitiveAction, that implements the PrivilegedAction interface. This interface is part of the java.security package and it has been shipped only with JDK 1.2 beta 4 or later. It has a single method, run(), that returns an Object. The above example shows how to create an implementation of that interface, providing a concrete implementation for the run() method. As the comment indicates, the run() method must contain the code that we want to be privileged. When the call to doPrivileged() is made, an instance of the PrivilegedAction implementation is passed to it. In general the doPrivileged() method calls the run() method from the PrivilegedAction implementation after enabling privileges, and returns the run() method's return value as the doPrivileged() return value. In this particular case, we have shown how to ignore the return value and return a null value.

If you need to return a value, you can do something similar to the following:

```
class SensitiveAction implements PrivilegedAction
{
  public Object run()
  {
    // privileged code goes here, for example:
    return System.getProperty("user.name");
  }
}

...  // normal code

someMethod()
{
  ...  // normal code

  String user = (String) AccessController.doPrivileged(new SensitiveAction());

  ...  // normal code
}

...  // normal code
```

Figure 9. Privileged Code Returning a non-null Value

The run() method returns as usual an Object, but this time the Object returned is not null. It is instead a String object. As in the lines of code shown in Figure 8 on page 19, this time also an instance of the PrivilegedAction implementation is passed to the doPrivileged() method as parameter. The doPrivileged() method calls the run() method from the PrivilegedAction implementation after enabling privileges, and returns the run() method's return value as the doPrivileged() return value. Notice the use of the casting operator, which is necessary to convert the doPrivileged() method's return value to a String object.

The last case that you should consider is if the action performed in your run() method could throw a *checked* exception (meaning, one of those exceptions listed in the `throws` clause of a method). Then you need to use the PrivilegedExceptionAction interface instead of the PrivilegedAction interface and you also need to catch a PrivilegedActionException in the `try{}catch(){}` block, as shown in the following example:

```
class SensitiveAction implements PrivilegedExceptionAction
{
  public Object run() throws FileNotFoundException
  {
    // privileged code goes here, for example:
    return new FileInputStream("someFile");
  }
}

...  // normal code

someMethod() throws FileNotFoundException
{
  ...  // normal code

  try
  {
    FileInputStream fis =
      (FileInputStream) AccessController.doPrivileged(new SensitiveAction());
  }

  catch (PrivilegedActionException e)
  {
    throw (FileNotFoundException) e.getException();
  }

  ...  // normal code
}
```

Figure 10. Privileged Code if the run() Method Throws a Checked Exception

Reading the source code for PrivilegedActionException, we saw that the getException() method returns an Exception object. For this reason it was necessary to use the casting operator to convert this Exception object to a FileNotFoundException object, as only checked exceptions will be wrapped in a PrivilegedActionException. In effect, PrivilegedActionException is a wrapper for an exception thrown by a privileged action.

The PrivilegedExceptionAction interface and the PrivilegedActionException class also did not exist in previous releases of the JDK and have been shipped only with JDK 1.2 beta 4 or later.

We recommend that you are very careful in your use of the privileged construct, and always remember to make the privileged code segment as small as possible.

2.4 Getting Familiar with JDK 1.2 Security

The security aspects of JDK have changed drastically from JDK 1.1 to JDK 1.2. In this section, we will try to familiarize you with the JDK 1.2 security model.

Installing JDK 1.2 on your computer is a very simple operation and it is not very different from installing JDK 1.1. JDK 1.2 is installed in a root directory that is indicated as ${java.home}. For example ${java.home} could be D:\jdk1.2. When you install JDK 1.2, two security configuration files are installed on your computer.

They are known as policy and properties files. Before examining them, we first need to introduce the concept of *keystore*.

2.4.1 Keystore

A keystore is a database of private keys and their associated certificates or certificate chains, which authenticate the corresponding public keys.

Certificate Chains

The CA who signed a certificate might not be a known, or trusted entity. Hence, for verification purposes, the certificates of the CA and the CA that certified this CA would be required. This is known as a certificate chain. You could see a demonstration of this at http://www.thawte.com.

The default keystore can be found in Windows NT in C:\WINNT\Profiles\Administrator\.keystore. What comes with the JDK 1.2 installation is the default implementation of the keystore, which stores all the keypairs and certificates onto a flat file (encrypted, of course).

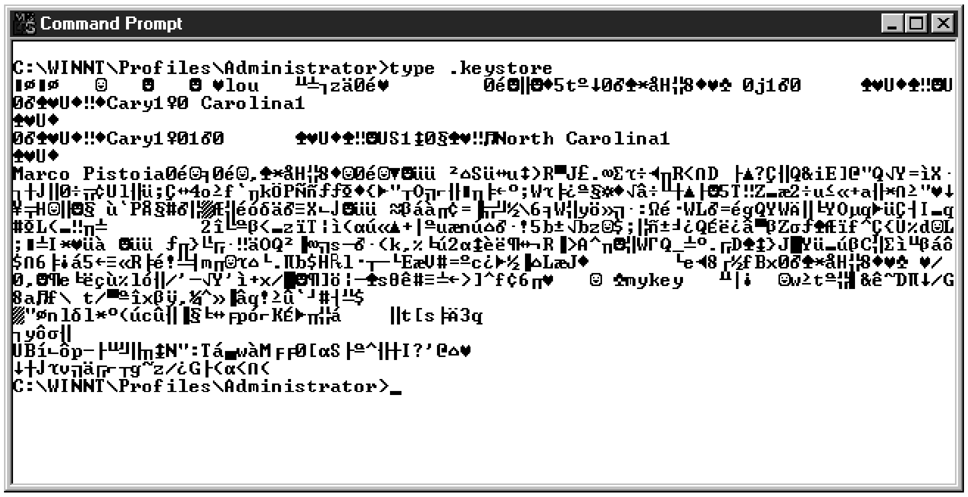
It is possible to change both the implementation and the location of the keystore that comes by default with the JDK 1.2 installation, but the system must be aware of what implementation and location have been selected.

1. The implementation of the keystore is specified, as we are going to see, in the security properties file java.security, defined by the value of the property named keystore.
2. The location of the keystore is in the policy file java.policy, defined by the keystore URL entry.

If you so desire, you could create a new keystore. You might want to do so to, say, store keys and certificates in a database. Then you need to refer to your own keystore implementation in the java.security file and to the location of the keystore in the java.policy file.

As we will see in 2.5.2, “The keytool Utility” on page 29, JDK 1.2 provides the `keytool` command utility for viewing or listing public information about a certificate. Since a keystore is password-protected, you need to enter a password to see that information. However, if you register a certificate as trusted and then try to run an applet signed by that certificate, the JVM automatically retrieves the public key from the keystore, *without* your intervention and *without* asking for the password. The reason for this is that all public information, such as public key and certificate, is stored unencrypted in the keystore, and only the private key is stored password-encrypted to provide confidentiality. The password is used for an integrity check only, so you are prompted to verify that this has not been tampered with.

A demonstration of this can be obtained in the following way. When you use the `type` command on a keystore, amidst all the junk, you can see the values of a certificate you knew existed in the keystore in plain text.



```
C:\WINNT\Profiles\Administrator>type .keystore
▮ø▮ø    ☺    ☻    ☻ ♥lou    ╨╧┐zä0é♥         0é☻║☻♦5tº↓0ó♠×åH¦╠8♦♥♠ 0j1ó0      ♠♥U♦♠‼☻U
0ó♠♥U♦‼♦Cary1♀0 Carolina1
♠♥U♦
0ó♠♥U♦‼♦Cary1♀01ó0       ♠♥U♦♠‼☻US1‡0§♠♥‼ƒNorth Carolina1
♠♥U♦
Marco Pistoia0é☺┐0é☺,♠×åH¦╠8♦☺0é☺▼☻üüü ²△Sü↔u‡)R▀J£.∞Στ÷◄╥R<∩D ├▲?Ç╢Q&iE]@"Q√Y=ìX·
╖┼J║0÷╥¢U1╢ü;Ç↔4o≥f`╓köPÑñƒƒ☼♦<►"┬0╖┌║▮╖╞←º;Wτ╞¿ª§☼♦√â÷╙┼▲├☻5T‼Z▬æ2÷u≤«+a╢×∩≥"♥↓
╪╤H☺║☻§ ù`PÅ§#ó│▓Æ╢éóδäó≡X∟J☻üüü ≈ßáà╓¢=▐╓╜½\6╕W╡│yö»╖·:Ωé·WLó=égQYWä║╚YOµq►üÇ┤I▬q
#☼L<▬‼╓╧         2î╚ªβ<▬zïT¦ì<αú«▲+│ªuænú△ó·!5b±√bz☺§;│ñ±╛¿Q£ëèä▀βZσƒ♠Æïf^Ç<U%d☺L
;▮╧I×♥üà ☻üüü ƒ╥>╙╓·‼äOQ² ▐∞╖s─ó·<k,% ╘ú2α‡èë¶↔¬R ▐>A^╓☻╣║WΓQ_╧º.╓D♠‡>J█Yü▬úßC╣║Σì╙ßáô
$∩6 ╞↓á5←≡«R╞é!╨┤m╓☺τ△└.₧b$HR1·┬─└EæU#=ºc¿►½▐△LæJ♦          └e◄8 ┌½fBx0ó♠×åH¦╠8♦♥♠ ♥/
0,☻¶e╘ëçù%1ó║/'─√Y'ì+x/█☻¶]ö¦─♠s0ê#≡╧←>]^f¢6╓♥    ☺ ♠mykey    ╨│↓  ☺w≥tª╡▌&ê~D₧↓/G
8aƒf\ t/▀ªîxßÿ,%^»▐âq!≥û`┘#┤╩$
▓"øn1ô1*º<úcû║▐§╘↔╒pó┌KÉ►╥╨á     ║t[s╞Ä3q
╖yôσ║
UBí∟ôp─├╙╝║╥‡N":Tá▄wàM╒╒0[αS╞ª^║┼I?'@△♥
↓┼Jτv╖ä╓┌╥g~z/¿G├<α<∩<
C:\WINNT\Profiles\Administrator>_
```

Figure 11. On Typing Out a Keystore

Note the information in the fields of a certificate being clearly visible in the data (`lou`, `mykey`, `Marco Pistoia`, `Cary`, `North Carolina`, `US`), among all other illegible information.

2.4.2 The Properties and Policy Files

After the installation of JDK 1.2, go to ${java.home}\lib\security. You will see two files - java.policy and java.security. These are default files that you can modify or rewrite. In the rest of this section we describe to you these two files. The official source of information is the Sun Web site http://java.sun.com/products/jdk/1.2/docs/guide/security/PolicyFiles.html.

2.4.2.1 java.security

The java.security file, amidst all the comments and explanations, contains important directives. These directives are all of the form:

```
property_variable=value
```

and we describe them all in the following list.

1. Security Provider

   If you are familiar with the JDK 1.1 package java.security and its subpackages, you will be already familiar with the term *provider*. If you are not, we are going to explain it to you right now.

   It is possible, using the java.security package and its subpackages, to write programs that use cryptography functions. These cryptography functions are not part of standard JDK for various reasons, including licensing problems. However, it is possible to install providers that are basically software vendors who supply implementations of these cryptography algorithms and functions. For example, if you have a program that, say, signs a file with an RSA private key you already have, this program cannot run on a basic installation of JDK. RSA libraries are not a part of standard JDK, and JDK supports public key encryption only using DSA. Now, since your keys are RSA keys, you would need a provider to be installed on your machine, to help this program to sign your files.

It is possible in the programs to specify a particular provider alongside the algorithm, but the programmer might choose not to do so. In such a case, the default order of priority can be set by you in this directive.

The exact syntax of this directive is:

`security.provider.`*n*=*classname*

The number *n* is the preference order, or the order in which providers are searched for requested algorithms. The order is 1-based, meaning that 1 indicates the most preferred, followed by 2 and so on. Each provider must implement a subclass of the Provider class. In this directive, *classname* must specify the subclass of the Provider class implemented by the specific provider.

The default provider that comes standard with JDK 1.2 is the Sun provider and its Provider subclass, named Sun, appears in the sun.security.provider package. For this reason the default security provider entry that you will find in your java.security file, immediately after your installation, is:

```
security.provider.1=sun.security.provider.Sun
```

Notice that there must be at least one security provider specification in the java.security file.

2. Policy Provider

   As mentioned before, the whole system functions on a policy that can be specified by you. The policy that you set is read and accessed by the system as an object. Using this directive, you can decide the name of the class that will be instantiated to get the Policy object. If more than one entry is made for this, JDK takes the first one in sequence that is available.

   The default for this directive is:

   ```
   policy.provider=java.security.PolicyFile
   ```

3. Policy File

   As we just mentioned, it is possible to have one single system-wide policy file, and each user can have his or her own policy file in the user's home directory. So by default you will have two entries in your java.security file:

   ```
   policy.url.1=file:${java.home}/lib/security/java.policy
   policy.url.2=file:${user.home}/.java.policy
   ```

   You can also mention more policy URLs, but be sure to number them in ascending order, without missing a number, or else the URLs after the first missing number will not be read.

4. Expand Properties

   This directive specifies whether or not to expand the values given to variables in the policy file. For example, ${java.home} could expand into D:\jdk1.2\. The default is set to `true`, which means that variables will be expanded:

   ```
   policy.expandProperties=true
   ```

   But it can be set to `false`, disabling the function.

5. Allow System Property

   This directive sets the value for the property policy.allowSystemProperty. By default you will find the following entry in your java.security file:

   ```
   policy.allowSystemProperty=true
   ```

which means that it is permissible to pass an extra policy at the command prompt by entering:

`-Djava.policy=`*filename*

Enabling this feature is not really a security hazard, since only those who have access to the command prompt can use this feature. Notice that in order to disable this feature you simply have to comment out the above entry.

6. User-Defined Keystore File

   Using this directive, you can specify name and location for a user-defined keystore file, by modifying the value for the property keystore.user. By default, this directive in the java.security file is:

   ```
   keystore.user=${user.home}${/}.keystore
   ```

   For example, in our environment, the default keystore was the file C:\WINNT\Profiles\Administrator\.keystore.

7. Class to Instantiate for X.509Certificate

   You might have a program that handles all your certificate-related work. This directive is where you declare it, by modifying the value for the property cert.provider.x509 to point to an appropriate implementation of X509Certificate. This program will call the appropriate provider to access the cryptography functions.

   As usual, the default implementation comes with the system:

   ```
   cert.provider.x509=sun.security.x509.X509CertImpl
   ```

8. CRL Provider

   Certification Revocation Lists (CRLs) are maintained by Certification Authorities (CAs) to provide information about the revocation of certificates issued by them. You might wish to have a CA Setup on your system using the keystore and some certificate providers, and so you might need to set up a CRL. This directive lets you decide which implementation of CRL you wish to use, by modifying the value for the property crl.provider.x509. This is the default entry:

   ```
   crl.provider.x509=sun.security.x509.X509CRLImpl
   ```

9. Keystore

   This directive lets you choose the implementation of keystore you wish to employ on your system. The default keystore is implemented by Sun, and stores all keys and certificates on a flat file:

   ```
   keystore=sun.security.tools.JavaKeyStore
   ```

   However, you might wish to have a different system, and so might want to have your own implementation. This directive helps you specify that, by modifying the value for the keystore property.

Here is the default java.security file that came with the installation of JDK 1.2 beta 3. Notice that we have removed all the comments from this file.

```
security.provider.1=sun.security.provider.Sun
policy.provider=java.security.PolicyFile
policy.url.1=file:${java.home}/lib/security/java.policy
policy.url.2=file:${user.home}/.java.policy
policy.expandProperties=true
policy.allowSystemProperty=true
keystore.user=${user.home}${/}.keystore
cert.provider.x509=sun.security.x509.X509CertImpl
crl.provider.x509=sun.security.x509.X509CRLImpl
keystore=sun.security.tools.JavaKeyStore
```

Figure 12. Default java.security File

2.4.2.2 java.policy

As we mentioned in 2.2, "The Evolution of the Java Security Model" on page 10, you can set a security policy for your system through a policy file. java.policy is the default file containing policy directives that comes standard with the JDK 1.2 installation. A user could specify an arbitrary number of policy files in the java.security properties file via the `policy.url`*n* properties (see 3 on page 24). However, there is only one policy (meaning a set of ProtectionDomain objects) in effect for the JVM at any given time. That policy might be the result of processing the information from many policy files. The default policy implementation, via the Policy class, has a public refresh() method that can be used to re-init the policy, eventually re-reading the policy file(s). Notice that there is no automatic policy change: refresh() must be called explicitly.

A policy file contains a list of entries or directives. It may contain first of all a `keystore` entry and must contain one or more `grant` entries. The keystore directive in the policy file is the URL to the keystore file. It is required if one of the `grant` entries in the policy file specify signers whose certificate is stored in a keystore different from the default one. The `keystore` entry in the policy file can be an absolute URL or can be relative to the location of the policy file itself. The location of the policy file is specified in the security properties file java.security by the policy.url.1 property.

Here's an example of a absolute keystore URL:

```
keystore "file:/C:/java/keystore";
```

And here's an example of a relative keystore URL:

```
keystore "aKeystore.file";
```

So, if the policy.url.1 property in java.security defines the policy file as follows:

```
policy.url.1=http://aserver.com/policies/apolicy.file
```

then the keystore file will be loaded from http://aserver.com/policies/aKeystore.file.

Let's talk now about the `grant` directives. In JDK 1.2 you will notice several Permission classes. All these have the same ancestor, java.security.Permission. This is an abstract class, and is subclassed to represent specific accesses. The specific accesses are usually a part of the package where they are most likely to be used - for instance, the permission FilePermission is a part of the java.io package, thus making it java.io.FilePermission, and the SocketPermission class is part of the java.net package, so that you will find java.net.SocketPermission. Most of the permissions can be instantiated by giving two parameters, the first being the target, such as the name of the file, or the socket number, and the second being the permitted action, like read, write, open, listen. In most cases, a set of actions can be specified together as a comma-separated, composite string.

Notice, however, that not all of the Permission classes defined by the JDK 1.2 have applicable actions yet and the second argument to the constructor would be null. The only system Permission classes that *do* have actions are FilePermission (read, write, execute, delete), PropertyPermission (read, write) and SocketPermission (resolve, accept, connect, listen).

We also want to mention that a special permission class exists and it is java.security.AllPermission. This is a permission that implies all permissions. It is introduced to simplify the work to system administrators who might need to perform multiple tasks that require all or numerous permissions. Of course much caution is recommended when considering granting this permission.

The permission policy that you are setting up is to be in the java.policy file. Each permission that you wish to grant must be a statement containing two parts: a CodeSource and a list of permissions.

1. The CodeSource is also comprised of two parts:
   a. codeBase URL

      The codeBase URL indicates where the classes originate from. This field is obtained by the keyword `codeBase` followed by a quoted string indicating the URL, for example:

      ```
      codeBase "http://www.redbooks.ibm.com"
      ```

      This field is optional. If omitted, then the code could be from any source.

   b. Digital certificate used to sign the classes

      This field is obtained by the keyword `signedBy` followed by a quoted string indicating the name assigned to a digital certificate used to sign the classes. More signers comma-separated are allowed. So, for examples, correct entries could be:

      ```
      signedBy "Julie"
      ```

      or

      ```
      signedBy "Julie,Tony"
      ```

      This field is optional. If omitted, AccessController will not care whether the code is signed or not.
2. Each permission of the list is comprised of four parts:
   a. The keyword `permission`.

      This field is required.

   b. The fully qualified name of the Permission class.

      This includes the package name, for example `java.util.PropertyPermission`. This field also is required.

   c. A quoted string naming the target of the Permission class.

      For example, `"java.version"` could be the target for a PropertyPermission, `"D:\\Works\\Stats.txt"` could be the target for a FilePermission. The only

Permission class that this target field is not applicable to is the special java.security.AllPermission.

d. A quoted string naming the actions requested.

   As we said, a set of actions can be specified together as a comma-separated composite string. For example, `"read"` or `read,write,delete,execute"` could be actions for a FilePermission, `"resolve,accept"` or `"listen"` could be the actions for a SocketPermission. Action fields are not applicable to all Permission classes, but only to FilePermission, PropertyPermission and SocketPermission.

e. The keyword `signedBy` followed by a quoted string naming a name that has been assigned to a digital certificate signing the Permission class.

   For example a valid entry could be

   `signedBy "Julie"`

   This field is optional. It may be necessary to prevent spoofing when the Permission class is not resident in the Java runtime but is loaded from over the network.

Some examples will clarify.

The following code lines, added to your java.policy file, will grant a FilePermission to all code, regardless of the signer and the codeBase:

```
grant {
  permission java.io.FilePermission ".tmp", "read,write";
};
```

The following code lines will grant three permissions to the code that is signed by both Tony and Larry and that comes from http://www.ibm.com:

```
grant codeBase "http://www.ibm.com", signedBy "Tony,Larry" {
  permission java.lang.RuntimePermission "print.queueJob";
  permission java.io.FielPermission "C:\\tmp\\file1", "write,delete,execute";
  permission java.net.SocketPermission "9.24.104.51:1345", "connect";
};
```

The following code lines, added to your java.policy file, will grant a permission to the code that is signed by both Tony and Marco, and only if the bytecodes implementing the class com.ibm.PCPermission are genuinely signed by Julie:

```
grant signedBy "Tony,Marco" {
  permission com.ibm.PCPermission "Token-ring", "activate", signedBy "Julie";
};
```

We will see several other examples of policy files in the next sections. Notice that the syntax of these entries is very important. Omitting even a simple comma could create serious problems on your system.

Unfortunately the strings for a file path must be written in a platform-dependent format. The above examples are appropriate on a Windows NT system. Strings are processed by java.io.StreamTokenizer, which considers a back slash as an escape string. So two back slashes are required to indicate one single back slash. This is the reason why the file C:\tmp\file1 is indicated in the above statement as `C:\\tmp\\file1`. However, if the policy.expandProperties in the java.security properties file is set to `true` (see 4 on page 24), you can write portable policy files. For example, the ${/} variable could be used, and it would be converted to the correct format, dependent on the platform.

2.5 JDK 1.2 Security Tools

In order to use the new security features provided by JDK 1.2, developers need four utilities that are automatically installed on their computers when they install the development kit itself. Those who are already familiar with Java 1.1 should know the `jar` command line tool, needed to compress one or more Java components into one single file. This utility is shipped with the new version too. The command line utility of JDK 1.1, `javakey`, has been replaced in JDK 1.2 by `keytool` and `jarsigner`. The Policy Tool is a Graphical User Interface (GUI) that assists a user in generating, editing, importing or exporting a security policy. It also is launched from the command prompt, by issuing the command `policytool`.

We will take these utilities one by one, and see how they work.

2.5.1 The jar Utility

JAR stands for Java ARchive, and is pretty similar to tar, in the sense that it is both a file format and a compression system. The solution is absolutely portable between all Java platforms. Just enter `jar` at the command prompt to get options. Notice that the `jar` tool automatically preserves the directory tree of the file archived. For further information about the JAR files, see the IBM redbook *Network Computing Framework Component Guide*, SG24-2119.

The `jar` function is pretty similar in JDK 1.2 to what it was in JDK 1.1. JAR files acquire specific significance, since the old `javakey`, and its newer version, `jarsigner`, can sign only JAR files. This technique of creating a JAR file, and then signing it, can be effectively used for code signing, and is currently being used for applets and servlets in JDK 1.1. However, in JDK 1.2, signed JAR files might be required for running local applications as well, since permissions can be set for code signed by different vendors based on the signatures.

As in JDK 1.1, you can create a JAR file in JDK 1.2 by giving the command

```
jar cvf myjarfile.jar component1.class component2.class
```

The above command generates the JAR file myjarfile.jar from the two files component1.class and component2.class.

2.5.2 The keytool Utility

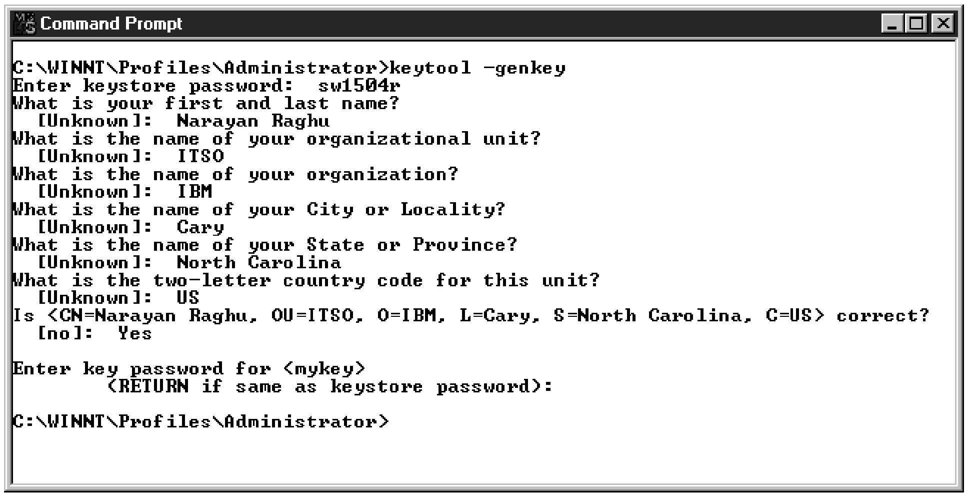
The `keytool` is a key and certificate management command line tool. It helps in storing, creating and managing certificates, certificate requests and keypairs, by interacting with a keystore.

The options that come along with `keytool` are explained clearly in the documentation accompanying your JDK 1.2 installation. However, we illustrate its options with an example. We show how to create a keystore and a self-signed certificate. We also show how this looks when viewed by a tool that displays the Basic Encoding Rules (BER) encoding of the certificate.

This is the command we used to create a new keystore and a keypair:

```
keytool -genkey
```

All the options that are necessary are asked for at the command prompt, and those that are optional are given default values if no options are specified while using `keytool`.



```
C:\WINNT\Profiles\Administrator>keytool -genkey
Enter keystore password:  sw1504r
What is your first and last name?
  [Unknown]:  Narayan Raghu
What is the name of your organizational unit?
  [Unknown]:  ITSO
What is the name of your organization?
  [Unknown]:  IBM
What is the name of your City or Locality?
  [Unknown]:  Cary
What is the name of your State or Province?
  [Unknown]:  North Carolina
What is the two-letter country code for this unit?
  [Unknown]:  US
Is <CN=Narayan Raghu, OU=ITSO, O=IBM, L=Cary, S=North Carolina, C=US> correct?
  [no]:  Yes

Enter key password for <mykey>
        (RETURN if same as keystore password):

C:\WINNT\Profiles\Administrator>
```

Figure 13. Creating a Keypair and a Self-Signed Certificate Using keytool

This just created a self-signed certificate for us, and placed it in a newly created keystore. Note how all the mandatory fields were prompted for, and the optional ones (the keyname and the validity period of the certificate) were set to default (`mykey` and `90 days`).

Alternately, we could have given the long option, as described in the JDK 1.2 documentation, specifying everything, and leaving nothing to default, by giving the following command:

```
keytool -genkey
-dname "CN=Narayan Raghu, OU=ITSO, O=IBM, L=Cary, S=North Carolina, C=US"
-alias narKey -keypass sw1504r
-keystore C:\WINNT\Profiles\Administrator\newKeyStore
-storepass sw1504r -validity 90
```

Note that all this will have to be entered on one line, though we have split it up for clarity.

This command would have created for us a certificate with the domain options given and with a validity of 90 days. It generates a key pair (a public key and associated private key) and wraps the public key into an X.509v1 self-signed certificate, which is stored as a single-element certificate chain. This certificate chain and the private key are saved in the newKeyStore keystore.

Coming back to the certificate we created with those default options, this certificate can be exported from the keystore in base64 format.

base64 Format

base64 format is a commonly used Internet standard. You could encode binary data in base64 by rearranging the bits of the data stream in such a way that only the 6 least significant bits are used in every byte. For more details, see RFC1421 at http://info.internet.isi.edu/in-notes/rfc/files/rfc1421.txt.

We downloaded the BERViewer tool from http://members.xoom.com/Aram_Perez/BERViewer.htm. This tool allows you to view certificates encoded in BER format. However BERViewer takes only binary file inputs. With another little program (which uses BSAFE, a toolkit from RSA), we converted base64 format to binary format, and then were able to view the certificate with the BERViewer tool.

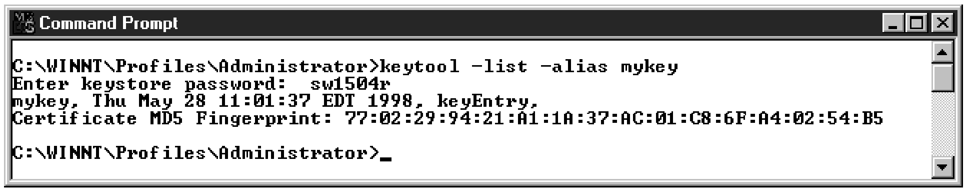
ASN.1 and BER

Abstract Syntax Notation #1 (ASN.1) is a formal notation of representing X.509 certificates in the format Type Length Value (TLV). The Type field, one byte long, specifies the type of data. The type could be INTEGER, OID (OBJECT IDENTIFIER), SEQUENCE, BIT STRING, NULL, etc. Each of these types has a specific code associated with it. The Length field (which could be any number of bytes so long as the most significant bit is a 1, and only one byte long if it is 0) specifies the length of the Value field. If the type is SEQUENCE, the value contains one or more group of TLVs, and so on. Thus a tree is created, with the topmost TLV containing all the TLVs under it, and its length being the cumulative length of all the TLVs (not just the sum of lengths of all the TLVs). In this sense, we say that SEQUENCE is a construct. Other constructs are OPTIONAL and CHOICE.

ASN.1 gives then a precise definition of the structure of a certificate file. The BER format describes precisely how to save this structure in a binary file. The BER format, in other words, describes how to encode data of the types specified above and constructs such as SEQUENCE, OPTIONAL and CHOICE.

We then viewed the entry associated with the key pair that has been generated, as shown, issuing the command:

```
keytool -list -alias mykey
```



```
C:\WINNT\Profiles\Administrator>keytool -list -alias mykey
Enter keystore password:  sw1504r
mykey, Thu May 28 11:01:37 EDT 1998, keyEntry,
Certificate MD5 Fingerprint: 77:02:29:94:21:A1:1A:37:AC:01:C8:6F:A4:02:54:B5

C:\WINNT\Profiles\Administrator>_
```

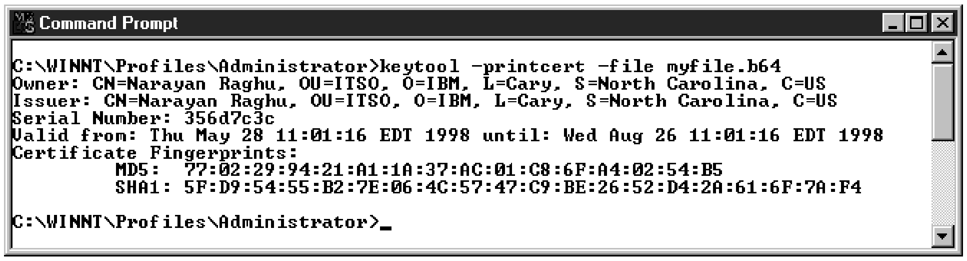
Figure 14. Seeing the Listed Entry in the Keystore

To export the certificate in base64 format, the full command given was:

```
keytool -export -alias mykey -file myfile.b64
```

It prompted us for the password, which we gave, and the output was written into the file myfile.b64. So, we viewed the contents of the file myfile.b64 using the `keytool` again. Exactly, the command entered was:

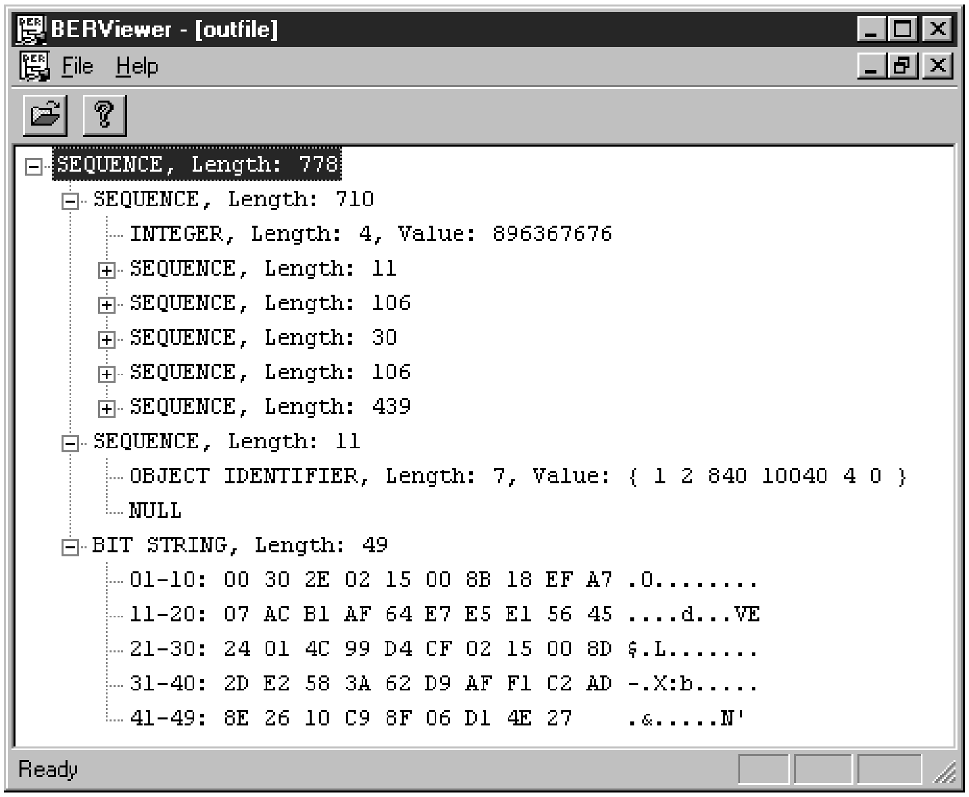
```
keytool -printcert -file myfile.b64
```



```
C:\WINNT\Profiles\Administrator>keytool -printcert -file myfile.b64
Owner: CN=Narayan Raghu, OU=ITSO, O=IBM, L=Cary, S=North Carolina, C=US
Issuer: CN=Narayan Raghu, OU=ITSO, O=IBM, L=Cary, S=North Carolina, C=US
Serial Number: 356d7c3c
Valid from: Thu May 28 11:01:16 EDT 1998 until: Wed Aug 26 11:01:16 EDT 1998
Certificate Fingerprints:
         MD5:  77:02:29:94:21:A1:1A:37:AC:01:C8:6F:A4:02:54:B5
         SHA1: 5F:D9:54:55:B2:7E:06:4C:57:47:C9:BE:26:52:D4:2A:61:6F:7A:F4

C:\WINNT\Profiles\Administrator>_
```

Figure 15. Viewing the Certificate Written to the File

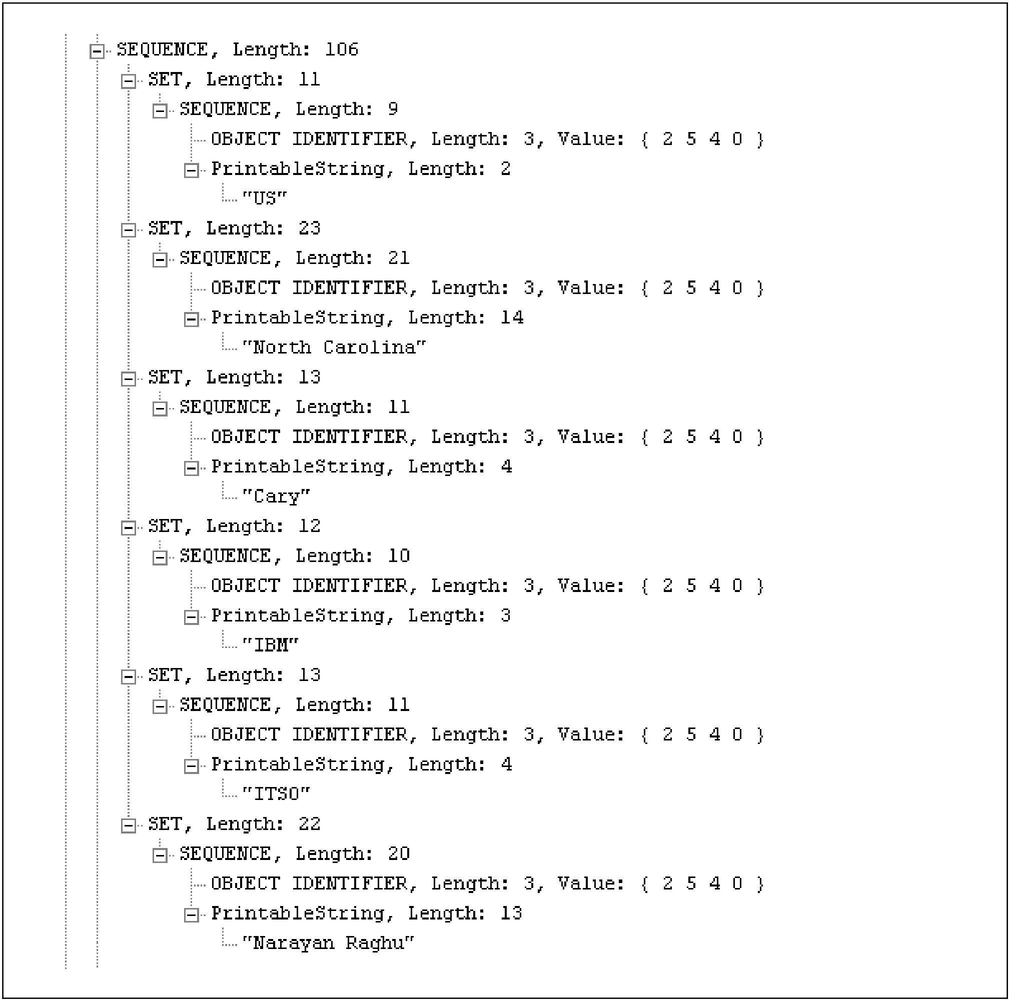
Then we converted it to binary, and viewed it on the BERViewer program we mentioned. At the top level, this is how the certificate looks.



```
SEQUENCE, Length: 778
   SEQUENCE, Length: 710
      INTEGER, Length: 4, Value: 896367676
      SEQUENCE, Length: 11
      SEQUENCE, Length: 106
      SEQUENCE, Length: 30
      SEQUENCE, Length: 106
      SEQUENCE, Length: 439
   SEQUENCE, Length: 11
      OBJECT IDENTIFIER, Length: 7, Value: { 1 2 840 10040 4 0 }
      NULL
   BIT STRING, Length: 49
      01-10: 00 30 2E 02 15 00 8B 18 EF A7 .0........
      11-20: 07 AC B1 AF 64 E7 E5 E1 56 45 ....d...VE
      21-30: 24 01 4C 99 D4 CF 02 15 00 8D $.L.......
      31-40: 2D E2 58 3A 62 D9 AF F1 C2 AD -.X:b.....
      41-49: 8E 26 10 C9 8F 06 D1 4E 27    .&.....N'
```

Figure 16. A Self-Signed Certificate

Note the two SEQUENCEs and the BIT STRING within the upper SEQUENCE. The first inner SEQUENCE contains the information about the certificate, and the second, information about the signing algorithm. The BIT STRING contains the signature. Let's pry into this some more, and see the particulars of this certificate.



```
SEQUENCE, Length: 106
  SET, Length: 11
    SEQUENCE, Length: 9
      OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 0 }
      PrintableString, Length: 2
        "US"
  SET, Length: 23
    SEQUENCE, Length: 21
      OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 0 }
      PrintableString, Length: 14
        "North Carolina"
  SET, Length: 13
    SEQUENCE, Length: 11
      OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 0 }
      PrintableString, Length: 4
        "Cary"
  SET, Length: 12
    SEQUENCE, Length: 10
      OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 0 }
      PrintableString, Length: 3
        "IBM"
  SET, Length: 13
    SEQUENCE, Length: 11
      OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 0 }
      PrintableString, Length: 4
        "ITSO"
  SET, Length: 22
    SEQUENCE, Length: 20
      OBJECT IDENTIFIER, Length: 3, Value: { 2 5 4 0 }
      PrintableString, Length: 13
        "Narayan Raghu"
```

Figure 17. Inside a Certificate

Notice how all the information we entered in the certificate is encoded in the certificate. This is a standard X.509v1 Certificate. Note that by default, `keytool` uses DSA for encryption and SHA-1 for digesting the certificate information. For more information about these algorithms, we'd recommend the RSA FAQ Web site, http://www.rsa.com/rsalabs/newfaq/.

There is a huge set of options along with keytool, and you could see all of them by typing:

`keytool -help 2>` *filename*

The list of options is more than one screen, and so the redirection to a file whose name is specified by *filename*. Important among them are the options to import CA certificates and import signed certificates. However, most of the popular CAs do not seem to be accepting the CSR that JDK 1.2 `keytool` generates, ostensibly owing to the SHA1/DSA signature that is on the Certificate Signing Request (CSR), while the de facto standard is MD5/RSA. So to really get this thing up and going, you would need to install the software of a provider to enable you to use RSA algorithm for public key encryption. Once that is done, you could even use `keytool` to act as a Certification Authority. However, if you intend to handle RSA signatures, you will have to use a provider's software for that.

2.5.3 The jarsigner Utility

The `jarsigner` utility can be used to sign JAR files, and to verify signatures on JAR files. Here's how it works. As usual, you can enter the command:

```
jarsigner -help
```

to get a list of all the valid options.

To use `jarsigner`, we first created a couple of dummy text files, and jared them by issuing:

```
jar cvf nuts.jar nuts.txt bingo.txt
```

Then we signed the resulting nuts.jar JAR file by launching the command

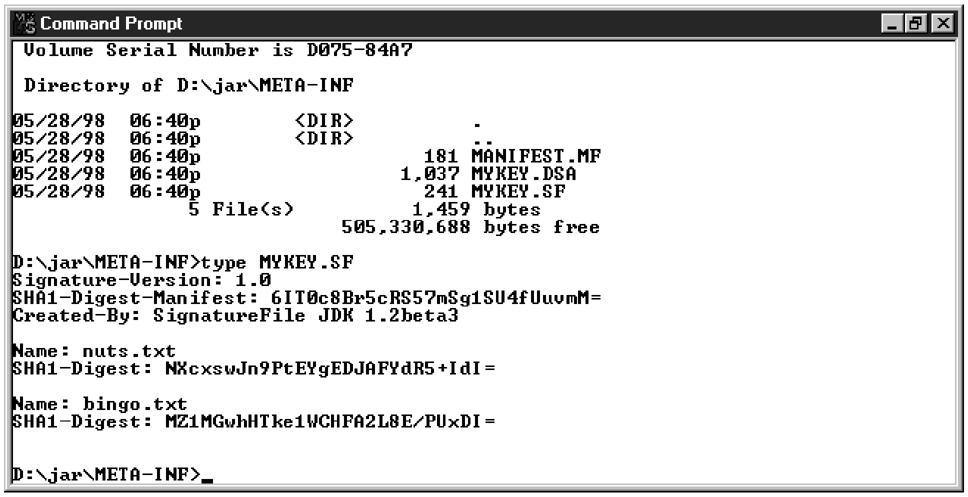
```
jarsigner nuts.jar mykey
```

Note how we didn't have to go through the cumbersome process of creating a sign_directive file, and specifying a signature filename as in JDK 1.1 (see for example the sign.direc file we created in 4.1.2, “Users” on page 128). The signature is stored within the JAR file itself now. You will not find a nuts.jar.sig file as you would have in JDK 1.1. Actually, the signature is added within the JAR file itself. It's something like unjaring it, adding the signature and jaring it again.

The signature information is stored in the JAR file itself in the files MYKEY.SF and MYKEY.DSA. The former has the digest information, and the latter, the signature, along with the signer's certificate in BER format. You can view these entries using the `jar` utility to blow up the JAR file:

```
jar xf nuts.jar
```

You will then see a directory called META-INF created in the directory where you blew up the JAR file. You will find MYKEY.DSA and MYKEY.SF in that directory. We verified, using the BERViewer mentioned earlier that the DSA file contains the signature and the CA Certificate. We also typed out the content of the SF file to verify that it contained the digest information.



```
 Volume Serial Number is D075-84A7

 Directory of D:\jar\META-INF

05/28/98  06:40p        <DIR>          .
05/28/98  06:40p        <DIR>          ..
05/28/98  06:40p                   181 MANIFEST.MF
05/28/98  06:40p                 1,037 MYKEY.DSA
05/28/98  06:40p                   241 MYKEY.SF
               5 File(s)          1,459 bytes
                            505,330,688 bytes free

D:\jar\META-INF>type MYKEY.SF
Signature-Version: 1.0
SHA1-Digest-Manifest: 6IT0c8Br5cRS57mSg1SU4fUuvmM=
Created-By: SignatureFile JDK 1.2beta3

Name: nuts.txt
SHA1-Digest: NXcxswJn9PtEYgEDJAFYdR5+IdI=

Name: bingo.txt
SHA1-Digest: MZ1MGwhHTke1WCHFA2L8E/PUxDI=


D:\jar\META-INF>_
```

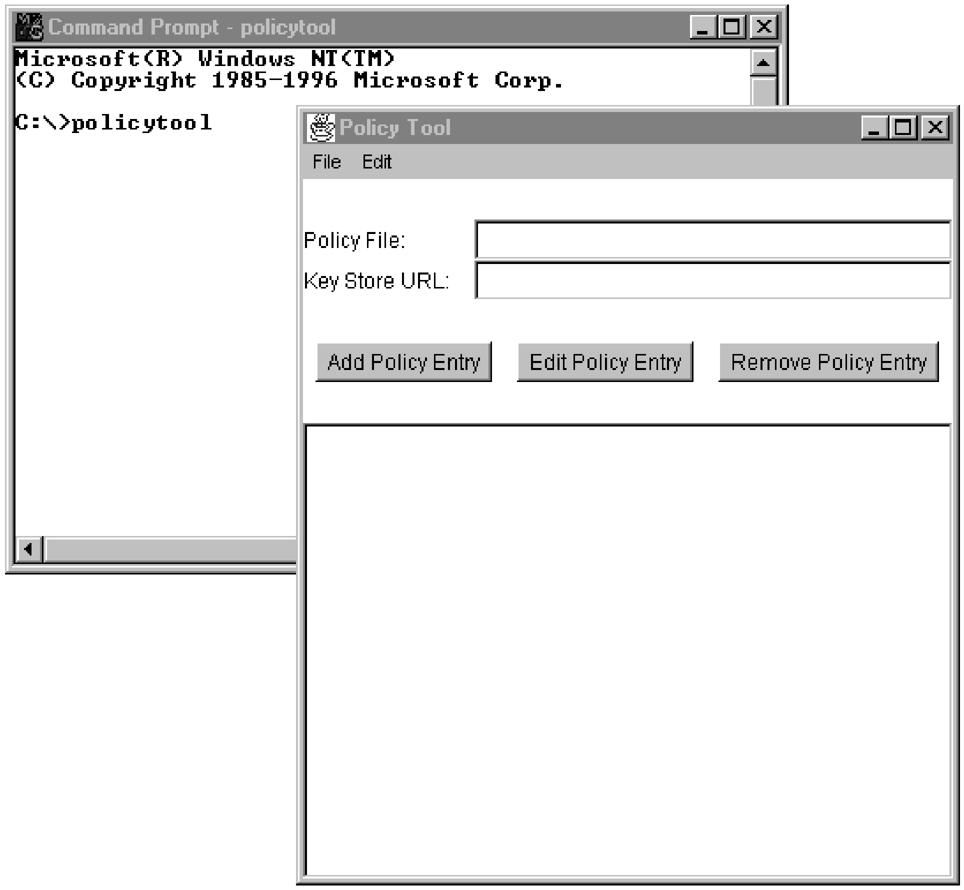
Figure 18. Inside the Signed JAR File

The other file you see, MANIFEST.MF, is a standard file that is always present with a JAR file, even in JDK 1.1. It has nothing to do with signatures. This manifest file contains information similar to the SF file. However, SF files are used particularly by `jarsigner`, and MF files are used by the `jar` utility. You could use the `jar` command with the `xf` option against any JAR file (JDK 1.1 or 1.2) and you would see it containing information about the files that the JAR contains.

Note here that using the `xf` option blows up the contents of the JAR file, and the two original text files used to create the JAR would be overwritten. You could move the JAR file to some other directory, or delete these constituent text files from the working directory and try out the same thing.

2.5.4 The Policy Tool

The Policy Tool is a GUI that can be used for generating, editing, importing and exporting a security policy. This tool can be invoked from the Command Line, by giving the command `policytool`.



```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>policytool
```

Figure 19. Launching the Policy Tool with the policytool Command

It is recommended that all changes to the security policy file during a real life situation be made through this. Apart from the fact that editing the policy file manually might cause inadvertent mistakes, which might cause havoc with granted permissions, there is yet another reason that makes using this tool a good practice. The policy file is a very sensitive document - in that any changes to it could compromise the security of the whole system. Hence, in future versions of JDK, the policy file might be encrypted, or stored in forms other than flat files. Hence, it is a good idea to stick with the GUI that takes care of the details of how exactly the policy file is stored.

Useful as it might seem, the Policy Tool implementation in the beta 3 version seemed to have many bugs. Most of the text boxes we were supposed to be entering data into had the same background and foreground color. Moreover, it threw several NullPointerException objects for what we were sure were legitimate actions, like adding permissions. So, notwithstanding all the recommendations, and the advantages of using the Policy Tool, we were forced to hand edit the policy file. Unless there have been major improvements in this tool in the version that you have, we'd recommend you do the same - but with caution.

However, the Policy Tool is just a basic utility. The JDK is a toolkit and lisencees to the JDK are expected to enhance the tools.

2.6 Access Restriction to Local Code with JDK 1.2

Let's start by adding the following code to our java.policy file:

```
grant codeBase "file:/D:/tmp/julie/" {
  permission java.io.FilePermission "D:\\tmp\\narry\\file1", "read,write";
  permission java.io.FilePermission "D:\\tmp\\narry\\newdir", "write";
};
```

Figure 20. Add This in Your java.policy File

What you have just said in this declaration is that you wish to grant the mentioned permissions to any Java code (meaning, to any Java class file) residing at the base file:/D:/tmp/julie/, irrespective of whether it is signed or not, or who it is signed by. Notice that we have simply removed the directive `signedBy`, according with what we said in 2.4.2.2, “java.policy” on page 26, since we want AccessController not to care whether the code is signed or not nor who signed the code.

Moreover notice that `file:/` is the protocol for finding the location. The protocol could either be `file:/` or `http://`.

In the beta 3 of JDK 1.2 a `*` is assumed after the final / in `file:/D:/tmp/julie/`. Hence, this directive really means:

```
grant codeBase file:/D:/tmp/julie/*
```

Similarly, if your directive is:

```
grant codeBase http://www.ibm.com
```

you are really meaning:

```
grant codeBase http://www.ibm.com/*
```

or, in other words, you mean http://www.ibm.com and everything below and there is no way to limit that. For this reason the Web server administrator should be very careful when they set up the Web server. However, this problem was then fixed in a later beta of JDK 1.2, so now when you specify a codeBase such as `file:/D:/tmp/julie/` or `http://www.ibm.com`, it matches code only in that directory, *not* in the directories below.

Let's now examine the permissions themselves. We are granting the permission java.io.FilePermission with the two string parameters, `D:\\tmp\\narry\\file1` and `read,write`. These are the same parameters that will be given at the time of instantiating the permission by a program, which we will show soon.

The directory structure for this example is as follows:

- The directory D:\tmp\julie has all the code - the Java source code and the class files.
- The directory D:\tmp\narry has the file that we will attempt to read.

We used the default java.security file in this example.

Let's now try writing a simple program that attempts to read the file.

```
import java.io.*;
public class ReadFile
{
  public static void main(String aa[])
  {
    System.out.println("Now in the function, will attempt to read file");
    byte b;

    try
    {
      File myFile = new File("D:\\tmp\\narry\\file1");
      FileInputStream myStream = new FileInputStream(myFile);
      b = (byte)myStream.read();
    }

    catch(Exception e)
    {
      System.out.println("There was an exception, the particulars are " + e.toString());
    }

    System.out.println("Exiting the function");
  }
}
```

Figure 21. The Code for ReadFile.java

You can then try compiling the file, entering as usual:

```
javac ReadFile.java
```

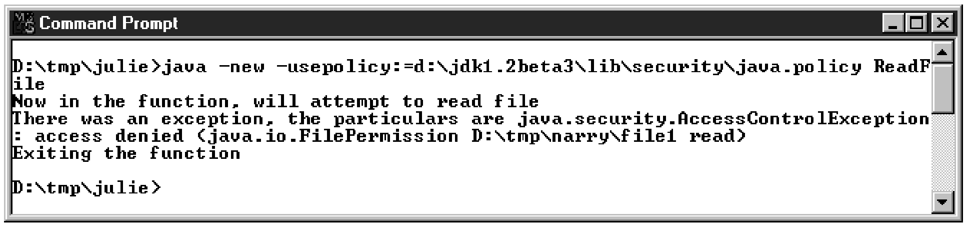
and then you can verify that it runs pretty well with:

```
java ReadFile
```

Until now, you have not invoked the security policy yet. You will be invoking the security policy when you enter:

```
java -new -usepolicy:=${java.home}\lib\security\java.policy ReadFile
```

Just to try it out, we deleted the entries that we shown in Figure 20 on page 36 and ran it. This is what we got at the Command Prompt.



```
D:\tmp\julie>java -new -usepolicy:=d:\jdk1.2beta3\lib\security\java.policy ReadF
ile
Now in the function, will attempt to read file
There was an exception, the particulars are java.security.AccessControlException
: access denied (java.io.FilePermission D:\tmp\narry\file1 read)
Exiting the function

D:\tmp\julie>
```

Figure 22. Running Using the Policy

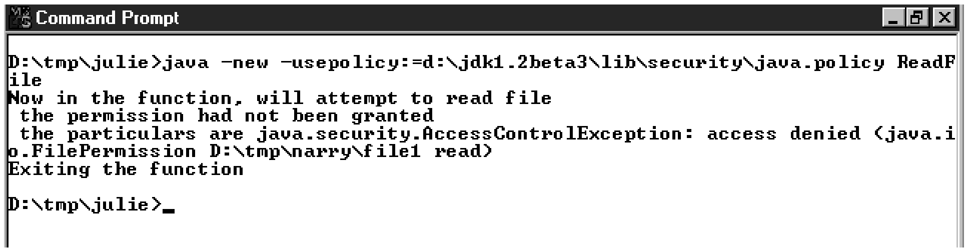
The `-new` option has to be used along with the `-usepolicy` option, and it permits you also to run a program directly out of a JAR file.

Then we tried putting back the directive in the policy file as suggested by Figure 20 on page 36 and the code was able to read the file. So now it's shown that it is possible to restrict access to local code also. Let's now see how to write Java code, to incorporate the new security features.

```
import java.io.*;
import java.security.*;
import java.lang.*;

public class ReadFile
{
  public static void main(String aa[])
  {
    System.out.println("Now in the function, will attempt to read file");

    byte b;
    Permission p;
    SecurityManager sm = System.getSecurityManager();
    boolean canRead = true;

    try
    {
      File myFile = new File("D:\\tmp\\narry\\file1");
      p = (Permission)new FilePermission("D:\\tmp\\narry\\file1", "read");

      try
      {
        sm.checkPermission(p);
      }

      catch(Exception e)
      {
        System.out.println(" the permission had not been granted");
        System.out.println(" the particulars are " + e.toString());
        canRead = false;
      }

      if (canRead)
      {
        FileInputStream myStream = new FileInputStream(myFile);
        b = (byte)myStream.read();
        System.out.println(" read from the file");
      }

    }

    catch(Exception e)
    {
      System.out.println("The exception particulars are " + e.toString());
    }

    System.out.println("Exiting the function");
  }
}
```

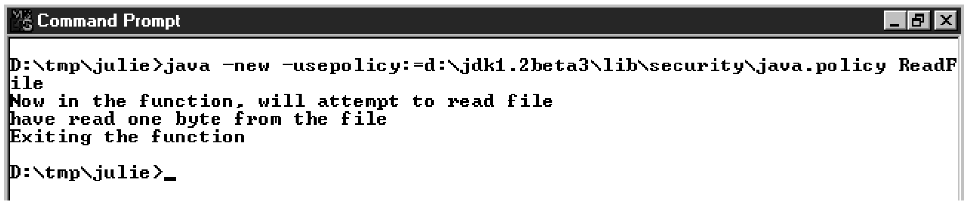
Figure 23. Modified Version of ReadFile.java

Now, notice how we have instantiated a Permission, giving the required values as parameters. The outputs are as shown in the two following figures:



```
D:\tmp\julie>java -new -usepolicy:=d:\jdk1.2beta3\lib\security\java.policy ReadF
ile
Now in the function, will attempt to read file
 the permission had not been granted
 the particulars are java.security.AccessControlException: access denied (java.i
o.FilePermission D:\tmp\narry\file1 read)
Exiting the function

D:\tmp\julie>_
```

Figure 24. Running the Code Without Granting Permissions



```
D:\tmp\julie>java -new -usepolicy:=d:\jdk1.2beta3\lib\security\java.policy ReadF
ile
Now in the function, will attempt to read file
have read one byte from the file
Exiting the function

D:\tmp\julie>_
```

Figure 25. Running the Code After Granting Permissions

Now, let's examine the code.

We defined a Permission p, and instantiated this Permission with the parameters matching the permission we require to do the actual job of the program - to read one byte from the file. We got the SecurityManager from the System class, and checked if this Permission we just instantiated had been granted.

If it was granted, we proceeded to perform the action (reading a byte from the file), else, we printed to the screen the reason why we are not able to proceed, and exited the program. Note here, that we did not stop because an exception was thrown *while* the required operation was being performed. We found out beforehand if an operation is permitted at all or not, and attempted the operation *only* if it was permitted.

This facility can be very useful if you are performing a series of operations that you, as a programmer, suspect might be restricted on the final installation machine. You might want to check all the permissions, and ascertain that all the required permissions for the series of actions that you will be performing are granted before beginning any of them. This would avoid an operation being aborted half way, because one of the operations is forbidden. This will be especially valuable if performing only some of the operations is undesirable, and you wish that either all the operations be performed, or none.

There is another way you can check if the permission has been granted or not. However, it relies on the target object's test methods, and might not be always available. The same old ReadFile program can be modified into this. If you see the API documentation of the class java.io.File, you will see that there are methods such as canRead() and canWrite(). These methods are expected to be used before actually attempting to read or write into the file, to see if the operating system permits the action or not. These methods return a boolean true if the action is permitted, and a boolean false if it is not, and throw an exception if the action is forbidden by the Java SecurityManager. This feature can be utilized, though not regularly as a programming practice, to check if the action can be performed or not.

```
import java.io.*;
import java.security.*;
import java.lang.*;

public class ReadFile
{
  public static void main(String aa[])
  {
    System.out.println("Now in the function, will attempt to read file");
    byte b;

    try
    {
      File myFile = new File("D:\\tmp\\narry\\file1");

      try
      {
        if (myFile.canRead())
        {
          FileInputStream myStream = new FileInputStream(myFile);
          b = (byte)myStream.read();
          System.out.println("have read one byte from the file");
        }
      }

      catch(Exception e)
      {
        System.out.println(" the permission had not been granted");
        System.out.println(" the particulars are " + e.toString());
      }
    }

    catch(Exception e)
    {
      System.out.println("There was an exception, the particulars are " + e.toString());
    }

    System.out.println("Exiting the function");
  }
}
```

Figure 26. ReadFile.java, Using the canRead() Method of the File Class

This program when run without and with the permissions, gave the type of messages as indicated before, in Figure 24 on page 40 and Figure 25 on page 40. Though this technique may seem pretty much simpler, we recommend the former one, since all target objects might not have this feature, and also because this is really an indirect way of checking for granted permissions.

2.6.1 A Complete Example

You could use the following sample code to complete the example.

```
import java.io.*;
import java.security.*;


public class PermTest
{

  private void checkPerm(SecurityManager sm, Permission p)
  {

    Exception exc = null;

    try
    {
      sm.checkPermission(p);
    }

    catch(Exception e)
    {
      exc = e;
    }

    finally
    {
      System.out.println(p.toString());

      if (exc == null)
        System.out.println("permission GRANTED");
      else
        System.out.println("permission DENIED");
    }
  }

  public static void main(String[] args)
  {
    PermTest pt = new PermTest();
    Permission p = null;
    String filename = "C:\\jdk1.2\\play\\file1";
    String dirname = "C:\\jdk1.2\\play\\newdir";

    //   Get the default security manager.  We have one because we invoked
    //   the java interpreter with the "-new" and "-usepolicy" flags.

    SecurityManager sm = System.getSecurityManager();

    if (sm == null)
      System.out.println("NULL security manager!");
    else
    {

      //   Here are some permission queries that go directly to the
      //   SecurityManager's checkPermission() method, via the local
      //   checkPerm() method.
```

Figure 27 (Part 1 of 4). PermTest.java, a Long Example Code

```
      p = (Permission) new FilePermission(filename, "read");
      pt.checkPerm(sm, p);
      p = (Permission) new FilePermission(filename, "write");
      pt.checkPerm(sm, p);
      p = (Permission) new FilePermission(filename, "delete");
      pt.checkPerm(sm, p);
      p = (Permission) new FilePermission(filename, "execute");
      pt.checkPerm(sm, p);
      p = (Permission) new FilePermission(filename, "read,write,execute,delete");
      pt.checkPerm(sm, p);

      File f = new File(filename);

      if (f == null)
        System.out.println("NULL file!");
      else
      {

      //   Here are some permission queries that are a little more
      //   indirect, relying on the target object's test methods.

        Exception exc = null;

        //   Test the read permission.

        try
        {
          f.canRead();
        }

        catch (Exception e)
        {
          exc = e;
        }

        if (exc == null)
          System.out.println(filename + " read permission GRANTED");
        else
        {
          exc = null;
          System.out.println(filename + " read permission DENIED");
        }

        //   Test the write permission.

        try
        {
          f.canWrite();
        }

        catch (Exception e)
        {
          exc = e;
        }
```

Figure 27 (Part 2 of 4). PermTest.java, a Long Example Code

```
        if (exc == null)
          System.out.println(filename + " write permission GRANTED");
        else
        {
          exc = null;
          System.out.println(filename + " write permission DENIED");
        }

        //   Here are some operations that will throw an exception if
        //   the required permission(s) are not met.

        //   java.io.File.exists() requires the read permission.

        try
        {
          f.exists();
        }

        catch (Exception e)
        {
          exc = e;
        }

        if (exc == null)
          System.out.println(filename + " read permission GRANTED");
        else
        {
          exc = null;
          System.out.println(filename + " read permission DENIED");
        }

        File d = new File(dirname);

        //   java.io.File.mkdir() requires the write permission.

        try
        {
          d.mkdir();
        }

        catch(Exception e)
        {
          exc = e;
        }

        if (exc == null)
          System.out.println(dirname + " write permission GRANTED");
        else
        {
          exc = null;
          System.out.println(dirname + " write permission DENIED");
        }

        //   java.io.File.delete() requires the delete permission.
```

Figure 27 (Part 3 of 4). PermTest.java, a Long Example Code

```
         try
         {
           f.delete();
         }

         catch(Exception e)
         {
           exc = e;
         }

         if (exc == null)
           System.out.println(filename + " delete permission GRANTED");
         else
         {
           exc = null;
           System.out.println(filename + " delete permission DENIED");
         }
       }
     }
   }
 }
```

Figure 27 (Part 4 of 4). PermTest.java, a Long Example Code

This example performs some basic permission checks using the java.io.FilePermission class. The first group of checks directly calls the java.security.SecurityManager.checkPermission() method with a specific permission. The second group of checks calls various java.io.File methods. These methods eventually end up calling the SecurityManager's checkPermission() method. In all cases, if the permission is not allowed, an exception is thrown and caught and the result is a `permission DENIED` message. If the permission is allowed, a `permission GRANTED` message is displayed.

This code was placed in C:\jdk1.2\play. A temporary file C:\jdk1.2\play\file1 and a temporary directory C:\jdk1.2\play\newdir were created, in accordance with what is indicated in the source code. The only permissions needed to get all the access checks to be granted are:

```
java.io.FilePermission "C:\\jdk1.2\\play\\file1", "read,write,execute,delete";
```

and

```
java.io.FilePermission "C:\\jdk1.2\\play\\newdir", "write";
```

Note that if you change the filename or dirname variables within the testcase, you must change the permission targets accordingly. Also note that all the access checks will be granted if more general permissions are granted to the codeSource in question.

Here is the policy file we used:

```
grant codeBase "file:/C:/jdk1.2/play" {
  permission java.io.FilePermission "C:\\jdk1.2\\play\\file1", "read,write,delete,execute";
  permission java.io.FilePermission "C:\\jdk1.2\\play\\newdir", "write"
};
```

Figure 28. Policy File for TestPerm.class

codeBase URL

We noticed that in the beta 3 and 4 releases of JDK 1.2 for Windows NT the value for the codeBase URL is case-sensitive. Even the drive letter creates problems if not capitalized. So for example `"file:/C:/jdk1.2/play"` would be considered different from `"file:/C:/JDK1.2/play"`. In particular `"file:/c:/jdk1.2/play"` wouldn't be accepted, because the proper drive letter is C and not c. However, this restriction didn't apply to the permission fields.

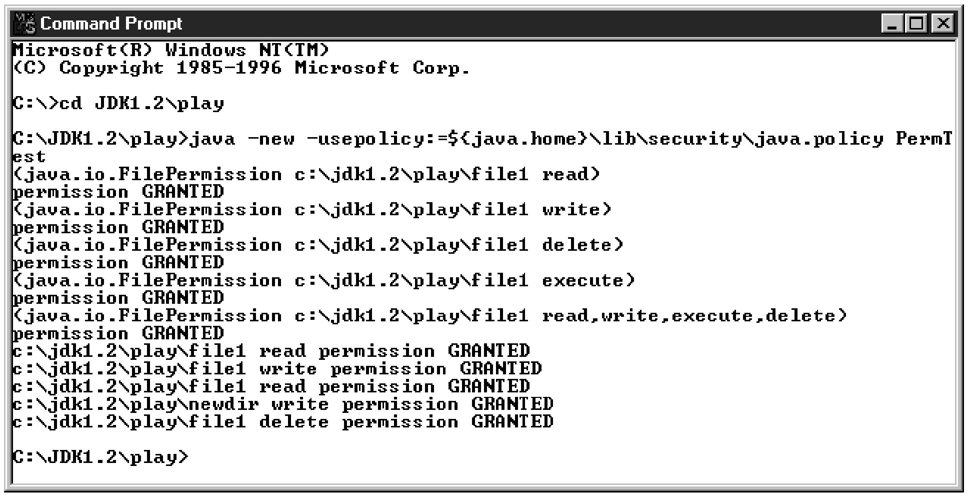
You can compile the above code by issuing, as usual:

```
javac PermTest.java
```

and then you can run it with the command:

```
java -new -usepolicy:=${java.home}\lib\security\java.policy PermTest
```

When you run PermTest with all the permissions, you should get a series of `permission GRANTED` messages, as shown.



```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>cd JDK1.2\play

C:\JDK1.2\play>java -new -usepolicy:=${java.home}\lib\security\java.policy PermT
est
(java.io.FilePermission c:\jdk1.2\play\file1 read)
permission GRANTED
(java.io.FilePermission c:\jdk1.2\play\file1 write)
permission GRANTED
(java.io.FilePermission c:\jdk1.2\play\file1 delete)
permission GRANTED
(java.io.FilePermission c:\jdk1.2\play\file1 execute)
permission GRANTED
(java.io.FilePermission c:\jdk1.2\play\file1 read,write,execute,delete)
permission GRANTED
c:\jdk1.2\play\file1 read permission GRANTED
c:\jdk1.2\play\file1 write permission GRANTED
c:\jdk1.2\play\file1 read permission GRANTED
c:\jdk1.2\play\newdir write permission GRANTED
c:\jdk1.2\play\file1 delete permission GRANTED

C:\JDK1.2\play>
```

Figure 29. Running PermTest With All the Permissions

If you remove some permissions from the policy file, you will see that the corresponding messages in the output window change to read `permission DENIED`.

If you wish for more information, you can add:

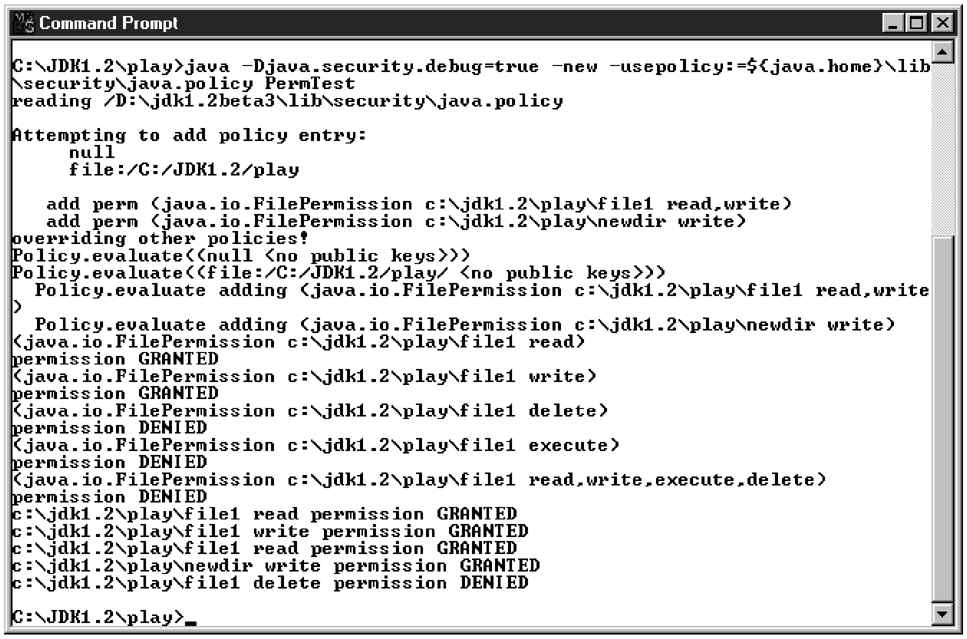
```
throws SecurityException
```

to the main() method and remove a denied permission from its `try{}` block. The exception stack will be displayed to stderr. You can also get some info by running the Java interpreter `java` with the `-Djava.security.debug=true` flag.

For example, if you the remove `delete` and `execute` from the policy file, it will look like the following screen:

```
grant codeBase "file:/C:/jdk1.2/play" {
  permission java.io.FilePermission "c:\\jdk1.2\\play\\file1", "read,write";
  permission java.io.FilePermission "c:\\jdk1.2\\play\\newdir", "write"
};
```

Then you can run the PermTest application with the `-Djava.security.debug=true` flag. And this is what you will see in the Command Prompt window:



```
C:\JDK1.2\play>java -Djava.security.debug=true -new -usepolicy:=${java.home}\lib
\security\java.policy PermTest
reading /D:\jdk1.2beta3\lib\security\java.policy

Attempting to add policy entry:
     null
     file:/C:/JDK1.2/play

   add perm (java.io.FilePermission c:\jdk1.2\play\file1 read,write)
   add perm (java.io.FilePermission c:\jdk1.2\play\newdir write)
overriding other policies!
Policy.evaluate((null <no public keys>))
Policy.evaluate((file:/C:/JDK1.2/play/ <no public keys>))
  Policy.evaluate adding (java.io.FilePermission c:\jdk1.2\play\file1 read,write
)
  Policy.evaluate adding (java.io.FilePermission c:\jdk1.2\play\newdir write)
(java.io.FilePermission c:\jdk1.2\play\file1 read)
permission GRANTED
(java.io.FilePermission c:\jdk1.2\play\file1 write)
permission GRANTED
(java.io.FilePermission c:\jdk1.2\play\file1 delete)
permission DENIED
(java.io.FilePermission c:\jdk1.2\play\file1 execute)
permission DENIED
(java.io.FilePermission c:\jdk1.2\play\file1 read,write,execute,delete)
permission DENIED
c:\jdk1.2\play\file1 read permission GRANTED
c:\jdk1.2\play\file1 write permission GRANTED
c:\jdk1.2\play\file1 read permission GRANTED
c:\jdk1.2\play\newdir write permission GRANTED
c:\jdk1.2\play\file1 delete permission DENIED

C:\JDK1.2\play>_
```

Figure 30. Running PermTest Removing Permissions and Using the Debugging Flag

2.6.2 An AWT Test

Let's now consider another example involving the Java Abstract Windowing Toolkit (AWT), or the java.awt packages. Here, we are trying to get access to a java.awt.PrintJob object using the java.awt.Toolkit.getPrintJob() method. The program works if the following permission is set in the java.policy file:

```
permission java.lang.RuntimePermission "print.queueJob";
```

If you have not already done so, add that permission into your java.policy file (it might also be the only permission in the policy file) and key in the following program:

```
import java.awt.*;
import java.awt.event.*;

class GetPrintJob extends Frame implements ActionListener
{
  boolean p = true;

  GetPrintJob()
  {
    super("Toolkit.getPrintJob() test case");
    setSize(300, 100);
    setLocation(200, 200);
    Button b = new Button("getPrintJob");
    add(b, BorderLayout.CENTER);
    b.addActionListener(this);

    show();
  }

  public void actionPerformed(ActionEvent evt)
  {
    try
    {
      Toolkit.getDefaultToolkit().getPrintJob(null, "PrintJob", null);
    }

    catch(Exception e)
    {
      System.out.println(" there was an exception, "+ e.toString());
      p=false;
    }
      if (p)
        System.out.println("No exception. Test is successful.");
  }

  public static void main(String[] args)
  {
    new GetPrintJob();
  }
}
```

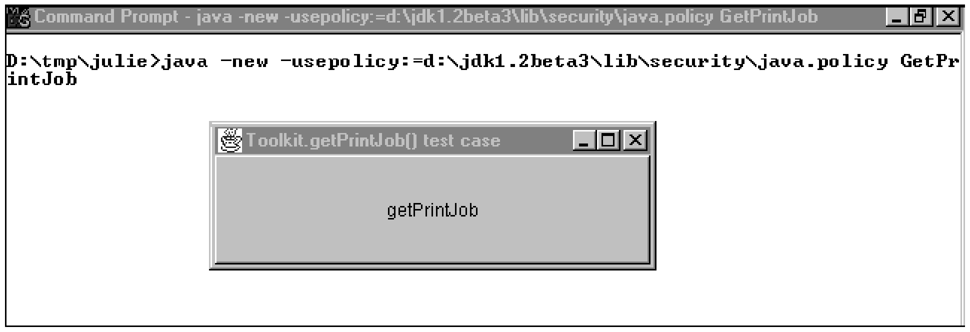
Figure 31. The Print Job Program

As you can see, the main() method simply invokes the constructor GetPrintJob(). This constructor, in turn, calls the constructor of the superclass java.awt.Frame, passing the String

```
Toolkit.getPrintJob() test case
```

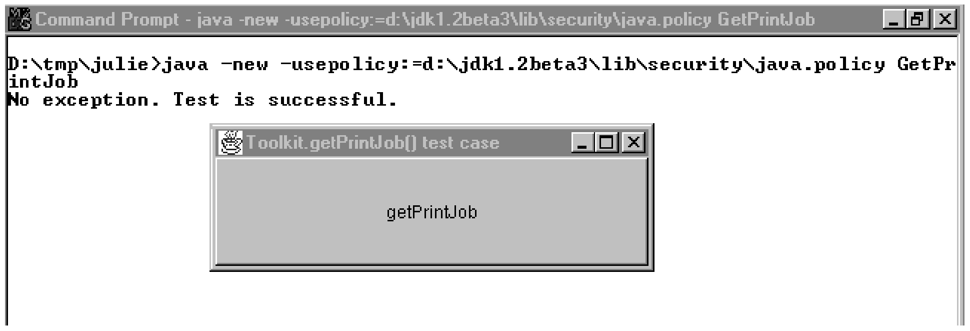
as parameter, that will be the title of the newly created Frame object.

This program does little else than display a graphical button and get the java.awt.PrintJob object when you click the button **getPrintJob**. Note that once this object is obtained, you can have access to the Graphics object using the getGraphics() method, which renders for an appropriate print device. Note that we have not tried instantiating permissions. We are just catching and printing the exception. You might want to try without the `try{}catch(){}` block too, to see what output it gives. Here's the output with the `try{}catch(){}` block.


Command Prompt - java -new -usepolicy:=d:\jdk1.2beta3\lib\security\java.policy GetPrintJob
Toolkit.getPrintJob() test case
getPrintJob

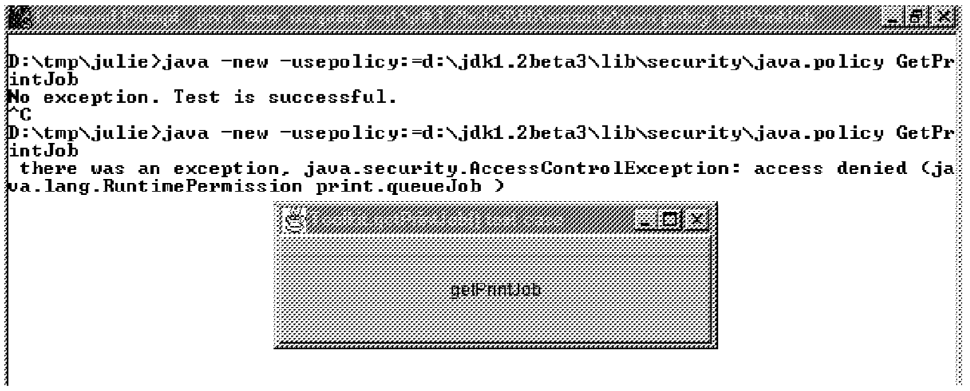
```
D:\tmp\julie>java -new -usepolicy:=d:\jdk1.2beta3\lib\security\java.policy GetPr
intJob
```

Figure 32. Running the GetPrintJob Program


Command Prompt - java -new -usepolicy:=d:\jdk1.2beta3\lib\security\java.policy GetPrintJob
Toolkit.getPrintJob() test case
getPrintJob

```
D:\tmp\julie>java -new -usepolicy:=d:\jdk1.2beta3\lib\security\java.policy GetPr
intJob
No exception. Test is successful.
```

Figure 33. Clicking on the Button - With Proper Permissions



```
D:\tmp\julie>java -new -usepolicy:=d:\jdk1.2beta3\lib\security\java.policy GetPr
intJob
No exception. Test is successful.
^C
D:\tmp\julie>java -new -usepolicy:=d:\jdk1.2beta3\lib\security\java.policy GetPr
intJob
 there was an exception, java.security.AccessControlException: access denied (ja
va.lang.RuntimePermission print.queueJob )
```

Figure 34. Clicking on the Button - Without the Permission

Now, here is a slightly more sophisticated way of doing the same thing. You have seen a similar technique before in Figure 23 on page 39.

```
import java.awt.*;
import java.awt.event.*;
import java.security.*;
import java.lang.*;

class GetPrintJob extends Frame implements ActionListener
{
  boolean p = true;

  GetPrintJob()
  {
    super("Toolkit.getPrintJob() test case");
    setSize(300, 100);
    setLocation(200, 200);
    Button b = new Button("getPrintJob");
    add(b, BorderLayout.CENTER);
    b.addActionListener(this);
    show();
  }

  public void actionPerformed(ActionEvent evt)
  {
    Permission perm;
    SecurityManager sm = System.getSecurityManager();
    perm = (Permission) new RuntimePermission("print.queueJob");

    try
    {
      sm.checkPermission(perm);
    }

    catch(Exception e)
    {
      System.out.println(" Permission does not exist");
      System.out.println(" Particulars :" + e.toString());
      p = false;
    }

  if (p)
  {
    Toolkit.getDefaultToolkit().getPrintJob(null, "PrintJob", null);
    System.out.println("No exception. Test is successful.");
  }
}

  public static void main(String[] args)
  {
    new GetPrintJob();
  }
}
```

Figure 35. Modified Version of GetPrintJob.java - Checking the Permissions

This would get you largely similar results. We believe this is probably a more systematic method of programming, and it might prove to be beneficial in the long run to make it a practice to check for the permissions before performing any action.

2.7 Applets in the New JDK 1.2 Security Model

In this last example, we compare signed and unsigned applets.

The applet we wrote loads a pretty screen and, on clicking on button **A**, attempts to open a remote socket. Here's the source code for the applet:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;

public class MyApplet extends Applet
{
  Button button[] = new Button[3];
  boolean p = true;

  public void init()
  {
    setBackground(Color.white);

    button[0] = new Button("A");
    button[1] = new Button("B");
    button[2] = new Button("C");

    add(button[0]);
    add(button[1]);
    add(button[2]);
  }

  public boolean action(Event evt, Object obj)
  {

    if (evt.target == button[0])
    {
      try
      {
        Socket n = new Socket("marcop.austin.ibm.com", 44444);
      }

      catch(Exception e)
      {
        System.out.println(" error -- exception " + e.toString());
        e.printStackTrace();
        showStatus("there was an exception " + e.toString());
        p = false;
      }

      if (p)
        showStatus(" Button A was pushed ");

      p = true;
      return true;
    }

    if (evt.target == button[1])
    {
      showStatus(" Button B was pushed ");
      return true;
    }
```

Figure 36 (Part 1 of 2). MyApplet.java, the Applet that Attempts the Socket Operation

```
    if (evt.target == button[2])
    {
      showStatus(" Button C was pushed ");
      return true;
    }

    return false;
  }
}
```

Figure 36 (Part 2 of 2). MyApplet.java, the Applet that Attempts the Socket Operation

We compiled the above source entering the command:

```
javac MyApplet.java
```

In the next two sections we will use this example to show how unsigned and signed applets are managed in the new Java 1.2 security model.

2.7.1 Unsigned Applets

First we examine the case of an unsigned applet. We wrote a simple HTML page, named myPage.html, whose code is as shown:

```
<HTML>
  <APPLET CODE=MyApplet WIDTH=200 HEIGHT=100>
  </APPLET>
</HTML>
```

Figure 37. The HTML Page Containing the Applet - myPage.html

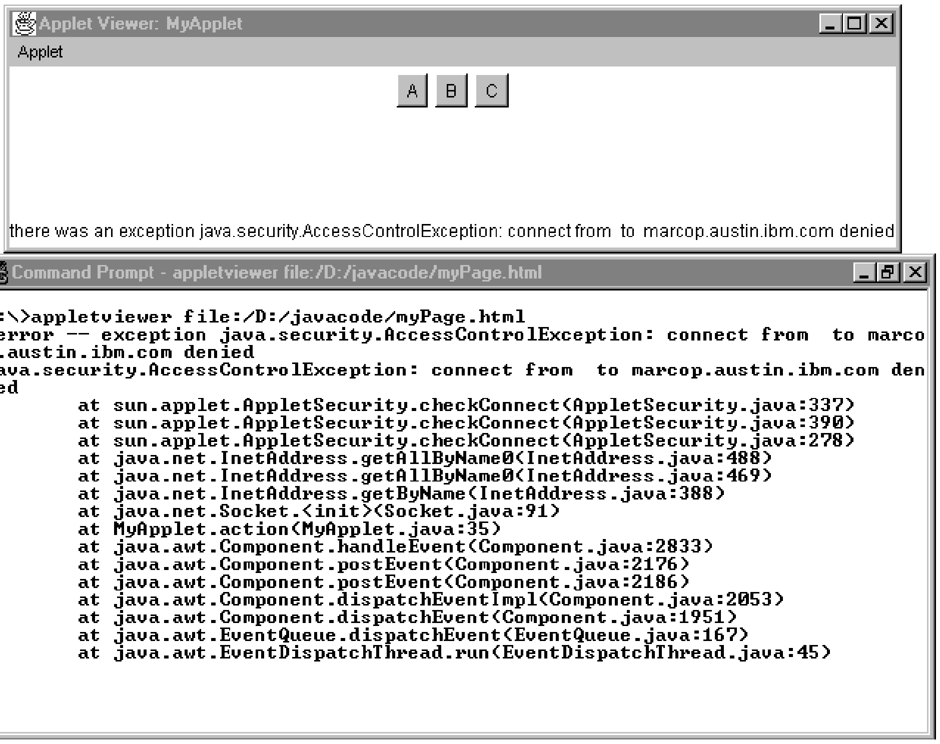
Notice that this HTML page invokes the class file of the applet MyApplet, which we have shown in Figure 36 on page 53.

At the time of this project, as we already said several times, JDK 1.2 was available only in beta, so no Web browser supported the 1.2 version yet. However, even if we didn't have the opportunity to test an applet using any real Java 1.2 enabled Web browser, we could efficiently use the JDK 1.2 Applet Viewer, which from a security point of view offered us the possibility to reach the same conclusions as a regular Java 1.2 browser. As you certainly know, the Applet Viewer is a tool shipped with the JDK that is used to run applets, commonly for testing purposes. It is launched on the command line with the command `appletviewer` followed by the name of the HTML file invoking the applet. You can also specify the URL to the HTML file. That command parses the HTML file and considers only everything between the two tags `<APPLET>` and `</APPLET>` plus the tags themselves. Everything else outside the two tags is completely ignored. This explains also why we wrote such a simple HTML file.

Notice that we had *not* given in the java.policy file *any* permissions to *anyone* to open *any* socket. When we tried to run the applet locally, with the command

```
appletviewer file:/D:/javacode/myPage.html
```

and we clicked on the button **A**, here's what we got:



```
there was an exception java.security.AccessControlException: connect from  to marcop.austin.ibm.com denied
```

```
D:\>appletviewer file:/D:/javacode/myPage.html
 error -- exception java.security.AccessControlException: connect from  to marco
p.austin.ibm.com denied
java.security.AccessControlException: connect from  to marcop.austin.ibm.com den
ied
        at sun.applet.AppletSecurity.checkConnect(AppletSecurity.java:337)
        at sun.applet.AppletSecurity.checkConnect(AppletSecurity.java:390)
        at sun.applet.AppletSecurity.checkConnect(AppletSecurity.java:278)
        at java.net.InetAddress.getAllByName0(InetAddress.java:488)
        at java.net.InetAddress.getAllByName0(InetAddress.java:469)
        at java.net.InetAddress.getByName(InetAddress.java:388)
        at java.net.Socket.<init>(Socket.java:91)
        at MyApplet.action(MyApplet.java:35)
        at java.awt.Component.handleEvent(Component.java:2833)
        at java.awt.Component.postEvent(Component.java:2176)
        at java.awt.Component.postEvent(Component.java:2186)
        at java.awt.Component.dispatchEventImpl(Component.java:2053)
        at java.awt.Component.dispatchEvent(Component.java:1951)
        at java.awt.EventQueue.dispatchEvent(EventQueue.java:167)
        at java.awt.EventDispatchThread.run(EventDispatchThread.java:45)
```

Figure 38. On the Applet Trying to Open a Socket

Note the applet window displayed above the Command Prompt window. It registered that a java.security.AccessControlException had been thrown when the applet had tried to open a socket to a remote host.

We also tried placing the two files myPage.html and MyApplet.class on a Web server, having IP address 9.24.104.51. We stored them in a directory named javacode that we created under the HTML directory of the Web server. Then we accessed the HTML file entering in the client machine the command:

```
appletviewer http://9.24.104.51/javacode/myPage.html
```

And we got the same results.

These two experiments simply demonstrated that our unsigned applet was by default untrusted. We had not granted any permission yet for unsigned code to open any socket and our applet was not allowed to do that, whether it was local or remote.

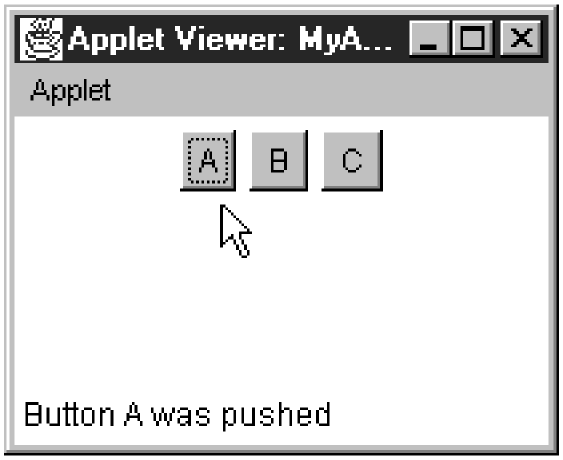
We can now try running this applet after giving appropriate permissions. So, in the java.policy file, we made this entry.

```
grant codeBase "http://9.24.104.51/javacode" {
  permission java.net.SocketPermission "marcop.austin.ibm.com:44444", "connect";
}
```

Figure 39. Giving the Permission to Any Code from the Given URL

Here we have said that any code originating from the given URL is permitted to connect to the particular socket on the particular machine, without any differences between signed and unsigned code (note in fact that we omitted the `signedBy` directive).

We then tried again, with the Applet Viewer, and this time there were no errors.



Button A was pushed

Figure 40. Trying to Open a Socket - With the Proper Permission

Note that, in accordance with the Java source code that we had written, the message:

```
Button A was pushed
```

is displayed only if the socket connection was successful.

We observed that the Applet Viewer in JDK 1.2 beta 3 seems to be caching applets, and changed permissions are not reflected immediately. This might be a bug.

2.7.2 Signed Applets

We can now try another thing. We could say that we will give access to any applets that have been signed by somebody, to do this action. Here's how this can be done.

First, we put MyApplet.class in the JAR file MyApplet.jar using the command:

```
jar cvf MyApplet.jar MyApplet.class
```

Then, we modified the HTML page on the Web server to incorporate the change in the applet source.

```
<HTML>
  <APPLET ARCHIVE=MyApplet.jar CODE=MyApplet WIDTH=200 HEIGHT=100>
  </APPLET>
</HTML>
```

Figure 41. Modified HTML Page myPage.html Invoking the JAR File of the Applet

We then created a new keystore called louKeyStore, using the `keytool` utility, as has been explained in 2.5.2, “The keytool Utility” on page 29. The command we issued was:

```
keytool -genkey
-dname "CN=Marco Pistoia, OU=ITSO, O=IBM, L=Cary, S=North Carolina, C=US"
-alias louKey -keypass sw1504r
-keystore D:\javacode\louKeyStore
-storepass sw1504r -validity 365
```

We remind you that this command should be entered on a single line, even though we have split it for clarity.

What we are trying to do here is to simulate the condition wherein the signer is on a given machine, but the applet will run on a different machine. Ideally, it is as if the applet were signed on one machine and run on another one, but we simulated this situation using two different keystores on the same machine. The first one was our default keystore, C:\WINNT\Profiles\Administrator\.keystore. The second one was the newly created louKeyStore. The technique that we used in this example was to sign the applet using the certificate stored in the louKeyStore, then to trust that signature by exporting the certificate from louKeyStore and importing it in the default keystore as trusted. In this way we perfectly simulated the presence of the two different machines. To really run an applet signed by the certificate stored in louKeyStore, we also need to specify in the java.policy file that the signer is trusted. We are just going to describe all these steps.

First of all, of course, we had to generate a certificate for louKey. We did that using the command:

```
keytool -selfcert -alias louKey -keystore D:\javacode\louKeyStore -storepass sw1504r
```

Then, we used that certificate to sign the JAR file MyApplet.jar containing the applet class MyApplet.class. To do this, we entered:

```
jarsigner -keystore D:\javacode\louKeyStore -storepass sw1504r MyApplet.jar louKey
```

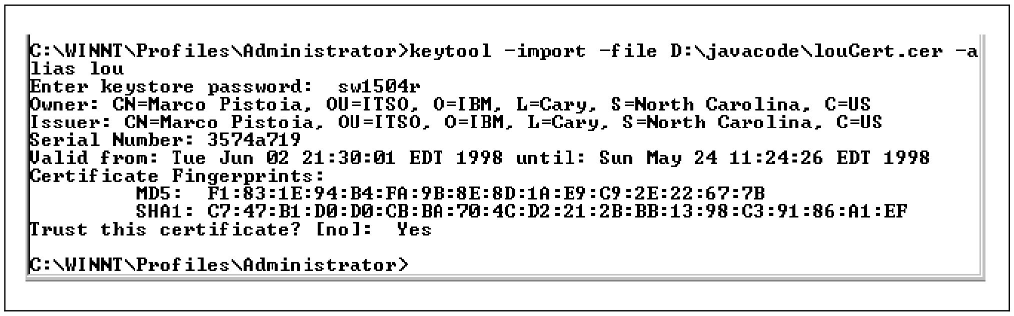
Then, we had to import the new certificate to the default keystore that we had, at C:\WINNT\Profiles\Administrator\.keystore. We did this by first exporting the self-signed certificate using the following command, entered on a single line:

```
keytool -export -keystore D:\javacode\louKeyStore -storepass sw1504r -alias louKey
-file louCert.cer
```

We then imported the certificate as a trusted root in the default keystore, using the command:

```
keytool -import -file D:\javacode\louCert.cer -alias lou
```

This prompted the certificate, and asked us if we trusted it. As you can see in the following screen, we entered `Yes`.



```
C:\WINNT\Profiles\Administrator>keytool -import -file D:\javacode\louCert.cer -a
lias lou
Enter keystore password:  sw1504r
Owner: CN=Marco Pistoia, OU=ITSO, O=IBM, L=Cary, S=North Carolina, C=US
Issuer: CN=Marco Pistoia, OU=ITSO, O=IBM, L=Cary, S=North Carolina, C=US
Serial Number: 3574a719
Valid from: Tue Jun 02 21:30:01 EDT 1998 until: Sun May 24 11:24:26 EDT 1998
Certificate Fingerprints:
         MD5:  F1:83:1E:94:B4:FA:9B:8E:8D:1A:E9:C9:2E:22:67:7B
         SHA1: C7:47:B1:D0:D0:CB:BA:70:4C:D2:21:2B:BB:13:98:C3:91:86:A1:EF
Trust this certificate? [no]:  Yes

C:\WINNT\Profiles\Administrator>
```

Figure 42. Importing a Trusted Certificate, Giving an Alias to the Signer

There is one last step. We had to enter the alias as trusted in the java.policy file. So we added the following entry in the policy file, and deleted all the other permissions.

```
grant signedBy "lou" {
  permission java.net.SocketPermission "marcop.austin.ibm.com:44444", "connect";
};
```

The above entry simply means that we intend to trust all the code signed by the signer whose alias is lou, wherever that code comes from (note in fact that the `codeBase` directive is omitted). And on running the applet from the Applet Viewer, it worked fine, as shown in Figure 40 on page 56.

We also tried giving the following entry in the policy file:

```
grant codeBase "http://9.24.104.51/javacode", signedBy "lou" {
  permission java.net.SocketPermission "marcop.austin.ibm.com:44444", "connect";
};
```

The above entry, in the policy file, means that we intend to trust all the Java code signed by the signer whose alias is lou. However, this time we don't actually trust all the code signed by lou, but only that code coming from the directory HTML\javacode of the Web server having IP address 9.24.104.51, as the value of the `codeBase` directive specifies. We then stored the applet and the HTML page myPage.html in the directory HTML\javacode of the Web server and when we pointed to the applet URL using the Applet Viewer, the applet worked just fine again, as shown in Figure 40 on page 56.

Note that the machine with which the applet establishes a socket connection is not the Web server that served the applet itself.

codeBase and Signer in a Directive

What we really did here was to identify the code using both codeBase and signature. Note that in general permissions are additive - in the sense that if you give permission A to `codeBase XX` and permission B to `signedBy YY`, and use these in two separate `grant` directives, any applet originating from XX and being signed by YY will have both the permissions A and B. We will demonstrate just this in 2.8, “Additive Permissions” on page 59.

However, in the above condition, we are using the codeBase and the signature to identify completely a set of applets that are permitted to do a certain set of operations. Hence, in this case, permissions are not additive. Both the codeBase and the signature together act as identifiers.

The Keystore URL Entry in the Policy File

Note that in the above example we have imported the signer's certificate into the default keystore C:\WINNT\Profiles\Administrator\.keystore. Hence, we did not need to mention the keystore in the policy file either. However, you might want to do the whole thing using some *other* keystore. In such a case, you would have to import the certificate into that particular keystore using the `-keystore` option of the `keytool` command, and then add the following directive in the policy file, specifying the URL of the keystore you intend to use:

```
keystore "file:/location/keyStoreName";
```

In our case, notwithstanding the default status of the keystore, we could have chosen to be more technically correct albeit verbose, and have said in our policy file

```
keystore "file:/C:/WINNT/Profiles/Administrator/.keystore";
```

2.8 Additive Permissions

In this section, we describe you how the permissions work together. Consider a modification of the MyApplet.java code as shown.

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;

public class MyApplet extends Applet
{
  Button button[] = new Button[3];
  boolean p = true;
  boolean p1 = true;
  boolean p2 = true;

  public void init()
  {
    setBackground(Color.white);

    button[0] = new Button("A");
    button[1] = new Button("B");
    button[2] = new Button("C");

    add(button[0]);
    add(button[1]);
    add(button[2]);
  }

  public boolean action(Event evt, Object obj)
  {
    if (evt.target == button[0])
    {
      try
      {
        Socket n = new Socket("marcop.austin.ibm.com", 44444);
      }

      catch (Exception e)
      {
        System.out.println(" error -- exception " + e.toString());
        e.printStackTrace();
        showStatus("there was an exception " + e.toString());
        p = false;
      }

      if (p)
        showStatus(" Button A was pushed ");

      p = true;
      return true;
    }

    if (evt.target == button[1])
    {
      File n = new File("D:\\tmp\\narry\\file1");
```

Figure 43 (Part 1 of 2). Modification of MyApplet.java to Show Additive Permissions

```
      try
      {
        n.canRead();
      }

      catch(Exception ee)
      {
        System.out.println(" error -- exception " + ee.toString());
        ee.printStackTrace();
        showStatus("there was an exception " + ee.toString());
        p1 = false;
      }

      if (p1)
        showStatus(" Button B was pushed ");

      return true;
    }

    if (evt.target == button[2])
    {
      File nn = new File("D:\\tmp\\tony\\larry");
      try
      {
        nn.canRead();
      }

      catch(Exception eee)
      {
        System.out.println(" error -- exception " + eee.toString());
        eee.printStackTrace();
        showStatus("there was an exception " + eee.toString());
        p2 = false;
      }

      if (p2)
        showStatus(" Button C was pushed ");

      return true;
    }

    return false;
  }
}
```

Figure 43 (Part 2 of 2). Modification of MyApplet.java to Show Additive Permissions

When we compiled this code, we entered:

```
java MyApplet.java
```

and we overwrote the existing MyApplet.class file.

Note that three permissions are required here - one SocketPermission permission and two FilePermission permissions. Now we can:

1. Grant the codeBase the SocketPermission.

2. Grant all code signed by the signer lou a FilePermission for the file named file1.
3. Create another signer in a keystore, and grant all the code signed by this new user, a FilePermission for the file named larry.

In 2.7.2, "Signed Applets" on page 56, we showed how to create a signer, and how to import his certificate into a keystore as a trusted certificate. Now you have to repeat similar steps. The following screens were captured *after* the louKey was imported as trusted certificate into the default keystore. We just had to create and import another key signer into the keystore. The following screens will help you follow the steps.

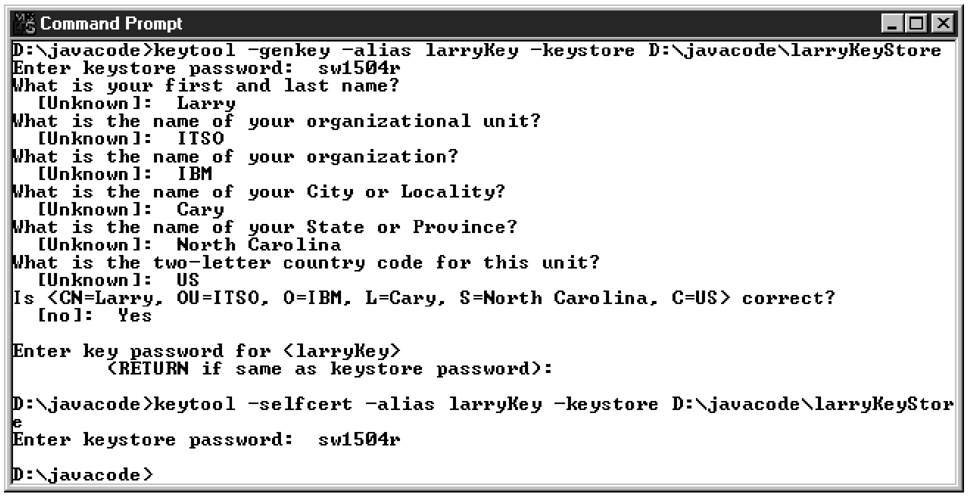
First, create the keystore D:\javacode\larryKeyStore and a self-signed certificate for the key, by entering the command:

```
keytool -genkey -alias larryKey -keystore D:\javacode\larryKeyStore
```

followed by:

```
keytool -selfcert -alias larryKey - keystore D:\javacode\larryKeyStore
```

You can see that in this case it is the `keytool` command itself that prompts you with the options to enter, if you do not enter them directly on the command line.

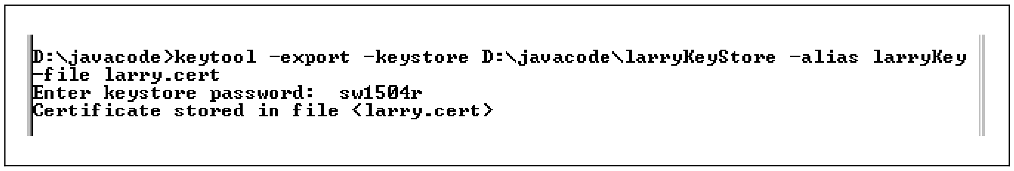


```
D:\javacode>keytool -genkey -alias larryKey -keystore D:\javacode\larryKeyStore
Enter keystore password:  sw1504r
What is your first and last name?
  [Unknown]:  Larry
What is the name of your organizational unit?
  [Unknown]:  ITSO
What is the name of your organization?
  [Unknown]:  IBM
What is the name of your City or Locality?
  [Unknown]:  Cary
What is the name of your State or Province?
  [Unknown]:  North Carolina
What is the two-letter country code for this unit?
  [Unknown]:  US
Is <CN=Larry, OU=ITSO, O=IBM, L=Cary, S=North Carolina, C=US> correct?
  [no]:  Yes

Enter key password for <larryKey>
        <RETURN if same as keystore password>:

D:\javacode>keytool -selfcert -alias larryKey -keystore D:\javacode\larryKeyStor
e
Enter keystore password:  sw1504r

D:\javacode>
```

Figure 44. Create the Keystore with a Key, and Create a Self-Signed Certificate

We exported the certificate, issuing:

```
keytool -export -keystore D:\javacode\larryKeyStore -alias larryKey -file larry.cert
```

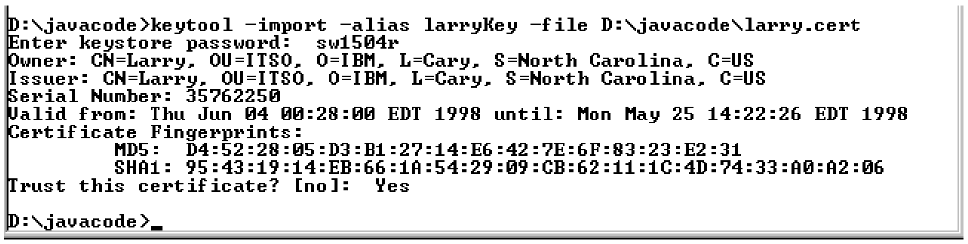


```
D:\javacode>keytool -export -keystore D:\javacode\larryKeyStore -alias larryKey
-file larry.cert
Enter keystore password:  sw1504r
Certificate stored in file <larry.cert>
```

Figure 45. Exporting the Certificate

Then we imported the certificate as trusted root in the default keystore. The command we used to do that was:

```
keytool -import -alias larryKey -file D:\javacode\larry.cert
```



```
D:\javacode>keytool -import -alias larryKey -file D:\javacode\larry.cert
Enter keystore password:  sw1504r
Owner: CN=Larry, OU=ITSO, O=IBM, L=Cary, S=North Carolina, C=US
Issuer: CN=Larry, OU=ITSO, O=IBM, L=Cary, S=North Carolina, C=US
Serial Number: 35762250
Valid from: Thu Jun 04 00:28:00 EDT 1998 until: Mon May 25 14:22:26 EDT 1998
Certificate Fingerprints:
         MD5:  D4:52:28:05:D3:B1:27:14:E6:42:7E:6F:83:23:E2:31
         SHA1: 95:43:19:14:EB:66:1A:54:29:09:CB:62:11:1C:4D:74:33:A0:A2:06
Trust this certificate? [no]:  Yes

D:\javacode>_
```

Figure 46. Import the Certificate in the default Keystore

As you can see, we entered `Yes` when the system asked us if we intended to trust the certificate.

Finally, we put the applet class file MyApplet.class in the JAR file MyApplet.jar as follows:

```
jar cvf MyApplet.jar MyApplet.class
```

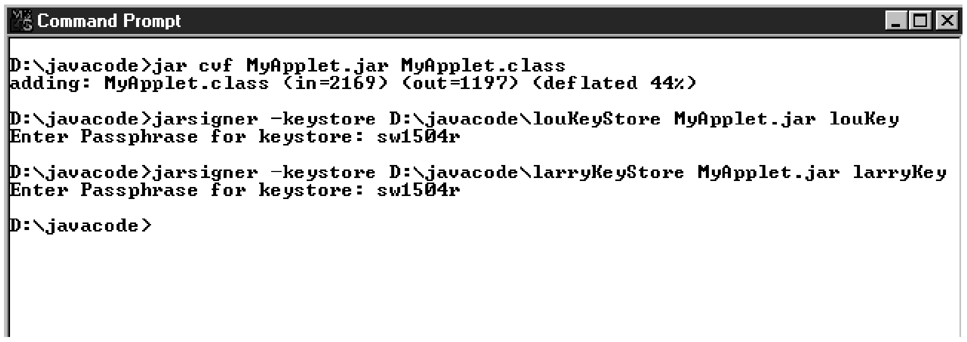
We signed it with *both the keys* by entering the command:

```
jarsigner -keystore D:\javacode\louKeyStore MyApplet.jar louKey
```

This was followed by:

```
jarsigner -keystore D:\javacode\larryKeyStore MyApplet.jar larryKey
```

as shown in the following figure:



```
D:\javacode>jar cvf MyApplet.jar MyApplet.class
adding: MyApplet.class (in=2169) (out=1197) (deflated 44%)

D:\javacode>jarsigner -keystore D:\javacode\louKeyStore MyApplet.jar louKey
Enter Passphrase for keystore: sw1504r

D:\javacode>jarsigner -keystore D:\javacode\larryKeyStore MyApplet.jar larryKey
Enter Passphrase for keystore: sw1504r

D:\javacode>
```

Figure 47. Creating and Signing Twice the JAR file Containing the Applet.

In other words, we applied a signature using the certificate stored in the louKeyStore and then, on the resulting file, still named MyApplet.jar, we applied a second signature, using this time the certificate stored in the larryKeyStore. We want to demonstrate, in this way, that permissions are additive in JDK 1.2.

Now wipe off all the entries in your policy file, and type a new policy file with the following entries.

```
grant codeBase "http://9.24.104.51/javacode/" {
  permission java.net.SocketPermission "marcop.austin.ibm.com:44444", "connect";
};

grant signedBy "lou" {
  permission java.io.FilePermission "D:\\tmp\\narry\\file1", "read";
};

grant signedBy "larryKey" {
  permission java.io.FilePermission "D:\\tmp\\tony\\larry", "read";
};
```

Figure 48. The Policy File to Experiment with Additive Permissions

In summary, what we've said here is the following:

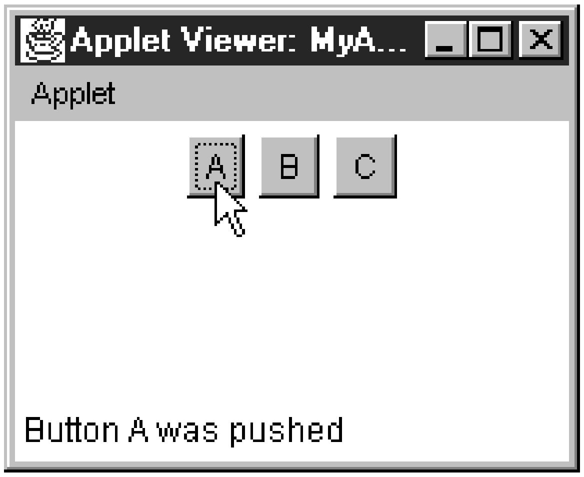
1. We wish to grant any code, originating from the directory HTML\javacode of the Web server having IP address 9.24.104.51, the SocketPermission necessary to connect to the machine marcop.austin.ibm.com on port 44444.
2. We wish to grant to any code signed by the public key registered with the default keystore as lou, the FilePermission to read the file named file1.
3. We wish to grant access to any code signed by the public key registered with the default keystore as larrykey, the FilePermission to read the file named larry.

Since our applet was signed by both these entities, and resided in the particular location HTML\javacode of the Web server having IP address 9.24.104.51, it got *all* the three permissions.

Note that if you want to use another keystore instead of the default keystore, you'll have to add its URL in the policy file as well:

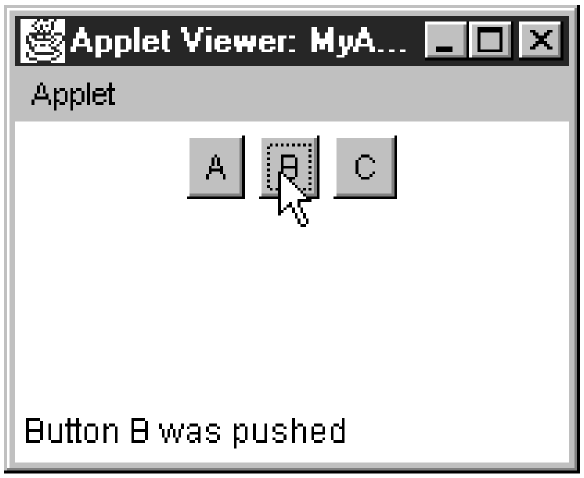
`keystore "file:/`*location*`/`*keyStoreName*`";`

The following three figures show the applet running in the Applet Viewer:



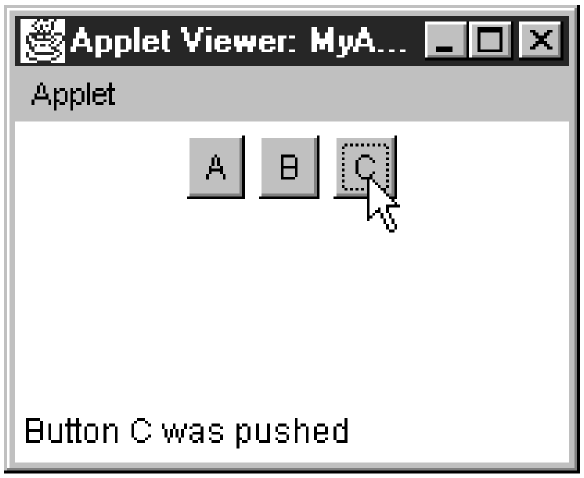
Button A was pushed

Figure 49. The Applet, With All the Permissions, Opens the Socket



Button B was pushed

Figure 50. The Applet, With All the Permissions, Reads the File file1



Button C was pushed

Figure 51. The Applet, With All the Permissions, Reads the File larry

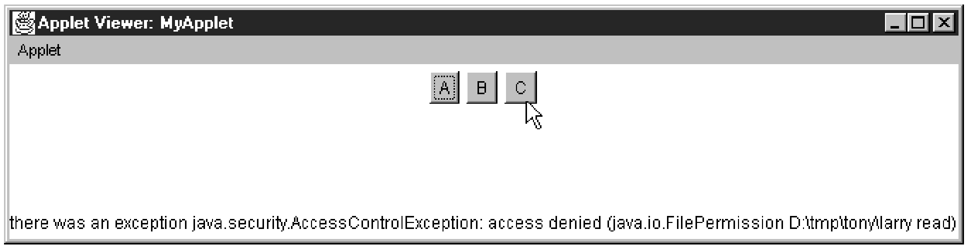
If you read the source code for the MyApplet applet, you will see that the socket connection and the file access actions are activated just when the buttons are pushed.

Now you can try revoking just one of the permissions, by commenting out one of the directives in the java.policy file. Your policy file would now look like this, in case you want to comment out the third `grant` directive:

```
grant codeBase "http://9.24.104.51/javacode/" {
  permission java.net.SocketPermission "marcop.austin.ibm.com:44444", "connect";
};

grant signedBy "lou" {
  permission java.io.FilePermission "D:\\tmp\\narry\\file1", "read";
};

// grant signedby "larryKey" {
//   permission java.io.FilePermission "D:\\tmp\\tony\\larry", "read";
// };
```

Figure 52. Additive Permissions - With One Permission Masked

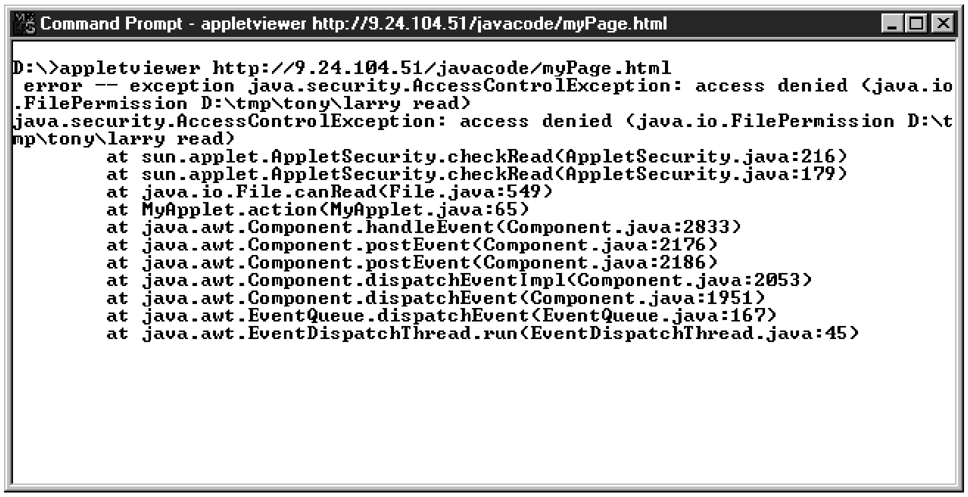
The results would be as expected. On clicking **A** and **B**, you would get what is shown in Figure 49 on page 64 and Figure 50 on page 65. However, on clicking button **C**, a java.security.AccessControlException would be thrown and the Applet Viewer would register that read access is denied to the file D:\tmp\tony\larry.



there was an exception java.security.AccessControlException: access denied (java.io.FilePermission D:\tmp\tony\larry read)

Figure 53. The Applet Window, With the Third Permission Masked

You would simultaneously get a host of exceptions on the console, as shown.

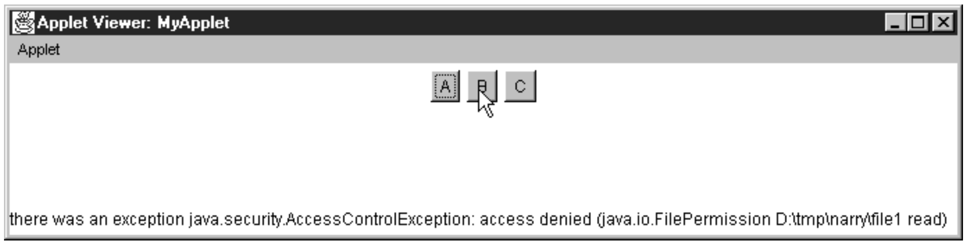

Command Prompt - appletviewer http://9.24.104.51/javacode/myPage.html

```
D:\>appletviewer http://9.24.104.51/javacode/myPage.html
 error -- exception java.security.AccessControlException: access denied (java.io
.FilePermission D:\tmp\tony\larry read)
java.security.AccessControlException: access denied (java.io.FilePermission D:\t
mp\tony\larry read)
        at sun.applet.AppletSecurity.checkRead(AppletSecurity.java:216)
        at sun.applet.AppletSecurity.checkRead(AppletSecurity.java:179)
        at java.io.File.canRead(File.java:549)
        at MyApplet.action(MyApplet.java:65)
        at java.awt.Component.handleEvent(Component.java:2833)
        at java.awt.Component.postEvent(Component.java:2176)
        at java.awt.Component.postEvent(Component.java:2186)
        at java.awt.Component.dispatchEventImpl(Component.java:2053)
        at java.awt.Component.dispatchEvent(Component.java:1951)
        at java.awt.EventQueue.dispatchEvent(EventQueue.java:167)
        at java.awt.EventDispatchThread.run(EventDispatchThread.java:45)
```

Figure 54. The Command Prompt - Showing the Stack Trace.

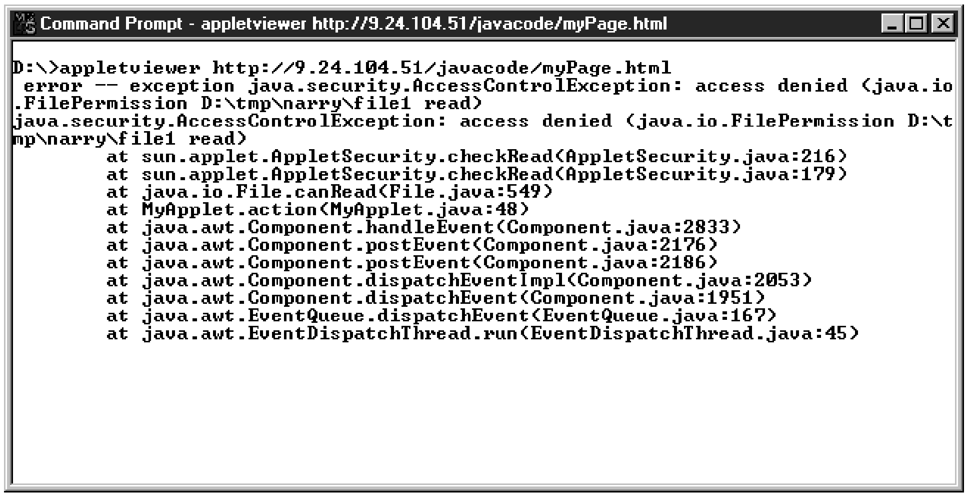
You would find similar outputs when you try revoking one or more permissions and running the applet.

For example, if you mask the second permission and click on **B**, here's the output:



```
there was an exception java.security.AccessControlException: access denied (java.io.FilePermission D:\tmp\narry\file1 read)
```

Figure 55. The Applet Window With the Second Permission Masked

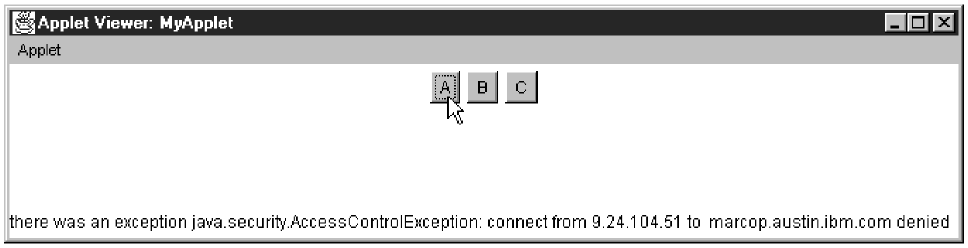

Command Prompt - appletviewer http://9.24.104.51/javacode/myPage.html

```
D:\>appletviewer http://9.24.104.51/javacode/myPage.html
 error -- exception java.security.AccessControlException: access denied (java.io
.FilePermission D:\tmp\narry\file1 read)
java.security.AccessControlException: access denied (java.io.FilePermission D:\t
mp\narry\file1 read)
        at sun.applet.AppletSecurity.checkRead(AppletSecurity.java:216)
        at sun.applet.AppletSecurity.checkRead(AppletSecurity.java:179)
        at java.io.File.canRead(File.java:549)
        at MyApplet.action(MyApplet.java:48)
        at java.awt.Component.handleEvent(Component.java:2833)
        at java.awt.Component.postEvent(Component.java:2176)
        at java.awt.Component.postEvent(Component.java:2186)
        at java.awt.Component.dispatchEventImpl(Component.java:2053)
        at java.awt.Component.dispatchEvent(Component.java:1951)
        at java.awt.EventQueue.dispatchEvent(EventQueue.java:167)
        at java.awt.EventDispatchThread.run(EventDispatchThread.java:45)
```

Figure 56. The Command Prompt Shows the Corresponding Stack Trace

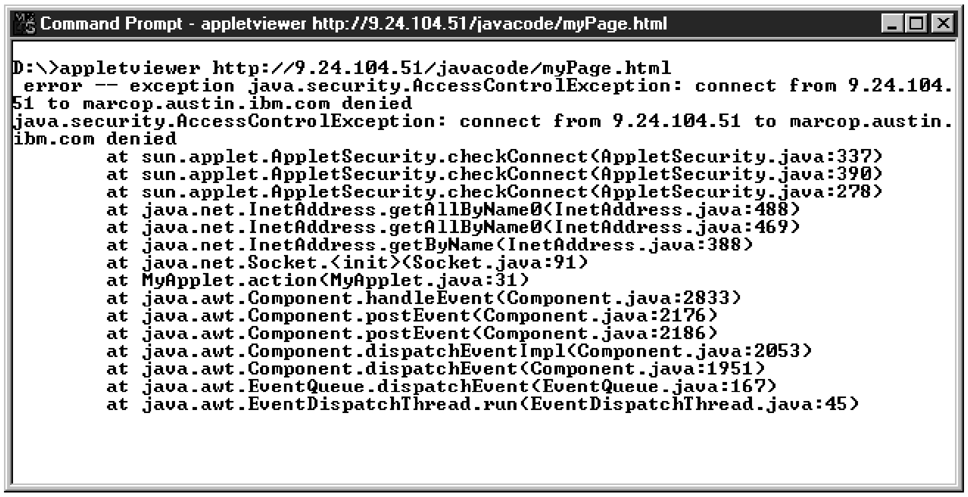
Notice that each time the Applet Viewer and the Java console exactly register the nature of the exception.

Similarly, here's the output when button **A** is pushed and the first permission is masked.



```
there was an exception java.security.AccessControlException: connect from 9.24.104.51 to  marcop.austin.ibm.com denied
```

Figure 57. The Applet Window With the First Permission Masked Out



Command Prompt - appletviewer http://9.24.104.51/javacode/myPage.html

```
D:\>appletviewer http://9.24.104.51/javacode/myPage.html
 error -- exception java.security.AccessControlException: connect from 9.24.104.
51 to marcop.austin.ibm.com denied
java.security.AccessControlException: connect from 9.24.104.51 to marcop.austin.
ibm.com denied
        at sun.applet.AppletSecurity.checkConnect(AppletSecurity.java:337)
        at sun.applet.AppletSecurity.checkConnect(AppletSecurity.java:390)
        at sun.applet.AppletSecurity.checkConnect(AppletSecurity.java:278)
        at java.net.InetAddress.getAllByName0(InetAddress.java:488)
        at java.net.InetAddress.getAllByName0(InetAddress.java:469)
        at java.net.InetAddress.getByName(InetAddress.java:388)
        at java.net.Socket.<init>(Socket.java:91)
        at MyApplet.action(MyApplet.java:31)
        at java.awt.Component.handleEvent(Component.java:2833)
        at java.awt.Component.postEvent(Component.java:2176)
        at java.awt.Component.postEvent(Component.java:2186)
        at java.awt.Component.dispatchEventImpl(Component.java:2053)
        at java.awt.Component.dispatchEvent(Component.java:1951)
        at java.awt.EventQueue.dispatchEvent(EventQueue.java:167)
        at java.awt.EventDispatchThread.run(EventDispatchThread.java:45)
```

Figure 58. The Corresponding Command Prompt Output

Thus, it has been verified that permissions work in a way to make them additive.

We would like to mention, however, a problem that we encountered with running a class from its home directory. It seemed to be a problem with the beta 3 version that we were using of the JDK, in that the policy file was not implemented if we run an applet using the Applet Viewer from the same directory that contained the particular class file of the applet. This might be a bug.

We also found a bug when we tried to add multiple PropertyPermission objects. It appears that any PropertyPermission overwrites any preceding PropertyPermission that applies to the same target. For example, let's consider the following policy file:

```
grant codeBase "file:/C:/tmp/julie" {
  permission java.util.PropertyPermission "java.version", "write";
};

grant signedBy "marco" {
  permission java.util.PropertyPermission "java.version", "read";
};
```

The code in C:\tmp\julie and signed by marco matches both `grant` entries. However, the code will *not* have write access to the java.version property because the permission in the first `grant` entry was overwritten by the permission in the second `grant` entry. PropertyPermission is the only class we found with this problem.

Chapter 3. Web Server Security

Security on the server side has gained importance in recent years, owing to well-publicized (albeit isolated) incidents of *break-ins*. In this chapter, we outline the aspects of Web server security. We start off with security features offered by Lotus Domino Go Webserver 4.6.2.2, including support for Secure Socket Layer (SSL) and the mechanism to protect resources through Access Control Lists (ACLs). Then, in Chapter 4, “IBM WebSphere Application Server Security” on page 121, we give a comprehensive treatment of IBM WebSphere Application Server security. Notice that, at the time this book went to print, IBM WebSphere Application Server was still in its beta 3 version, and carried the name of IBM ServletExpress. Moreover, as you can see in Chapter 2, “The New Java 1.2 Security Model” on page 9, at the time of this project the Java Development Kit 1.2 was available only in beta version, so Lotus Domino Go Webserver, ServletExpress and all the Web browsers we used still supported JDK 1.1.

If you want to see the simple steps required to install and configure Lotus Domino Go Webserver correctly, you can refer to 7.2, “Lotus Domino Go Webserver 4.6.2.2” on page 302.

3.1 Lotus Domino Go Webserver Security Features

Lotus Domino Go Webserver is part of the recently announced IBM WebSphere software family. Domino Go is a secure server, with support for SSL 3.0, HTTP 1.1, JDK 1.1, Just-in-Time (JIT) compiler, Java servlets, site indexing capabilities, advanced server statistics reporting and relational database connectivity with Net.Data. It is available on many platforms, like OS/390, AIX, Solaris, HP-UX, OS/2 Warp, Windows NT and Windows 95. It also has the ability to generate certificate requests, sign certificates and even act as a Certification Authority (CA) for an intranet with the help of the CAServlet being shipped with ServletExpress. Domino Go Webserver is characteristically very scalable, and has hosted some of the most heavily loaded sites like the Atlanta Summer Olympics Web site and the Big Blue vs. Kasparov chess match site. In the next few sections, we will be sharing with you our experiences using the security features of this Web server.

3.1.1 SSL Overview

SSL is a standard proposed by Netscape for secure communication over the Internet. SSL operates above the transport layer of the ISO/OSI Network Model, and can be used to secure the communication between any of the TCP/IP suite application layer protocols, like HTTP, telnet, FTP. The HTTPS protocol is HTTP over SSL. The URL of a secure Web site starts with `https://`, rather than the usual `http://`.

We are giving here a very brief description of the basic functioning of the SSL protocol. For further information, the user is directed to the IBM redbook *The Domino Defense: Security in Lotus Notes and Internet*, SG24-2109.

In its most basic form, SSL guarantees transmission of encrypted data over the net and enables server authentication. SSL grants:

- Confidentiality

All client requests and server responses are encrypted to maintain the confidentiality of the data exchanged over the network.

- Data integrity

  Data that flows between a client and a server is protected from a third party's tampering.

Optionally, clients may be required to authenticate to the server by providing their own digital certificates.

When the client connects a secure server, the process known as SSL handshake starts. The server authenticates itself to the client by first sending to the client it's digital certificate. If the client decides to trust the server, the process goes on involving the server and the client agreeing upon a secret session key with which to encrypt all the messages sent either way (from the client to the server, or the other way). This is done by the client generating a random symmetric key, encrypting it with the server's public key, and sending it back to the server. This could well be the SSL session key if the server agrees. Else, they negotiate till they agree upon a single symmetric key with which to encrypt all the information sent across the Internet. Note that a separate session is maintained for each client, and it is the server's responsibility to maintain a mapping of client vs. session key. Also note that even the URLs are encrypted while the request is being sent to the server. This could be very significant while using the GET or POST HTTP methods with an HTML form, to send some sensitive information, say, a password or a credit card number, to the server.

The latest version of SSL, Version 3.0, enables client authentication. Actually SSL 2.0 also supported client authentication, but there were some bugs with the original version. These bugs were subsequently fixed, but not many browsers picked up the changes. For further details about the enhancements of SSL Version 3.0 vs. SSL Version 2.0, see 5.1.1, “Netscape Navigator” on page 181. Using the client authentication, the server can now be aware of who exactly the client is. The server could, among other things, restrict access to clients based on their digital certificates, or give them personalized information.

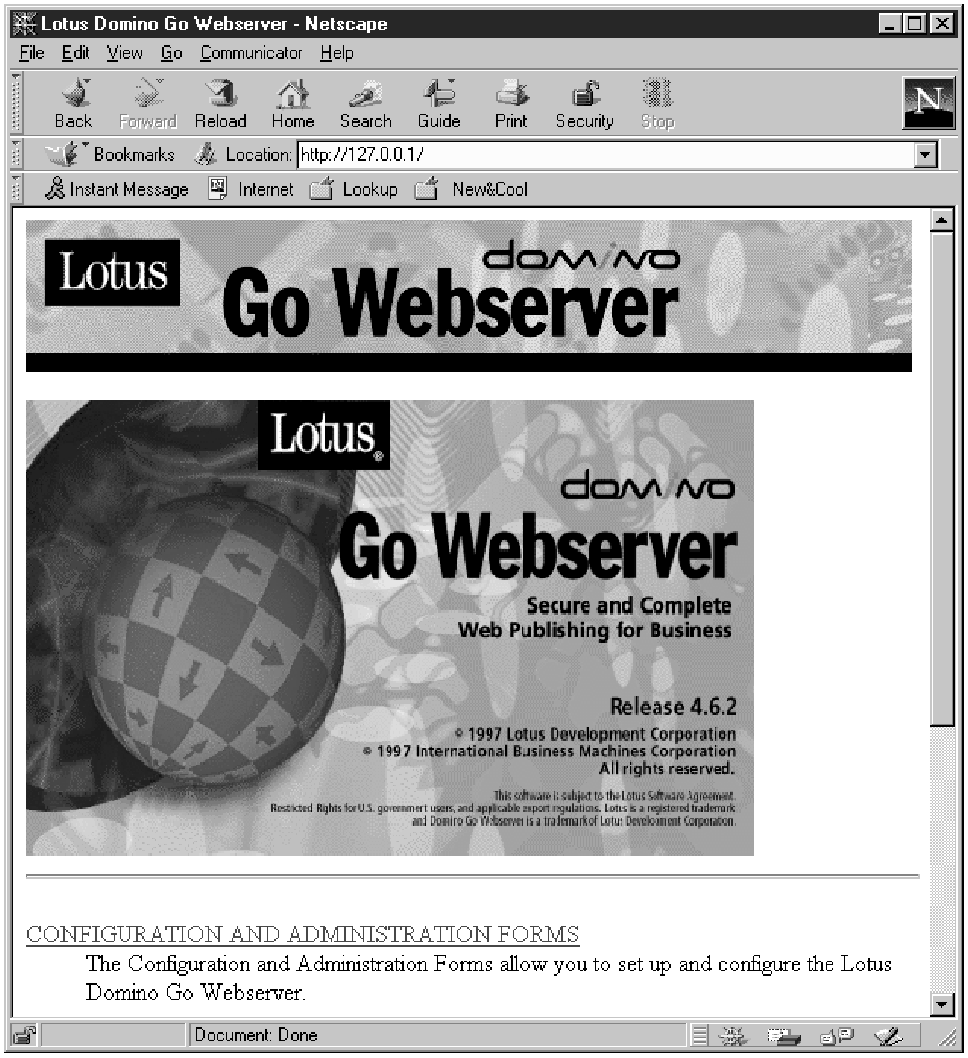
An example for this can be found on the VeriSign Web site http://www.verisign.com/products/sites/job_demo/index.html. Make sure you visit this site *after* collecting the trial browser certificate from http://www.verisign.com/client/index.html (see 5.4.1, “Obtaining a VeriSign Evaluation Certificate” on page 232). When you do so, you will see how it is possible for the Web server to provide customized information to the user, without requiring the user to remember yet another user ID and password, using SSL 3.0 authentication and client certificates (see 5.4.2, “Using a Certificate to Access Secure Web Sites” on page 238).

In this chapter we will show you, among other things, how to set up Lotus Domino Go Webserver for using SSL and how SSL server authentication works. Client authentication, which is optional and not really necessary to activate a SSL session, is described in 3.1.4, “SSL Client Authentication” on page 84, where you can also find how to install a CA certificate into Lotus Domino Go Webserver.

3.1.2 Lotus Domino Go Webserver SSL Setup

This section shows how to configure SSL on Lotus Domino Go Webserver. SSL-supported server authentication uses RSA public key cryptography, together with an independent Certificate Authority (CA) for server certificate authentication. In this section we will not use any independent CA for server authentication, but we will use the functionality of Lotus Domino Go Webserver to be self-signed CA. This functionality is very useful not only for testing purposes but also when the client and the server are both hosts of the same intranet.

First of all, you must access the Configuration and Administration Forms for your Lotus Domino Go Webserver. To do this, it is enough that you invoke the host name or the IP address of your Web server machine from a client's browser while Lotus Domino Go Webserver is running.

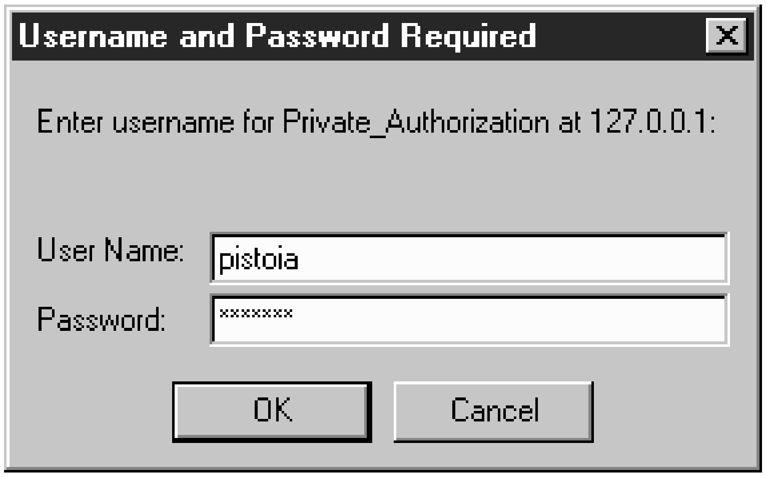

Lotus Domino Go Webserver - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://127.0.0.1/
Instant Message Internet Lookup New&Cool
Lotus domino Go Webserver
Secure and Complete Web Publishing for Business
Release 4.6.2
© 1997 Lotus Development Corporation
© 1997 International Business Machines Corporation
All rights reserved.
Document: Done

CONFIGURATION AND ADMINISTRATION FORMS

The Configuration and Administration Forms allow you to set up and configure the Lotus Domino Go Webserver.

Figure 59. How to Access the Configuration and Administration Forms

Of course, you must enter the Administrator user ID and password to log on.



Username and Password Required

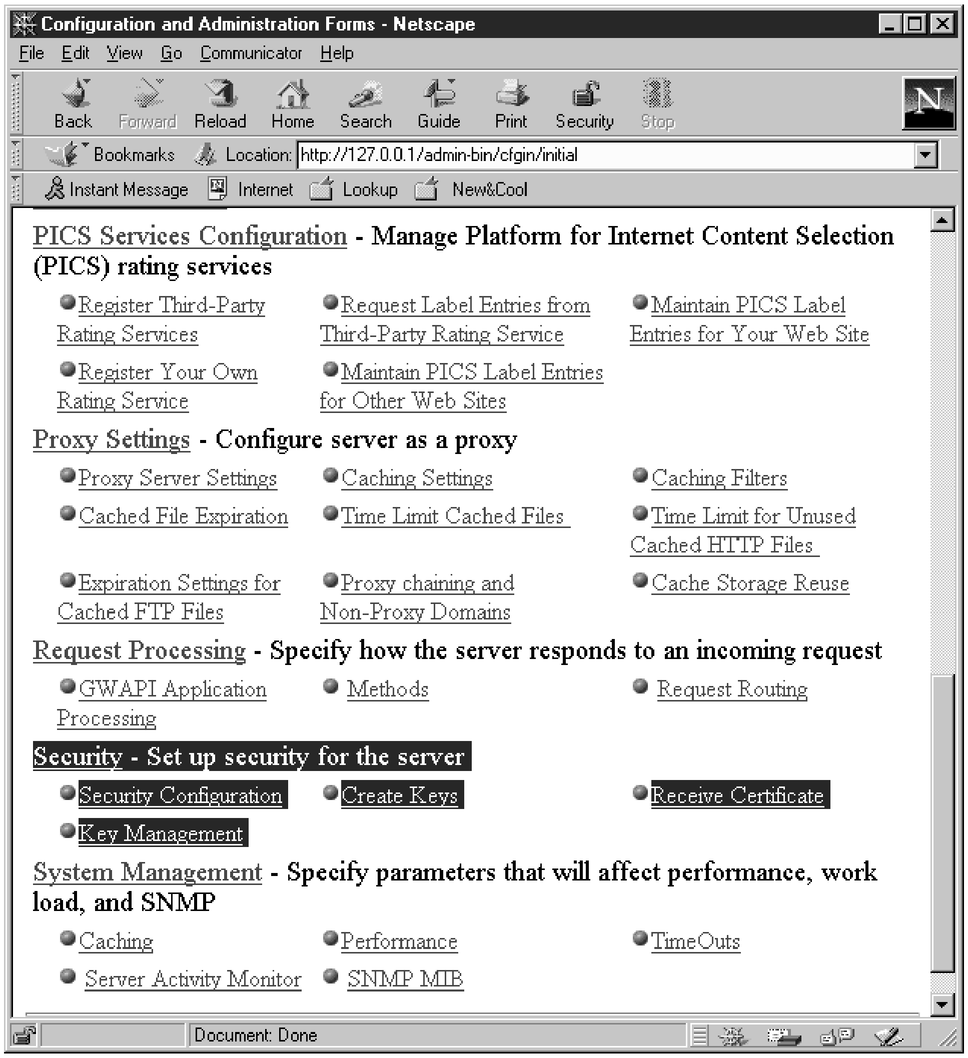
Enter username for Private_Authorization at 127.0.0.1:

User Name: pistoia

Password: *******

Figure 60. Entering User ID and Password

Once the Configuration and Administration Forms home page for your Web server is displayed, you should move down the scroll bar in order to find the Security section, which is highlighted in the following screen.



Configuration and Administration Forms - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location:
Instant Message Internet Lookup New&Cool

http://127.0.0.1/admin-bin/cfgin/initial

PICS Services Configuration - Manage Platform for Internet Content Selection (PICS) rating services

- Register Third-Party Rating Services
- Request Label Entries from Third-Party Rating Service
- Maintain PICS Label Entries for Your Web Site
- Register Your Own Rating Service
- Maintain PICS Label Entries for Other Web Sites

Proxy Settings - Configure server as a proxy

- Proxy Server Settings
- Caching Settings
- Caching Filters
- Cached File Expiration
- Time Limit Cached Files
- Time Limit for Unused Cached HTTP Files
- Expiration Settings for Cached FTP Files
- Proxy chaining and Non-Proxy Domains
- Cache Storage Reuse

Request Processing - Specify how the server responds to an incoming request

- GWAPI Application Processing
- Methods
- Request Routing

Security - Set up security for the server

- Security Configuration
- Create Keys
- Receive Certificate
- Key Management

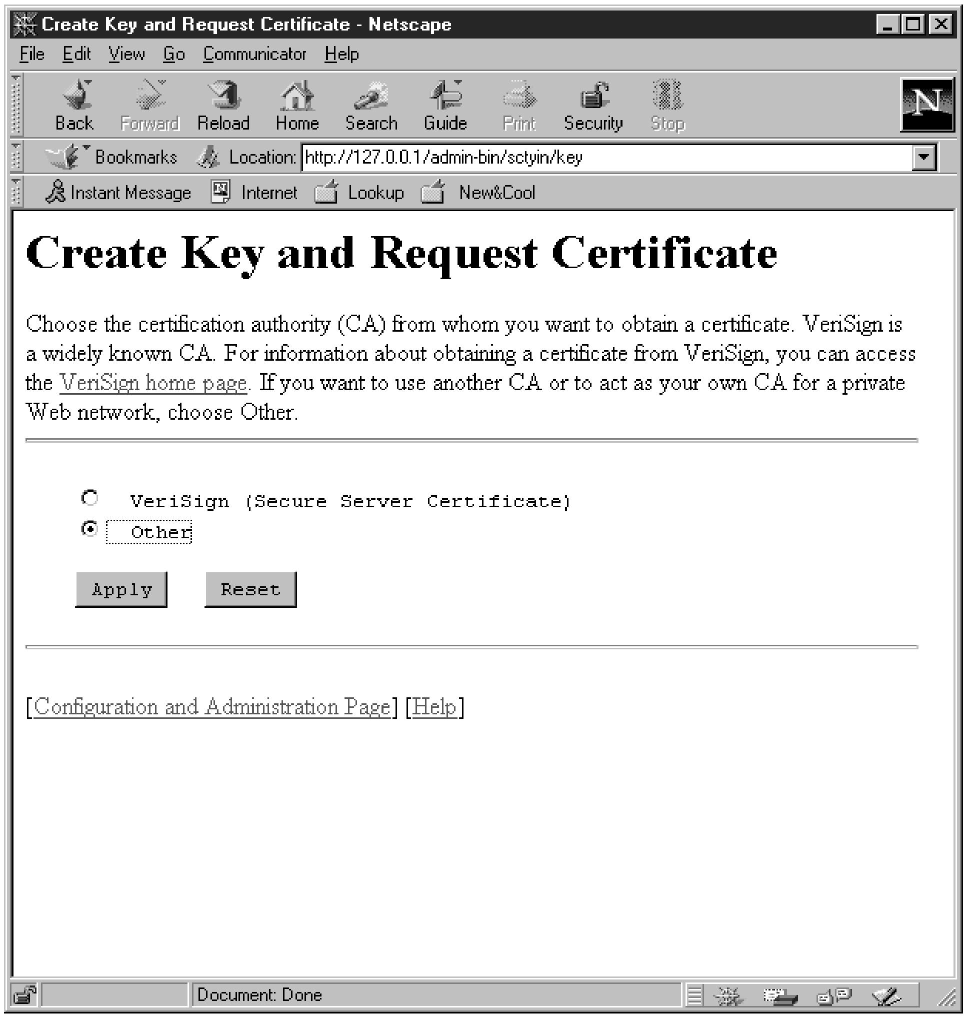
System Management - Specify parameters that will affect performance, work load, and SNMP

- Caching
- Performance
- TimeOuts
- Server Activity Monitor
- SNMP MIB

Document: Done

Figure 61. Security Section in the Configuration and Administration Forms

When you select **Create Keys**, the following window will be displayed:


Create Key and Request Certificate - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://127.0.0.1/admin-bin/sctyin/key
Instant Message Internet Lookup New&Cool
Document: Done

Create Key and Request Certificate

Choose the certification authority (CA) from whom you want to obtain a certificate. VeriSign is a widely known CA. For information about obtaining a certificate from VeriSign, you can access the VeriSign home page. If you want to use another CA or to act as your own CA for a private Web network, choose Other.

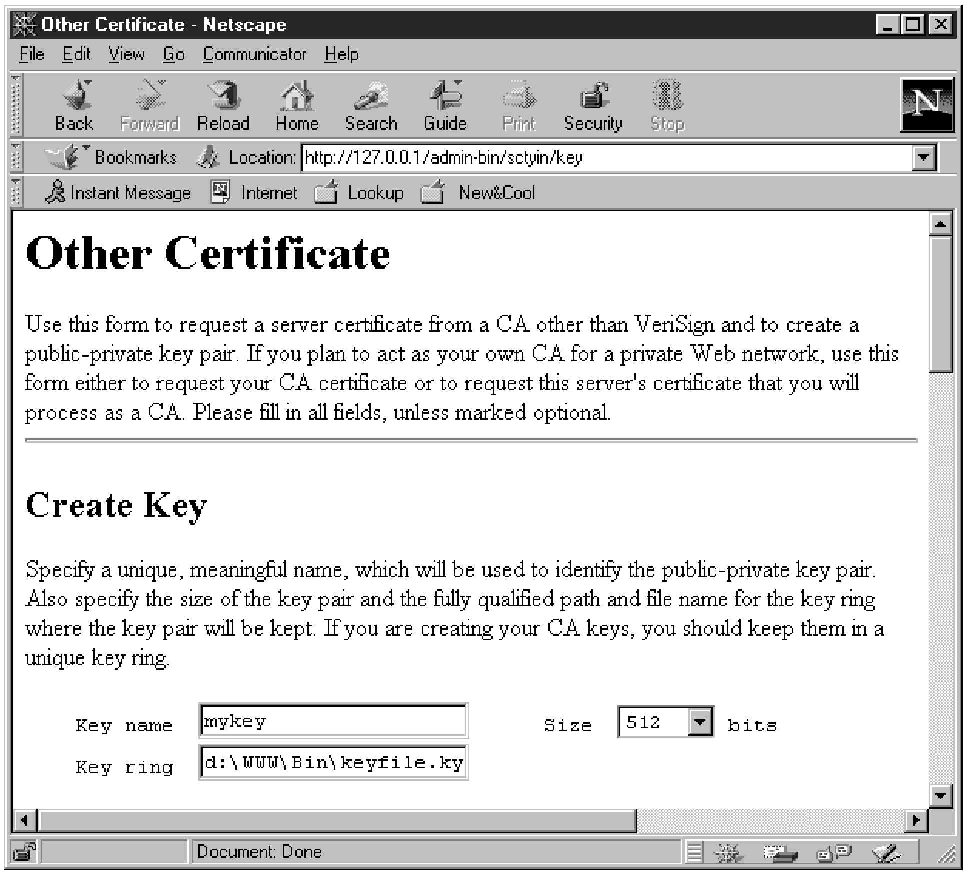
VeriSign (Secure Server Certificate) [✓ Other]

Apply Reset

[Configuration and Administration Page] [Help]

Figure 62. Request Certificate

On this page, you choose the CA from which you want to obtain the certificate. We marked **Other**, because we wanted to act as our own CA for our intranet. When we clicked **Apply**, we got the Other Certificate Web page, shown here divided into four different figures. The first section is to create a key:


Other Certificate - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://127.0.0.1/admin-bin/sctyin/key
Instant Message Internet Lookup New&Cool
Document: Done

Other Certificate

Use this form to request a server certificate from a CA other than VeriSign and to create a public-private key pair. If you plan to act as your own CA for a private Web network, use this form either to request your CA certificate or to request this server's certificate that you will process as a CA. Please fill in all fields, unless marked optional.

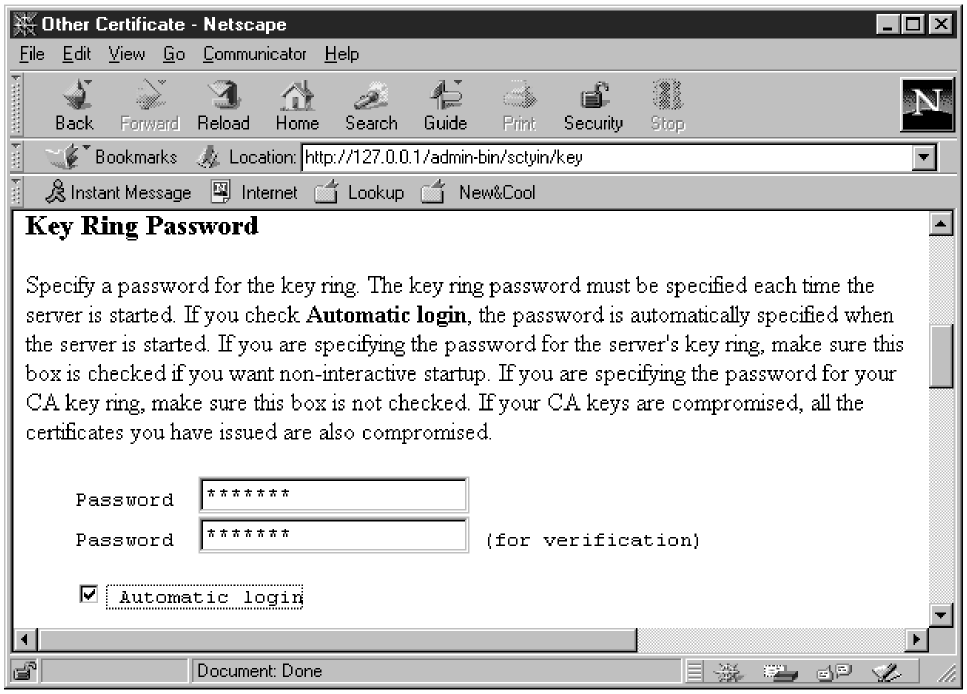
Create Key

Specify a unique, meaningful name, which will be used to identify the public-private key pair. Also specify the size of the key pair and the fully qualified path and file name for the key ring where the key pair will be kept. If you are creating your CA keys, you should keep them in a unique key ring.

Key name: mykey Size: 512 bits
Key ring: d:\WWW\Bin\keyfile.ky

Figure 63. Create Key

The second section is named Key Ring Password. A *key ring* is a file used to store public keys, private keys, certificates and trusted root keys. The Key Ring Password section is used to specify a password for the key ring and also to select whether you want an automatic login, so that the password is automatically entered when the server starts. We specified the password and accepted the automatic login function, as shown in the following figure:


Other Certificate - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://127.0.0.1/admin-bin/sctyin/key
Instant Message Internet Lookup New&Cool
Document: Done

Key Ring Password

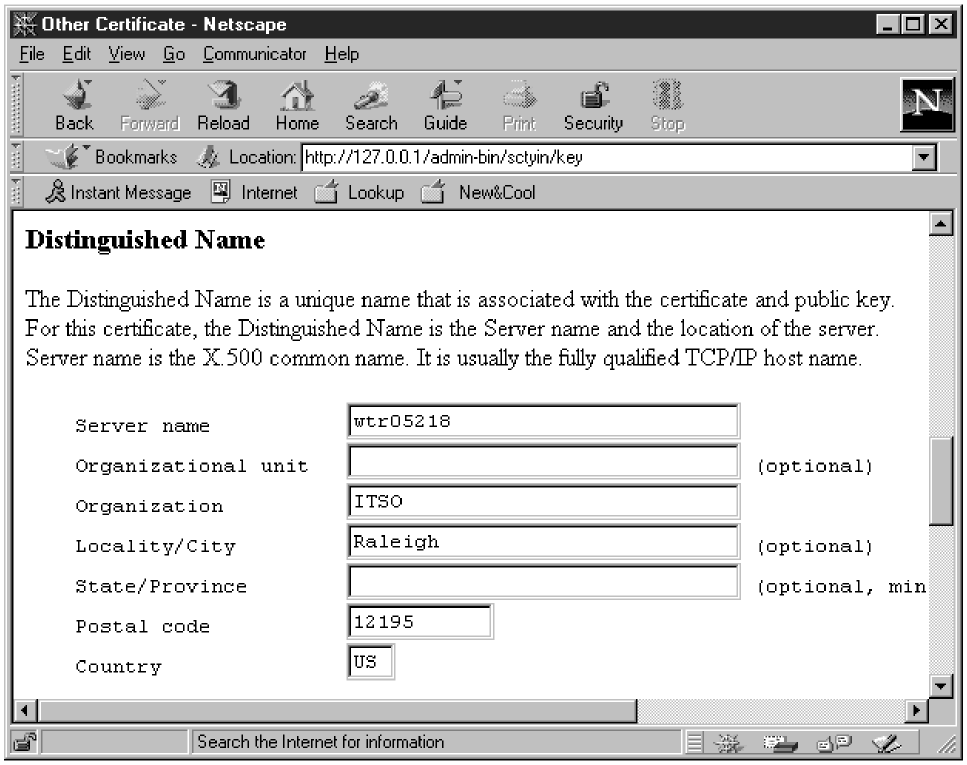
Specify a password for the key ring. The key ring password must be specified each time the server is started. If you check **Automatic login**, the password is automatically specified when the server is started. If you are specifying the password for the server's key ring, make sure this box is checked if you want non-interactive startup. If you are specifying the password for your CA key ring, make sure this box is not checked. If your CA keys are compromised, all the certificates you have issued are also compromised.

Password	*******	
Password	*******	(for verification)

[✓] Automatic login

Figure 64. Key Ring Password

The third section of the Other Certificate page appears if you move down the scroll bar. This section is named Distinguished Name. A Distinguished Name is a unique name associated with the certificate and the public key. We filled in the necessary fields following the directions, as shown in this figure:


Other Certificate - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://127.0.0.1/admin-bin/sctyin/key
Instant Message Internet Lookup New&Cool
Search the Internet for information

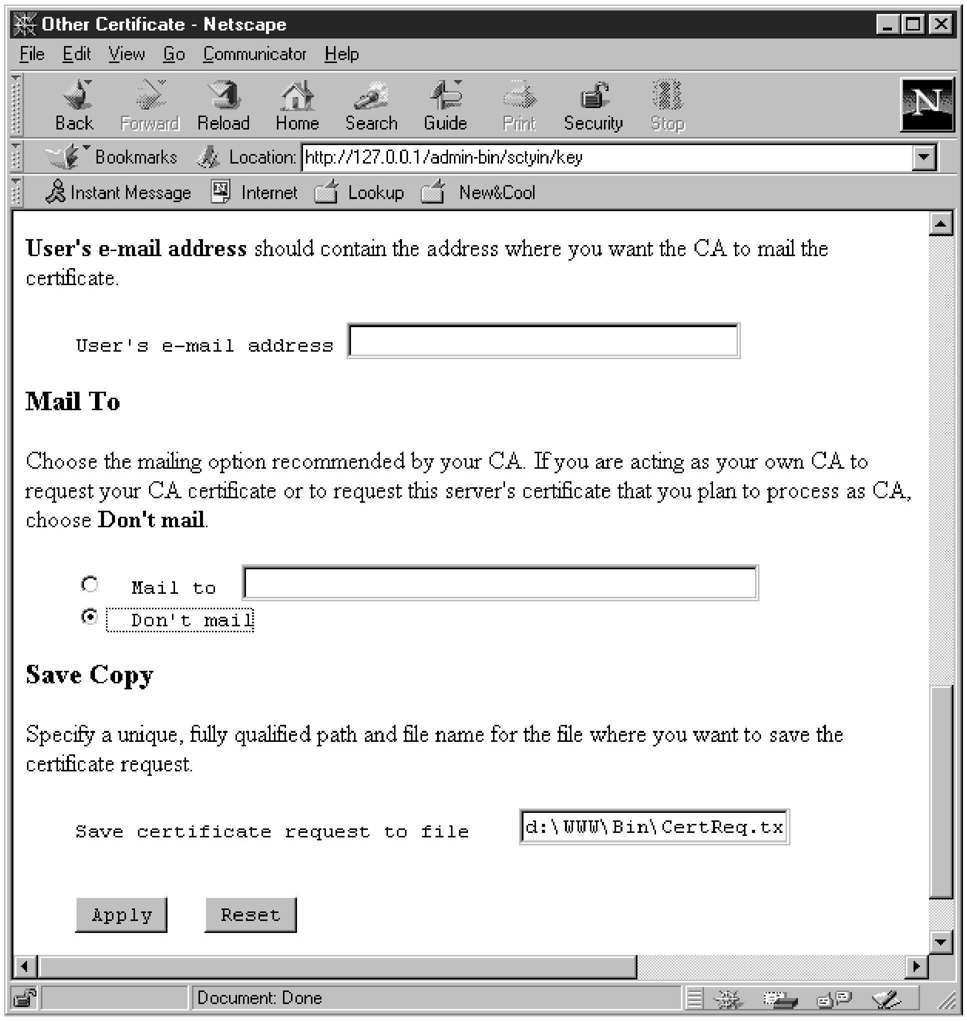
Distinguished Name

The Distinguished Name is a unique name that is associated with the certificate and public key. For this certificate, the Distinguished Name is the Server name and the location of the server. Server name is the X.500 common name. It is usually the fully qualified TCP/IP host name.

Server name	wtr05218	
Organizational unit		(optional)
Organization	ITSO	
Locality/City	Raleigh	(optional)
State/Province		(optional, min
Postal code	12195	
Country	US	

Figure 65. Distinguished Name

Finally, you can specify the e-mail address where the CA should mail the certificate and also the e-mail address of the CA that should receive your certificate request. Since we were acting as our own CA, we selected **Don't mail**.


Other Certificate - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://127.0.0.1/admin-bin/sctyin/key
Instant Message Internet Lookup New&Cool
Document: Done

User's e-mail address should contain the address where you want the CA to mail the certificate.

User's e-mail address

Mail To

Choose the mailing option recommended by your CA. If you are acting as your own CA to request your CA certificate or to request this server's certificate that you plan to process as CA, choose **Don't mail**.

○ Mail to
[✓ Don't mail]

Save Copy

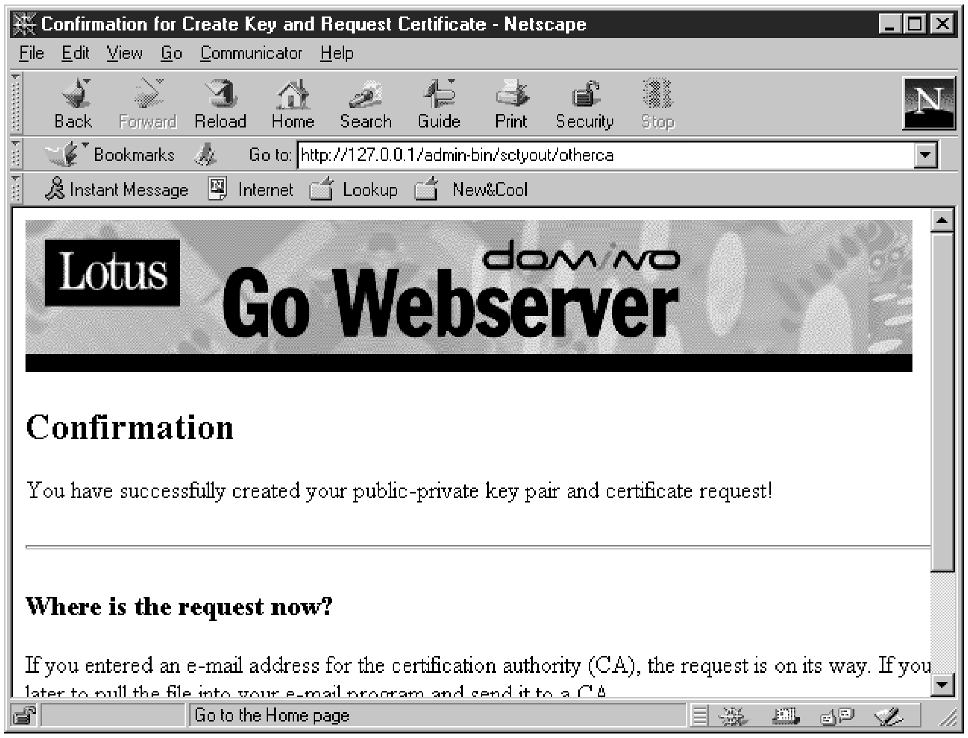
Specify a unique, fully qualified path and file name for the file where you want to save the certificate request.

Save certificate request to file d:\WWW\Bin\CertReq.tx

Apply Reset

Figure 66. Certificate Request

When we clicked **Apply**, we got the following Confirmation window.


Confirmation for Create Key and Request Certificate - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Go to: http://127.0.0.1/admin-bin/sctyout/otherca
Instant Message Internet Lookup New&Cool
Lotus
domino
Go Webserver
Go to the Home page

Confirmation

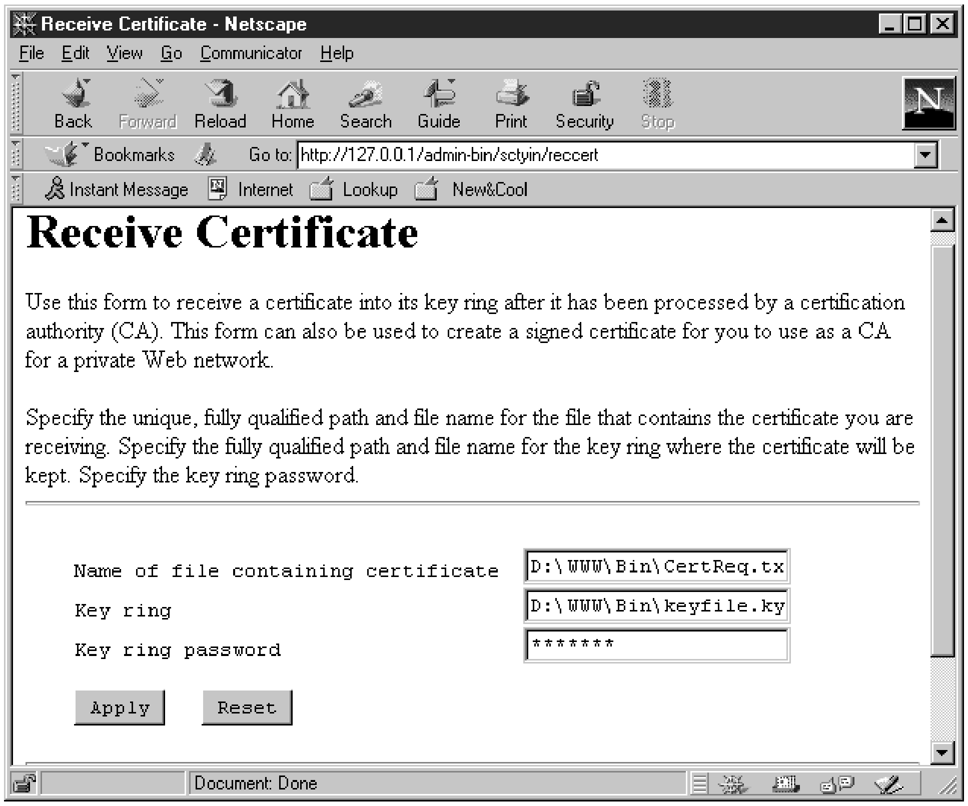
You have successfully created your public-private key pair and certificate request!

Where is the request now?

If you entered an e-mail address for the certification authority (CA), the request is on its way. If you later to pull the file into your e-mail program and send it to a CA

Figure 67. Confirmation Window

You now have a certificate request and you should send it to a Certification Authority, such as Thawte or VeriSign, to sign the certificate request. Lotus Domino Go Webserver permits us to import the certificate request itself to generate a self-signed certificate. To do this, we first went to the Configuration and Administration Forms home page and clicked on the link **Receive Certificate**. It took us to the following page.


Receive Certificate - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Go to: http://127.0.0.1/admin-bin/sctyin/reccert
Instant Message Internet Lookup New&Cool
Apply Reset
Document: Done

Receive Certificate

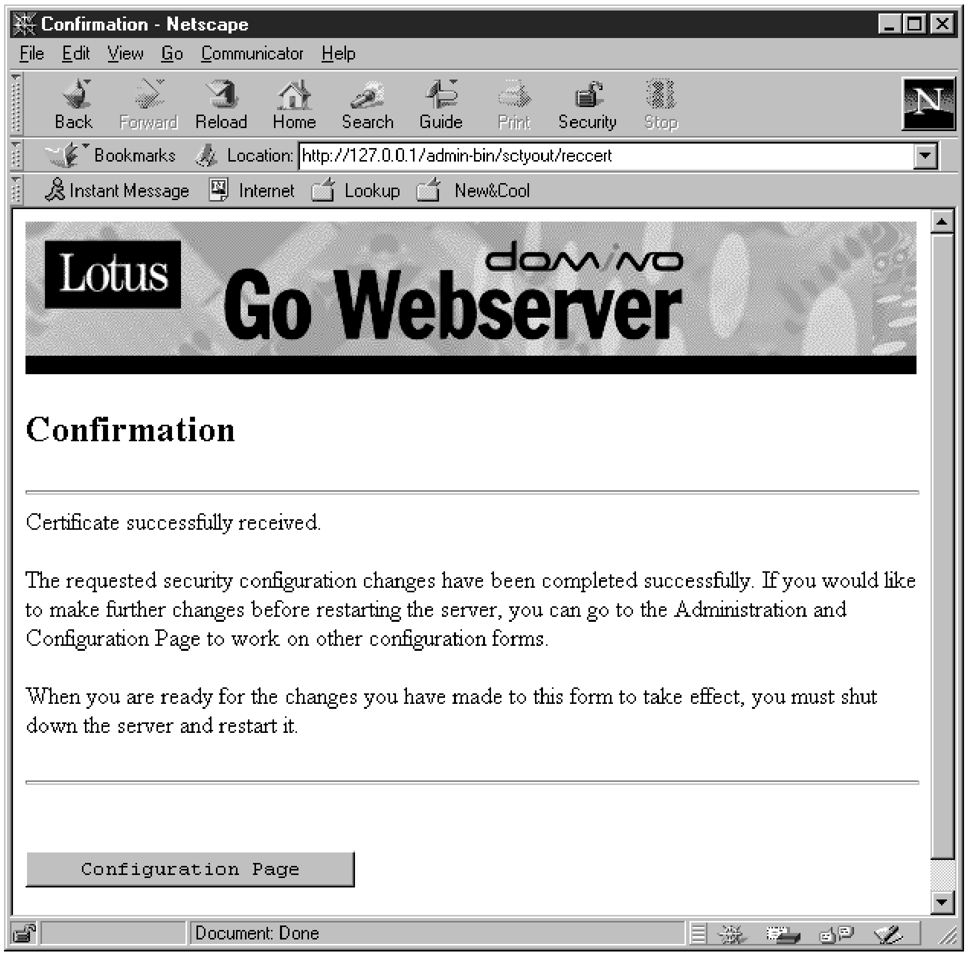
Use this form to receive a certificate into its key ring after it has been processed by a certification authority (CA). This form can also be used to create a signed certificate for you to use as a CA for a private Web network.

Specify the unique, fully qualified path and file name for the file that contains the certificate you are receiving. Specify the fully qualified path and file name for the key ring where the certificate will be kept. Specify the key ring password.

Name of file containing certificate	D:\WWW\Bin\CertReq.tx
Key ring	D:\WWW\Bin\keyfile.ky
Key ring password	*******

Figure 68. Receive Certificate Window

After entering the name of the file into which we wanted to store the certificate request and entering the password, we clicked on **Apply**. We got the following Confirmation page.


Confirmation - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://127.0.0.1/admin-bin/sctyout/reccert
Instant Message Internet Lookup New&Cool
Lotus domino Go Webserver
Configuration Page
Document: Done

Confirmation

Certificate successfully received.

The requested security configuration changes have been completed successfully. If you would like to make further changes before restarting the server, you can go to the Administration and Configuration Page to work on other configuration forms.

When you are ready for the changes you have made to this form to take effect, you must shut down the server and restart it.

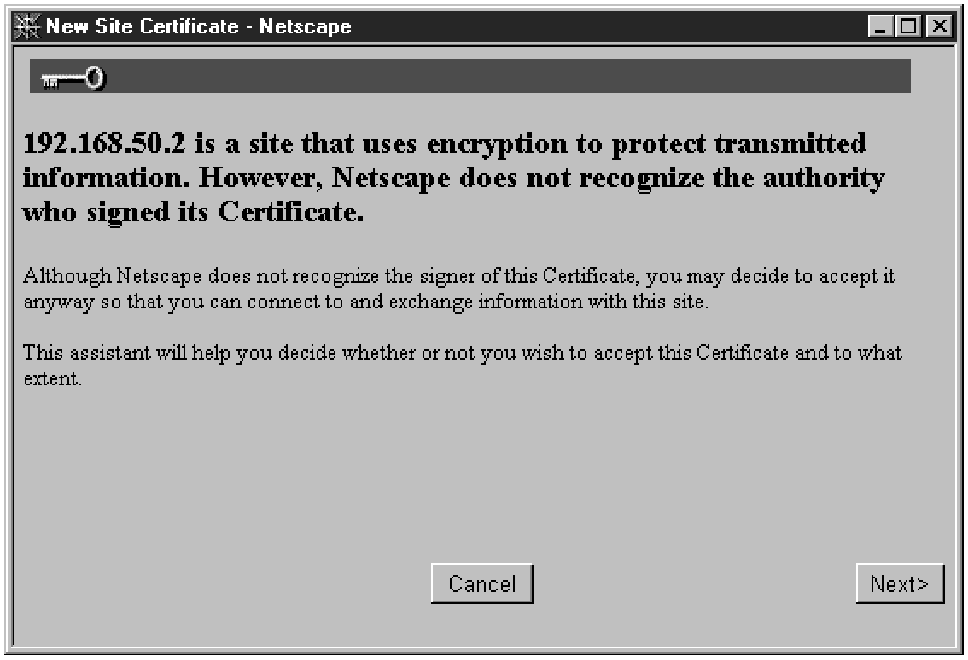
Figure 69. Confirmation Window

We pressed **Configuration Page** and then we restarted the Web server, as the Confirmation window suggests. Now the server is all set to use the self-signed certificate for secure connections with the clients.

3.1.3 Lotus Domino Go Webserver SSL Server Authentication

We will show in this section how the Web server is authenticated by Netscape Communicator running on a client machine before an SSL session is in place. First of all, when you invoke a secure Web server from a client's browser, the URL should begin with `https://` instead of the usual `http://`. You will use `https://` for HTTP URLs with SSL and `http://` for HTTP URLs without SSL.

Netscape Communicator immediately displays a warning message, because Netscape does not recognize the private CA that we set up.



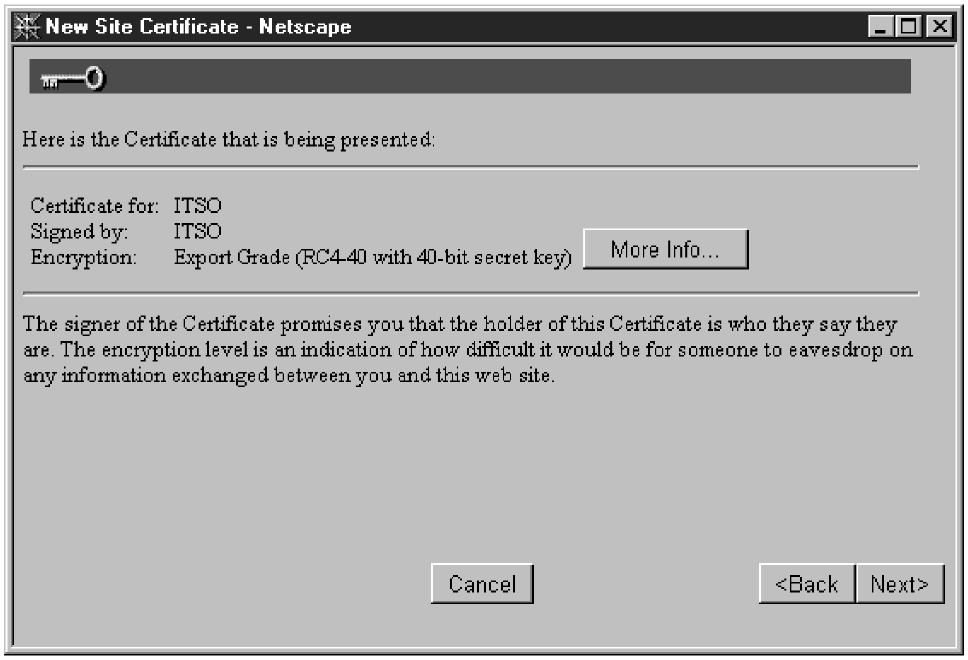
192.168.50.2 is a site that uses encryption to protect transmitted information. However, Netscape does not recognize the authority who signed its Certificate.

Although Netscape does not recognize the signer of this Certificate, you may decide to accept it anyway so that you can connect to and exchange information with this site.

This assistant will help you decide whether or not you wish to accept this Certificate and to what extent.

Figure 70. Security Warning Message from Netscape Communicator

Users must judge whether the certificate proposed by the Web server is acceptable. If you click **Next** you can get basic information about that new certificate and the following page is displayed:



Here is the Certificate that is being presented:

Certificate for:	ITSO
Signed by:	ITSO
Encryption:	Export Grade (RC4-40 with 40-bit secret key)

The signer of the Certificate promises you that the holder of this Certificate is who they say they are. The encryption level is an indication of how difficult it would be for someone to eavesdrop on any information exchanged between you and this web site.

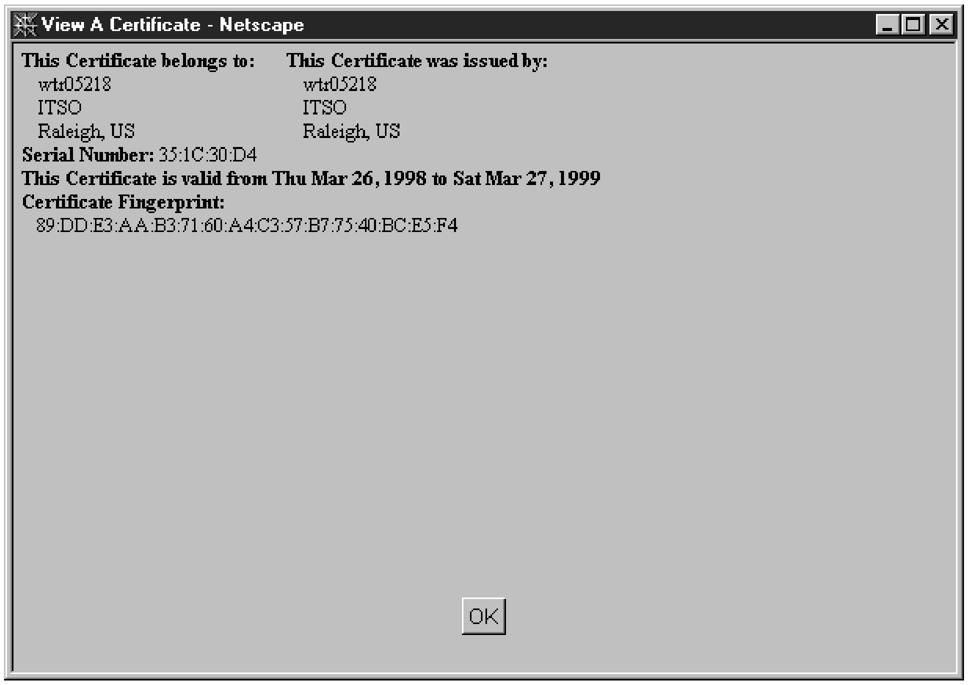
Figure 71. Basic Information About the Web Server Certificate

Notice that the Encryption field indicates the encryption level. In this case we have:

```
RC4-40 with 40-bit secret key
```

This offers us the opportunity to understand how secure the communication between the client and the server will be or, in other words, how easy it would be for someone to decrypt such communication.

In general it is advisable to get more information about a certificate that is not immediately recognized by our browser. So you should click **More Info...** to get the certificate details.



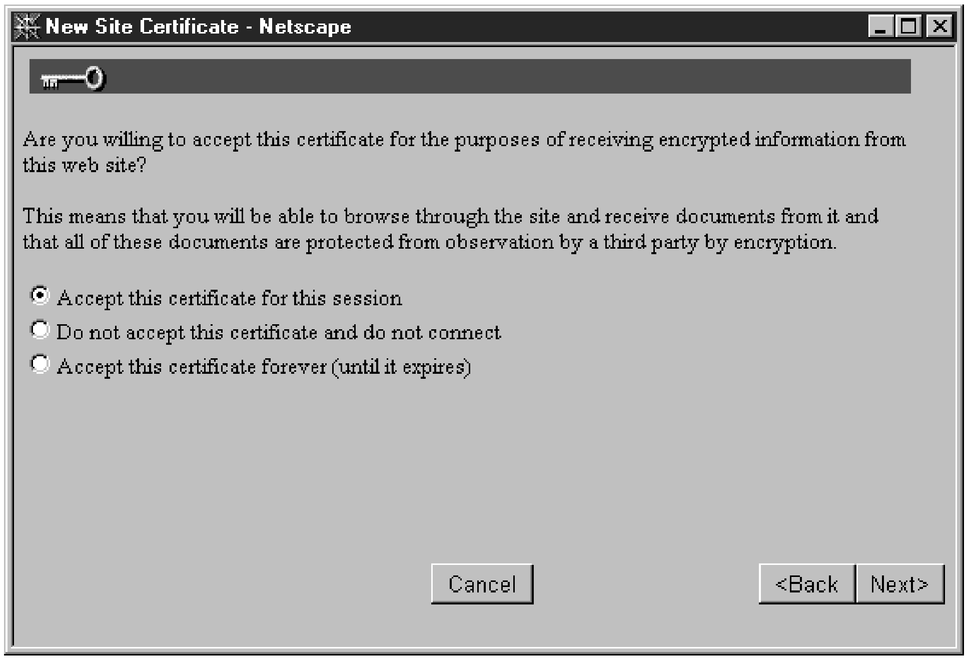
This Certificate belongs to:
wtr05218
ITSO
Raleigh, US

This Certificate was issued by:
wtr05218
ITSO
Raleigh, US

Serial Number: 35:1C:30:D4
This Certificate is valid from Thu Mar 26, 1998 to Sat Mar 27, 1999
Certificate Fingerprint:
89:DD:E3:AA:B3:71:60:A4:C3:57:B7:75:40:BC:E5:F4

Figure 72. Certificate Information

Clicking **OK**, you go back to Figure 71 on page 80 and then you can press **Next** to decide if you want to accept this certificate for the current session only, don't want to accept that certificate at all, or accept it until it expires.



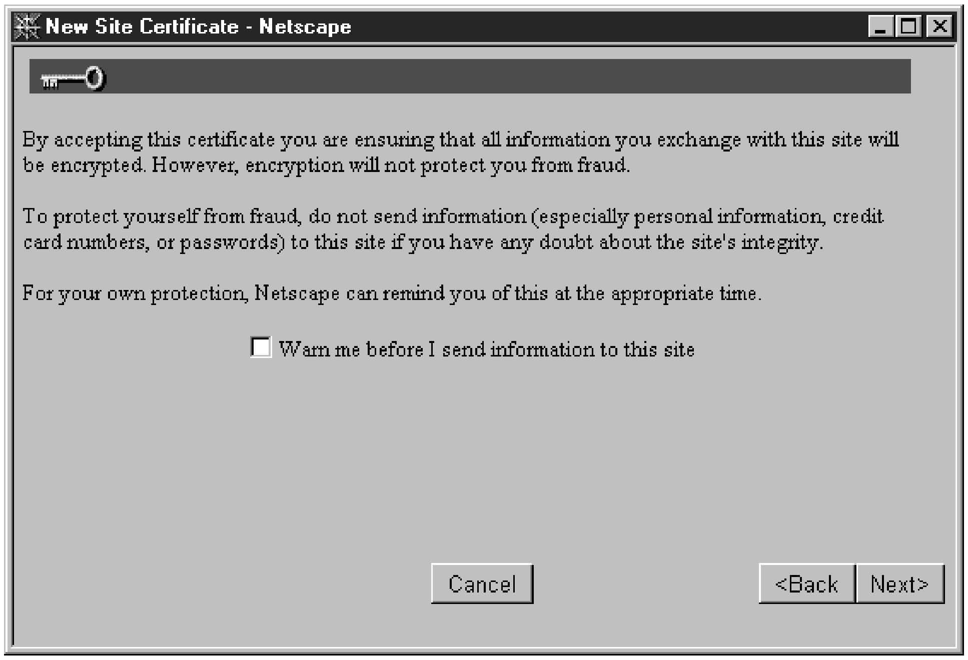
Are you willing to accept this certificate for the purposes of receiving encrypted information from this web site?

This means that you will be able to browse through the site and receive documents from it and that all of these documents are protected from observation by a third party by encryption.

● Accept this certificate for this session
○ Do not accept this certificate and do not connect
○ Accept this certificate forever (until it expires)

Figure 73. Making a Security Decision about the Unknown Web Server

We selected the opportunity to accept this certificate for the current session only, so we pressed **Next** again. Netscape Communicator displays a warning message, reminding us that even if accepting the new certificate ensures encryption in the transmission, this does not exclude the possibility that the Web server is impersonating another well-known server or that the Web server will misuse the information provided to commit fraud. You can also decide if you want to see again this warning message in future circumstances or not.



By accepting this certificate you are ensuring that all information you exchange with this site will be encrypted. However, encryption will not protect you from fraud.

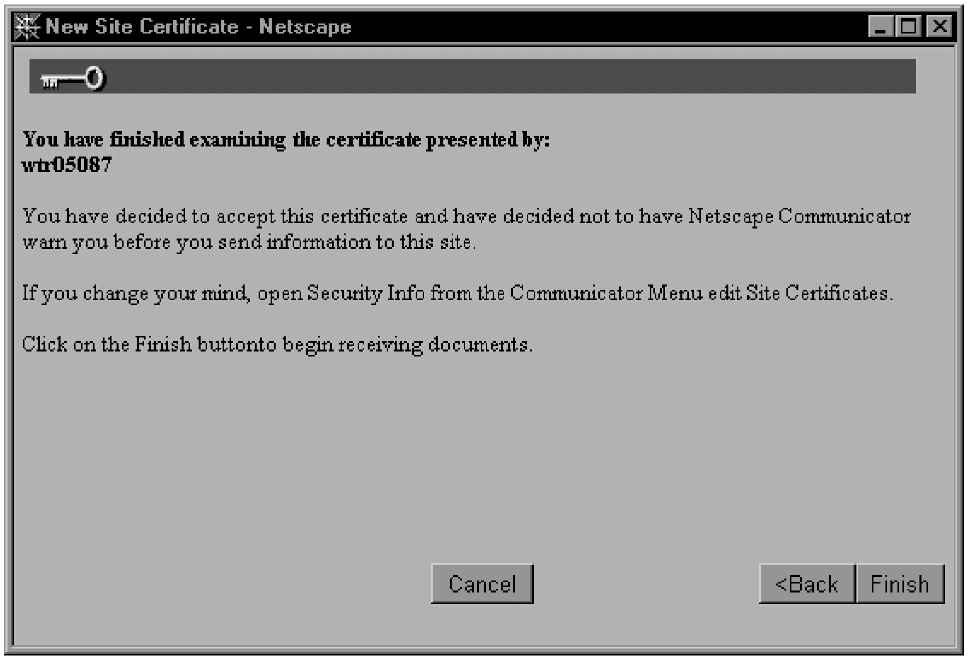
To protect yourself from fraud, do not send information (especially personal information, credit card numbers, or passwords) to this site if you have any doubt about the site's integrity.

For your own protection, Netscape can remind you of this at the appropriate time.

☐ Warn me before I send information to this site

Figure 74. Warning Message

At the end, when you press **Next**, Netscape verifies that the host name of the Web server corresponds to its IP address. If the host name and the IP address match, you will see a Confirmation message similar to the following.



You have finished examining the certificate presented by:
wtr05087

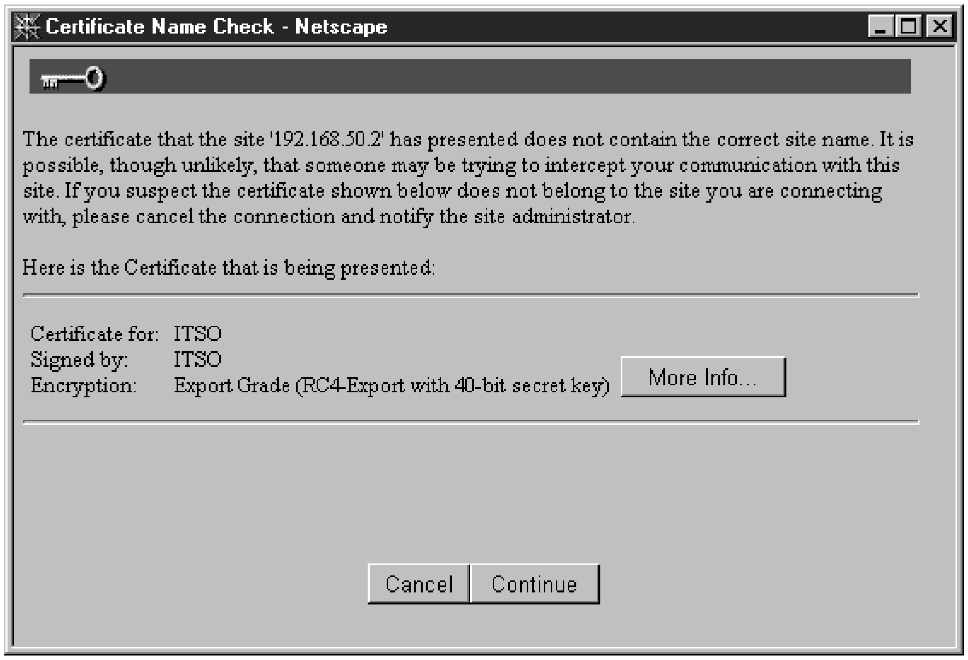
You have decided to accept this certificate and have decided not to have Netscape Communicator warn you before you send information to this site.

If you change your mind, open Security Info from the Communicator Menu edit Site Certificates.

Click on the Finish buttonto begin receiving documents.

Figure 75. Site Certificate Finished

We also wanted to experiment with the case where the host name and the IP address of the Web server do not match. Netscape Communicator displays another warning message, giving you a further possibility to refuse the new certificate.



The certificate that the site '192.168.50.2' has presented does not contain the correct site name. It is possible, though unlikely, that someone may be trying to intercept your communication with this site. If you suspect the certificate shown below does not belong to the site you are connecting with, please cancel the connection and notify the site administrator.

Here is the Certificate that is being presented:

Certificate for:	ITSO
Signed by:	ITSO
Encryption:	Export Grade (RC4-Export with 40-bit secret key)

Figure 76. Certificate Name Check Provided by Netscape

We got this warning message by disabling the Domain Name Service (DNS) in our intranet. In this case Netscape Communicator was not able to compare IP address and host name. This is a common situation. In fact, in many intranets, DNS lookup for external hosts is disabled. The client browser informs the user that host name and IP address do not match through a panel like that shown in Figure 76. Keep in mind, however, that if the host name and the IP address do not match, it is also possible that a spoofing or Man-In-the-Middle (MIM) attack is running, meaning that someone is trying to intercept your communication with that Web site.

Finally, if you accept the certificate, the SSL session is established and you get the Web page you had requested using the SSL protocol.

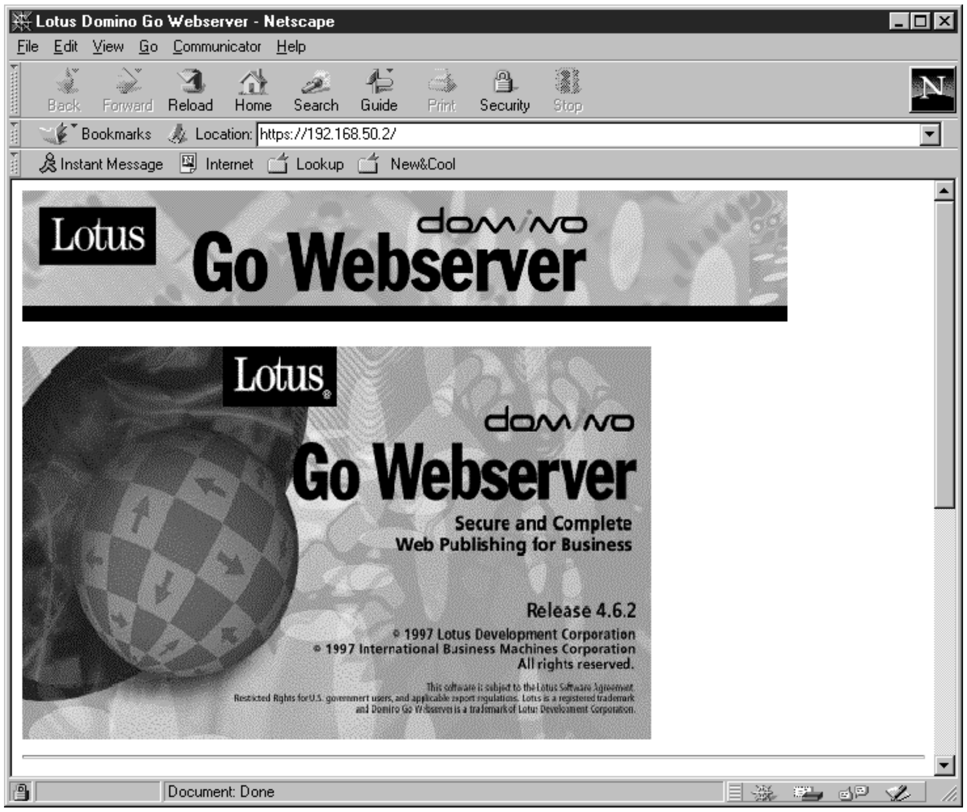

Lotus Domino Go Webserver - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: https://192.168.50.2/
Instant Message Internet Lookup New&Cool
Lotus
domino
Go Webserver
Secure and Complete
Web Publishing for Business
Release 4.6.2
© 1997 Lotus Development Corporation
© 1997 International Business Machines Corporation
All rights reserved.
Document: Done

Figure 77. SSL Communication

In this case, it was the same Web page that you normally use to access the Configuration and Administration Forms for your Web server. Notice the presence of a closed lock at the bottom left corner. It indicates that an SSL secure connection is active.

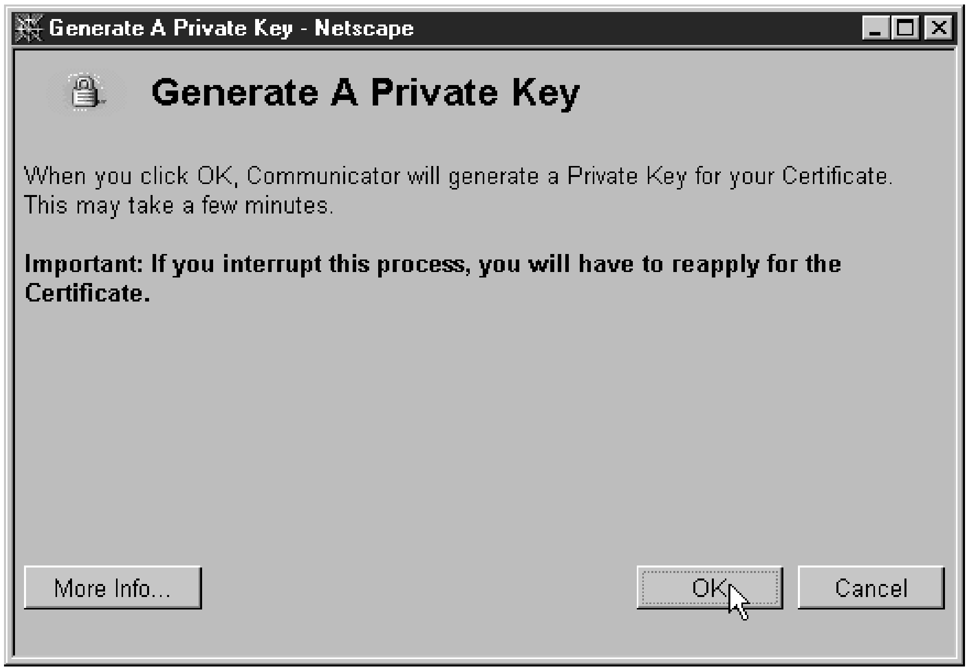
3.1.4 SSL Client Authentication

In 3.1.3, “Lotus Domino Go Webserver SSL Server Authentication” on page 79, we have indicated how one can set up Lotus Domino Go Webserver for SSL, using the basic server authentication. We now present to you the steps we took to incorporate client authentication. In this case, the clients are also issued certificates (not necessarily by the same CA who issued the certificate for the server) and this certificate is used during the SSL handshake. This enables the server to ascertain the identity of the client. For this reason, the server should first trust the CA who issued the certificate to the client.

We could easily use a client certificate obtained from VeriSign (see 5.4.1, “Obtaining a VeriSign Evaluation Certificate” on page 232). It would work very well with Domino Go Webserver, since VeriSign's self-signed certificates are already pre-installed in Lotus Domino Go Webserver. However we want to show you what happens if the client certificate is signed by a CA whose self-signed certificate is not installed in Domino Go Webserver or, in other words, if the CA that signed the client certificate is not considered trusted by the Web server. For this reason we did not want to use a client certificate issued by VeriSign.

Notice that we used Netscape Communicator 4.05 as a client. Things might be different with Microsoft Internet Explorer, since these two browsers handle certificates in a different way. You might want to see http://www.microsoft.com/security.

The steps we followed are somewhat as follows. First, we went to the CA site to get the client certificate for our browser. The CA we chose is a demo CA set up by CryptSoft, and can be found at the URL http://www.cryptsoft.com/˜tjh/usercert.cgi. We entered the values in the mandatory (and in some optional) fields and pressed the **Generate Client Certificate Key** button. We got the following window from Netscape:



Generate A Private Key

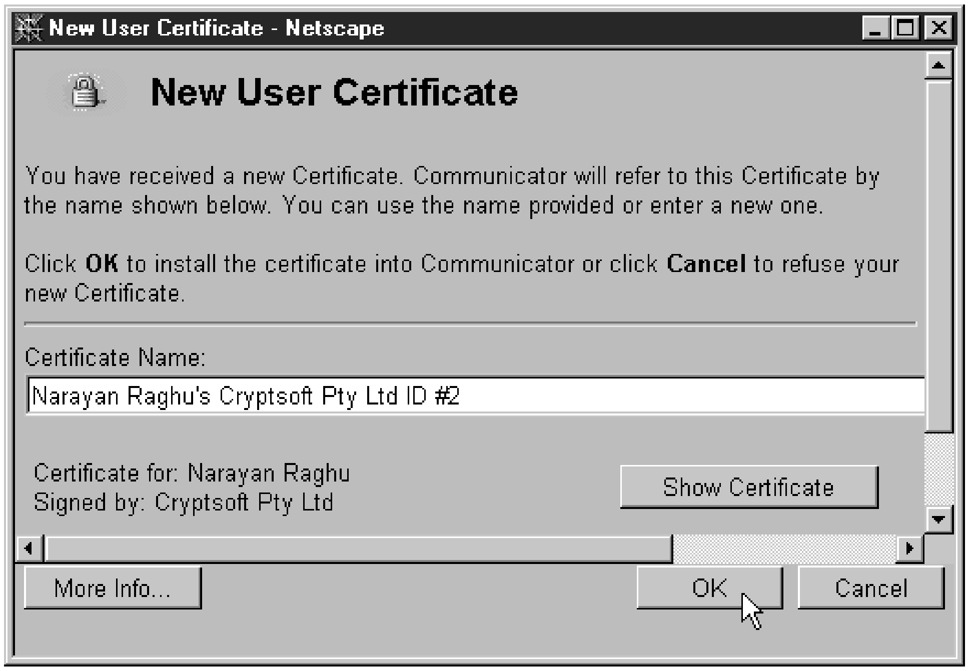
When you click OK, Communicator will generate a Private Key for your Certificate. This may take a few minutes.

Important: If you interrupt this process, you will have to reapply for the Certificate.

Figure 78. Generate A Private Key

We clicked on **OK** to continue the process, and we got a prompt for the password for the keys database of the Communicator. If you are generating keys for the first time since the installation of Communicator, you will not get this prompt. Instead, Communicator will ask you to choose a new password at the time of receiving the certificate. We entered the keys.db password, and the certificate was generated for us by the demo CA. In a real-life situation, it would never happen this fast. The CA would like to first ascertain if the information the client has provided is true or not before issuing the certificate. However, since we are using a demo CA, we get to save a lot of time and trouble, and collect the certificate as soon as we applied for it.

While the certificate is being written into the browser, the following window is shown by Netscape:



New User Certificate

You have received a new Certificate. Communicator will refer to this Certificate by the name shown below. You can use the name provided or enter a new one.

Click **OK** to install the certificate into Communicator or click **Cancel** to refuse your new Certificate.

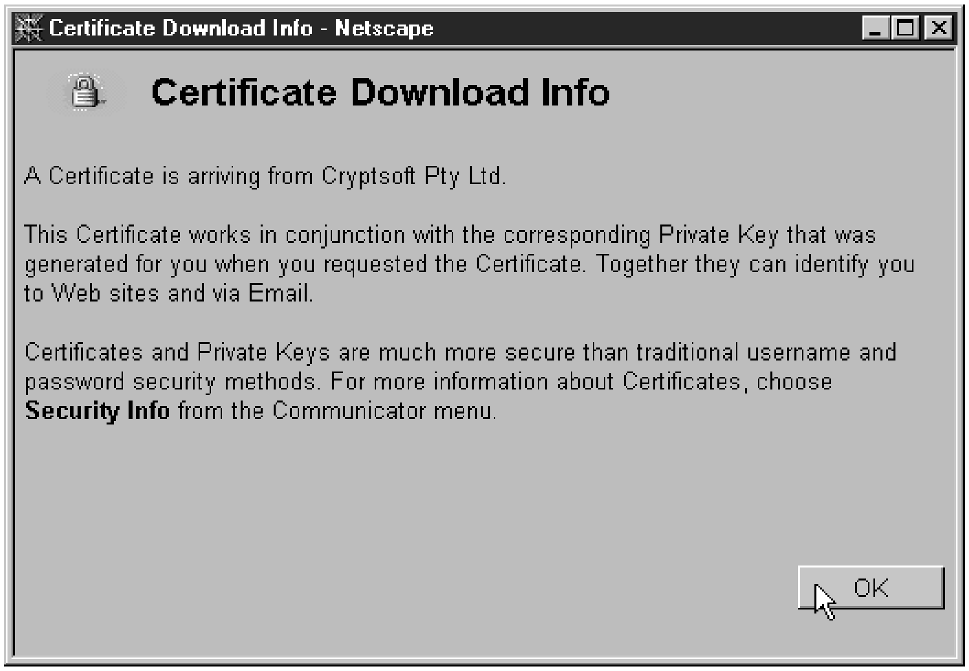
Certificate Name:

Narayan Raghu's Cryptsoft Pty Ltd ID #2

Certificate for: Narayan Raghu
Signed by: Cryptsoft Pty Ltd

Figure 79. Certificate Installation

As we have already mentioned, collecting a certificate is not likely to happen immediately after requesting it. Netscape provides an option to the user to view the information in the certificate before importing it into the certificates database of Netscape. To see this, click on **More Info**. On doing this, we saw the following screen:



Certificate Download Info

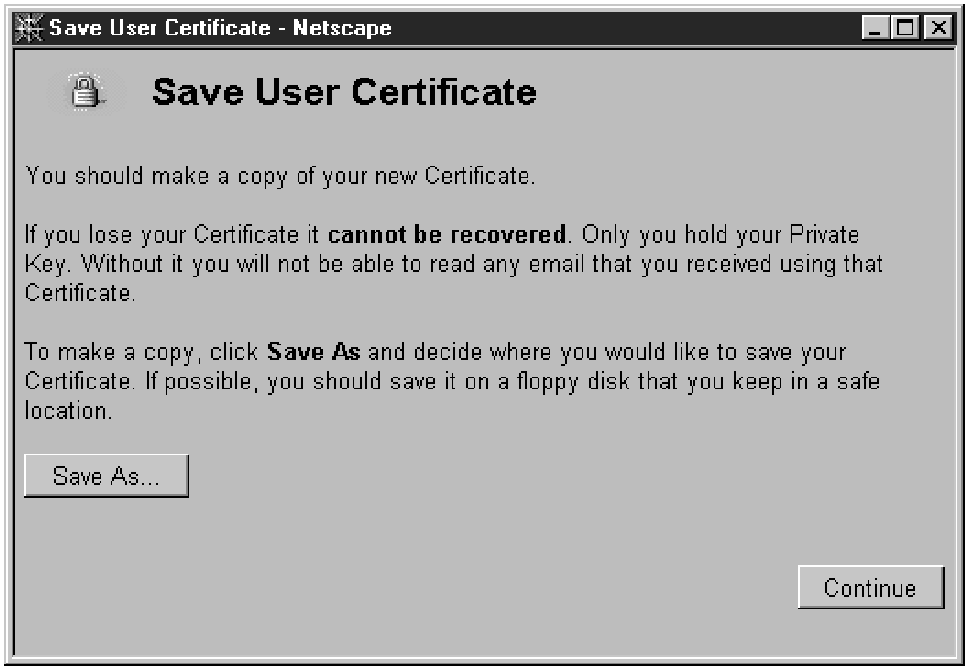
A Certificate is arriving from Cryptsoft Pty Ltd.

This Certificate works in conjunction with the corresponding Private Key that was generated for you when you requested the Certificate. Together they can identify you to Web sites and via Email.

Certificates and Private Keys are much more secure than traditional username and password security methods. For more information about Certificates, choose **Security Info** from the Communicator menu.

Figure 80. Certificate Information

After corroboration of the information, we clicked on **OK** to return back to the previous window. Here, we clicked on **OK** again since we wanted to import the certificate into our browser. The browser then popped up a window asking us if we wanted to make a copy of the certificate (along with the private key):



Save User Certificate

You should make a copy of your new Certificate.

If you lose your Certificate it **cannot be recovered**. Only you hold your Private Key. Without it you will not be able to read any email that you received using that Certificate.

To make a copy, click **Save As** and decide where you would like to save your Certificate. If possible, you should save it on a floppy disk that you keep in a safe location.

Figure 81. Save User Certificate

We chose not to, since this was only for trial. We clicked on **Continue**. The certificate was installed on our client. The CA server is really in no position to indicate to the client that a certificate has been installed, since the CA can only write into one stream at a time. Hence after the certificate has been installed, the confirmation of the installation of the certificate, which one would normally expect from the CA server, is given only by the client browser (by popping up the window in Figure 81) and not by the CA server. Hence, if you are using Netscape, you will find yourself in the same page even after installation of the certificate, and it might seem a bit awkward. This limitation is not present in MSIE 4.0, however, since it uses a complicated set of ActiveX components aided by VBScript to install certificates, and hence it has the option to point the browser to a new page if the certificate is successfully installed, and to another page if there was some foul-up.

That the certificate has indeed been installed can be ascertained by going to the menu in Netscape, and verifying manually. Hence, to complete the loop, we went to the Security Info page, as shown in the following screen:

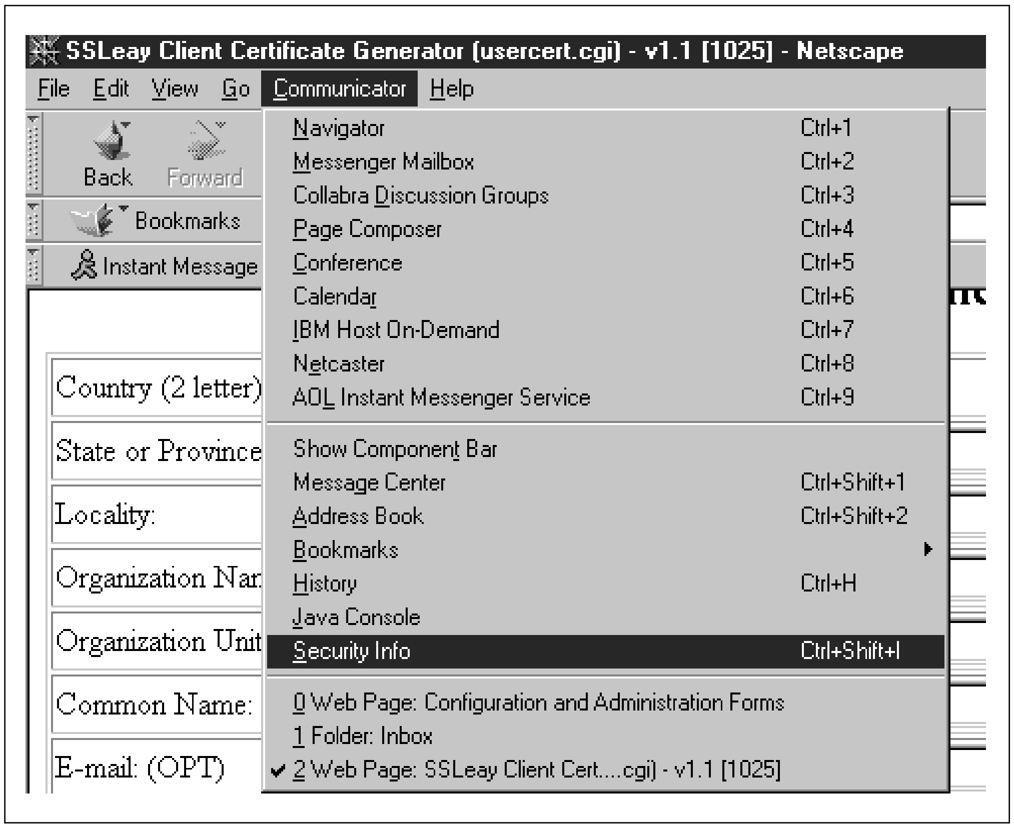
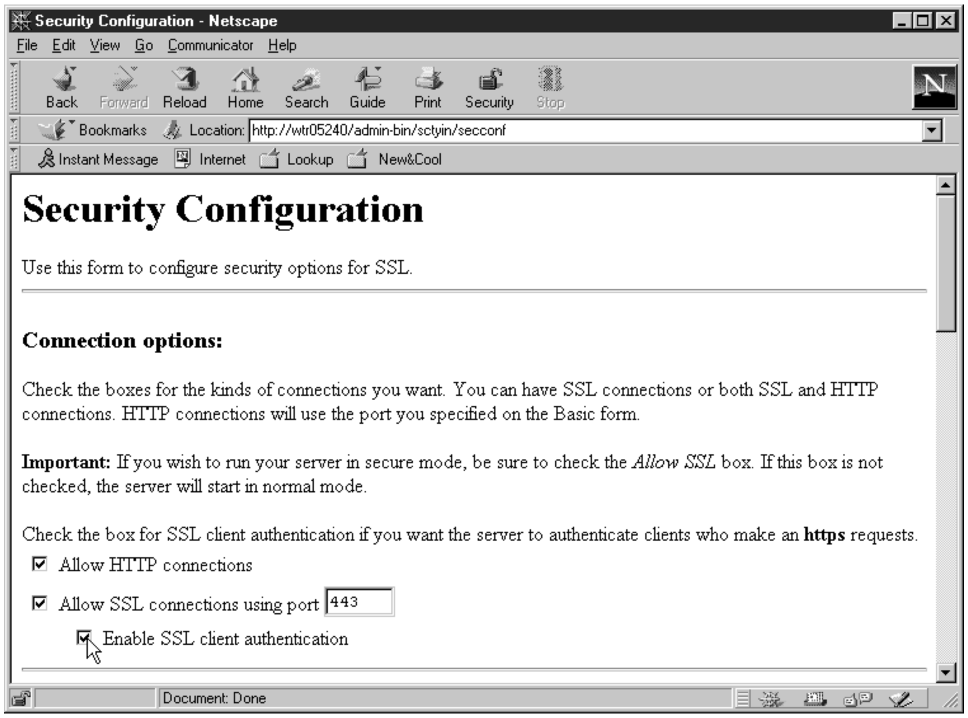

SSLeay Client Certificate Generator (usercert.cgi) - v1.1 [1025] - Netscape
File Edit View Go Communicator Help
Back Forward
Bookmarks
Instant Message
Navigator Ctrl+1
Messenger Mailbox Ctrl+2
Collabra Discussion Groups Ctrl+3
Page Composer Ctrl+4
Conference Ctrl+5
Calendar Ctrl+6
IBM Host On-Demand Ctrl+7
Netcaster Ctrl+8
AOL Instant Messenger Service Ctrl+9
Show Component Bar
Message Center Ctrl+Shift+1
Address Book Ctrl+Shift+2
Bookmarks
History Ctrl+H
Java Console
Security Info Ctrl+Shift+I
0 Web Page: Configuration and Administration Forms
1 Folder: Inbox
2 Web Page: SSLeay Client Cert....cgi) - v1.1 [1025]
Country (2 letter)
State or Province
Locality:
Organization Nar
Organization Unit
Common Name:
E-mail: (OPT)

Figure 82. Certificate Information

We clicked on **Certificates** and then on **Yours** and we saw a screen like that shown in Figure 227 on page 233, and thus ascertained that the certificate had been successfully installed.

Now that we had a client certificate, we tried to use client authentication. This is set up on the server side: one has to go to the Configuration and Administration Forms of Lotus Domino Go Webserver and click on **Security Configuration** in the Security section (see Figure 61 on page 72), check the **Enable SSL Client Authentication** check box, and restart the server:


Security Configuration - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/admin-bin/sctyin/secconf
Instant Message Internet Lookup New&Cool
Document: Done

Security Configuration

Use this form to configure security options for SSL.

Connection options:

Check the boxes for the kinds of connections you want. You can have SSL connections or both SSL and HTTP connections. HTTP connections will use the port you specified on the Basic form.

Important: If you wish to run your server in secure mode, be sure to check the *Allow SSL* box. If this box is not checked, the server will start in normal mode.

Check the box for SSL client authentication if you want the server to authenticate clients who make an **https** requests.

☑ Allow HTTP connections

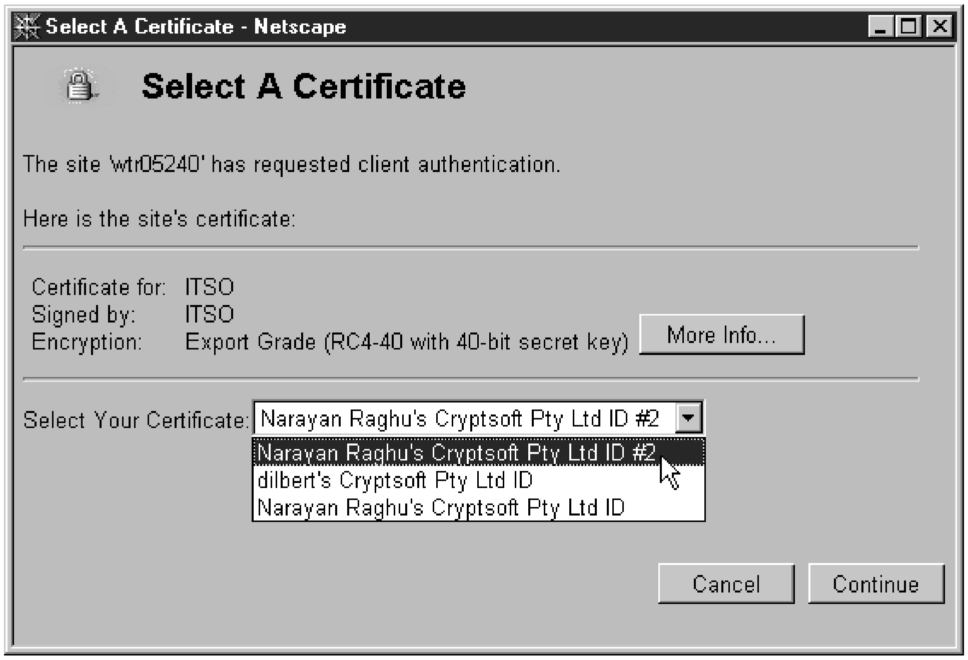
☑ Allow SSL connections using port 443

☑ Enable SSL client authentication

Figure 83. Certificate Information

The rest of the options given on that page are of no immediate concern to us, and we can safely let them be. Now that the client authentication is enabled on the server, we tried accessing the server with the Netscape client which we had just got certified. The connection began well, with the server first presenting its self-signed certificate to us as shown in Figure 70 on page 80 through Figure 76 on page 83.

After we choose to trust the server, the server requests the client certificate, as shown. The next window to be popped up tells us that the server has asked the client to present its certificate to it. We had three certificates to choose from, and we chose the one we just installed.


Select A Certificate - Netscape

Select A Certificate

The site 'wtr05240' has requested client authentication.

Here is the site's certificate:

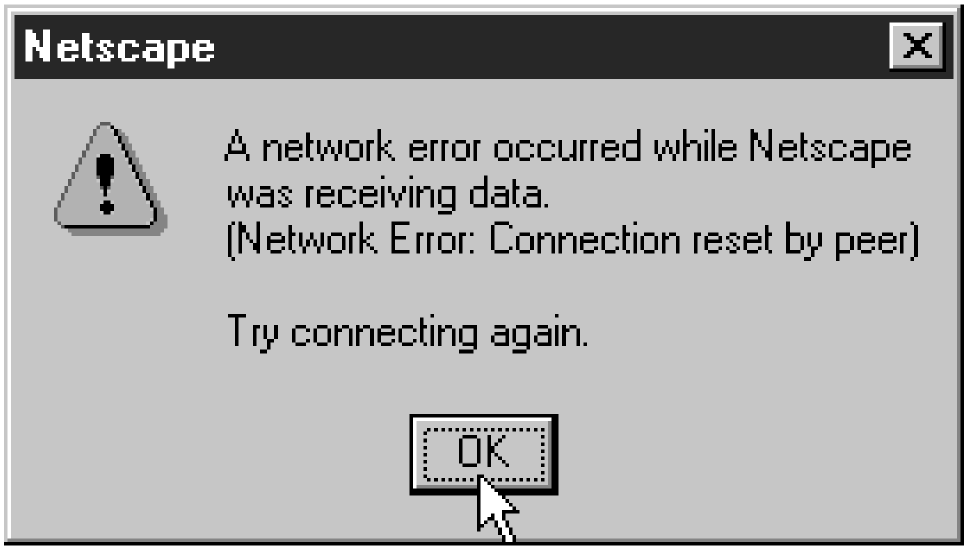
Certificate for:	ITSO
Signed by:	ITSO
Encryption:	Export Grade (RC4-40 with 40-bit secret key)

Select Your Certificate: Narayan Raghu's Cryptsoft Pty Ltd ID #2

- Narayan Raghu's Cryptsoft Pty Ltd ID #2
- dilbert's Cryptsoft Pty Ltd ID
- Narayan Raghu's Cryptsoft Pty Ltd ID

Figure 84. Select a Certificate

But on doing this, we got an error message. As you can see, the error message was not very indicative of the problem found.


Netscape

A network error occurred while Netscape was receiving data.
(Network Error: Connection reset by peer)

Try connecting again.

Figure 85. Netscape Error

Looking at the error log file for Lotus Domino Go Webserver, we found that it mentioned that the SSL handshake had failed. The reason for this was that we had not designated the CA that signed the client certificate as a trusted root.

Now, how is one supposed to do this? First, we have to obtain the self-signed certificate from the CA. This could be done in many ways, including ftp and e-mail. Note that VeriSign's self-signed CA certificates are pre-installed in Domino Go Webserver. Hence, had we used VeriSign to issue a certificate for our client, we would not have had to go through the following few steps. As we have already said, we deliberately chose a demo CA so that we could illustrate installing the CA's certificate into the server.

Domino Go Webserver expects the CA certificate to be in the Privacy Enhanced Message (PEM) format. The certificate can then be imported into the working key ring file, and all the clients certified by the particular CA (to be precise, by the particular private key corresponding to the public key on the certificate) will be trusted.

Lotus Domino Go Webserver 30-Day Trial Version

If you are working on the 30-day trial version of Domino Go, you might have the little problem that it trusts just about all client certificates regardless of who issued it. However, in the actual purchased version, this defect no longer exists.

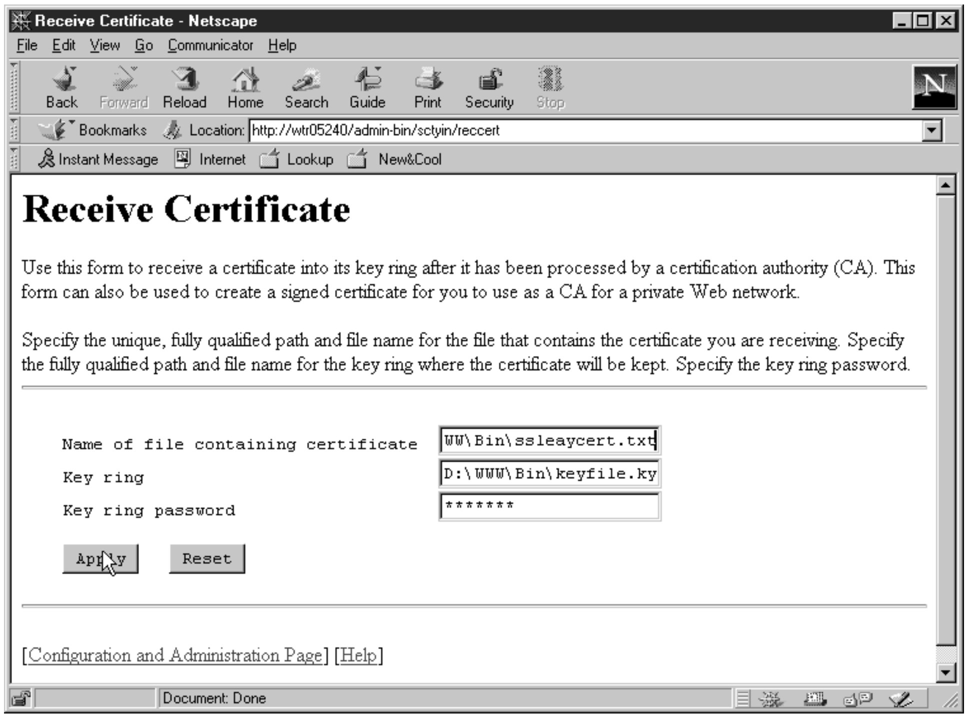
On the Demo CA page, we chose the option to give us a PEM format of the CA certificate, and clicked on **Get CA Cert** to get the CA certificate in the PEM format. The certificate is formatted as shown in Figure 86, with the dotted lines indicating the start and end of the certificate itself. This is how the CA certificate in PEM format would look.

```
-----BEGIN CERTIFICATE-----
MIICEDCCAboCAQMwDQYJKoZIhvcNAQEEBQAwgZIxCzAJBgNVBAYTAkFVMRMwEQYD
VQQIEwpRdWVlbnNsYW5kMREwDwYDVQQHEwhCcmlzYmFuZTEaMBgGA1UEChMRQ3J5
cHRzb2Z0IFB0eSBMdGQxIjAgBgNVBAsTGURFTU9OU1RSQVRJT04gQU5EIFRFU1RJ
TkcxGzAZBgNVBAMTEkRFTU8gWkVSTyBWQUxVRSBDQTAeFw05ODAzMDMwNzQxMzJa
Fw0wODAyMjkwNzQxMzJaMIGSMQswCQYDVQQGEwJBVTETMBEGA1UECBMKUXVlZW5z
bGFuZDERMA8GA1UEBxMIQnJpc2JhbmUxGjAYBgNVBAoTEUNyeXB0c29mdCBQdHkg
THRkMSIwIAYDVQQLExlERU1PTlNUUkFUSU9OIEFORCBURVNUSU5HMRswGQYDVQQD
ExJERU1PIFpFUk8gVkFMVUUgQ0EwXDANBgkqhkiG9w0BAQEFAANLADBIAkEAv7QT
Z8t6PcVILB7YDaJGcy4LfPXaocinorLXCjbNlzF+3sAj5F+StatIa1+Z/MES4+st
1+6WdwLljWk1wzxFNQIDAQABMA0GCSqGSIb3DQEBBAUAA0EAZJssymEvoXmmnezy
jRfKHzoUcV2lU7BjHT0QsuS4bXtX8/2mAhzQSj+AMxs34pPch2AQJ/xL/+kJEDpG
s2Nxcg==
-----END CERTIFICATE-----
```

Figure 86. CA Certificate in PEM Format

This we copy-pasted onto a notepad, and saved it as ssleaycert.txt in the D:\WWW\bin directory on the Web server machine.

We then went to the Configuration and Administration Forms, and chose **Receive Certificate** which led us to the following page:


Receive Certificate - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/admin-bin/sctyin/reccert
Instant Message Internet Lookup New&Cool

Receive Certificate

Use this form to receive a certificate into its key ring after it has been processed by a certification authority (CA). This form can also be used to create a signed certificate for you to use as a CA for a private Web network.

Specify the unique, fully qualified path and file name for the file that contains the certificate you are receiving. Specify the fully qualified path and file name for the key ring where the certificate will be kept. Specify the key ring password.

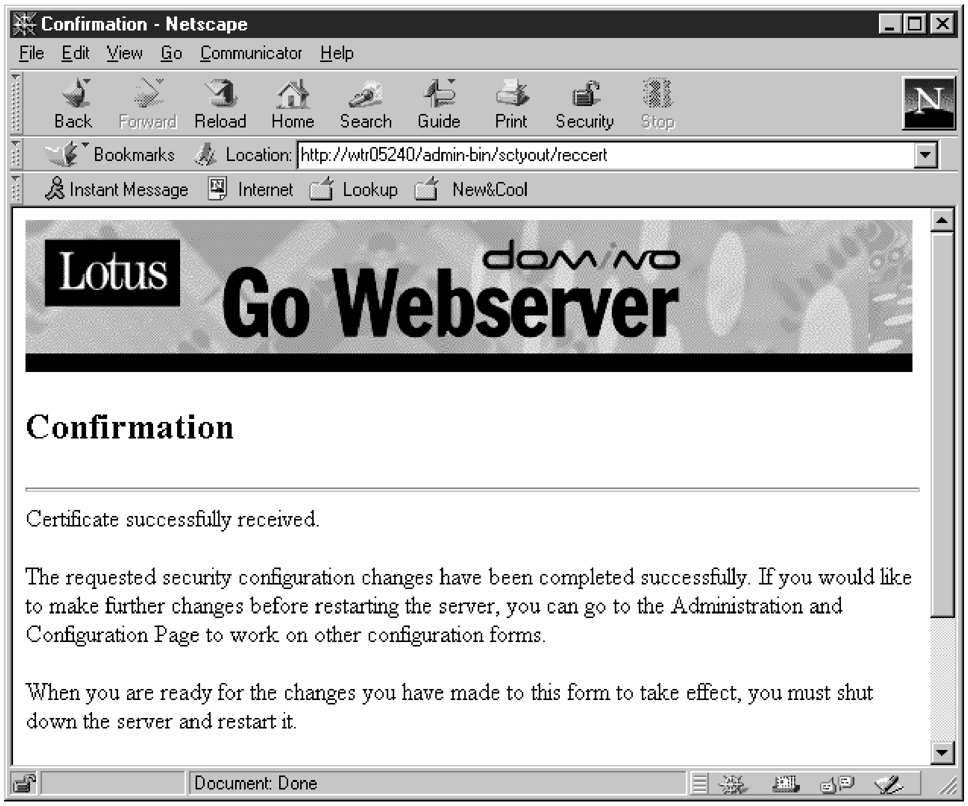
Name of file containing certificate	WW\Bin\ssleaycert.txt
Key ring	D:\WWW\Bin\keyfile.ky
Key ring password	*******

Figure 87. Import Certificate

Note that we received the CA certificate into the working key ring. The CA certificate has to be present in the working key ring. Our working key ring was D:\WWW\keyfile.kyr and so we imported the CA certificate into that one.

It is important for the CA Certificate to be in the working key ring. Actually all the certificates and key pairs needed for the SSL handshake including the server's key pair are to be present in the working key ring. The administrator of Domino Go Webserver could choose to change the working key ring, say, to enable the server to use a different server certificate (probably issued by a different CA) for SSL communications. The working key ring can be changed by selecting **Security Configuration** as shown in Figure 61 on page 72.

On clicking on **Apply**, a confirmation message was obtained, indicating that the certificate had been successfully imported into the key ring.


Confirmation - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/admin-bin/sctyout/reccert
Instant Message Internet Lookup New&Cool
Lotus
domino
Go Webserver
Document: Done

Confirmation

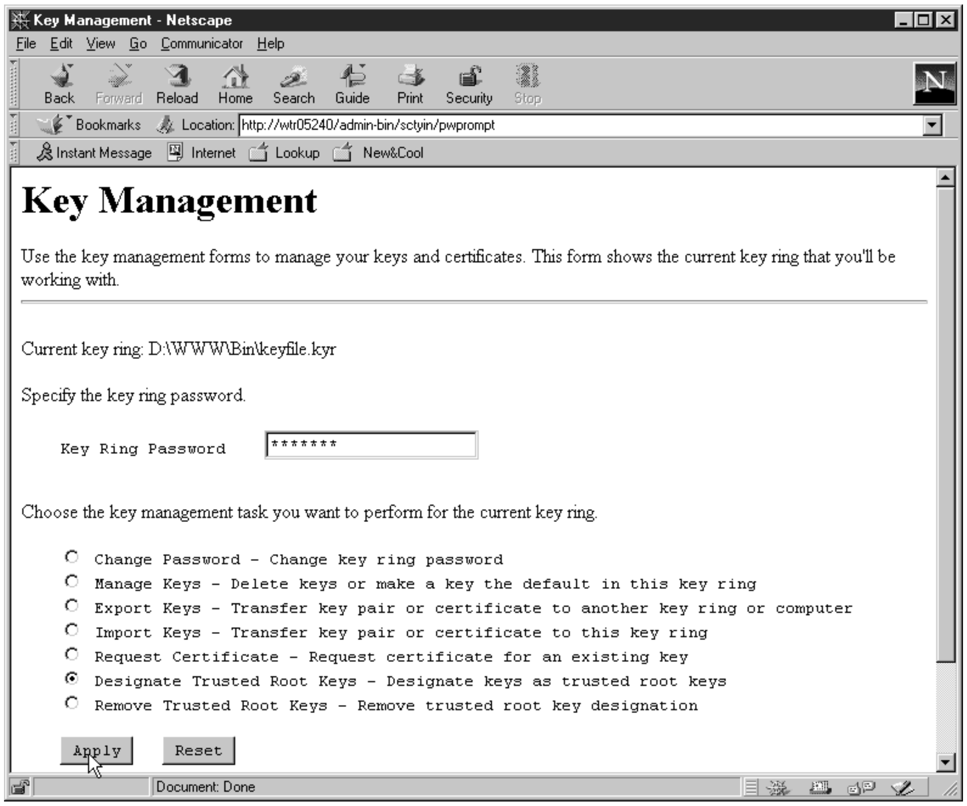
Certificate successfully received.

The requested security configuration changes have been completed successfully. If you would like to make further changes before restarting the server, you can go to the Administration and Configuration Page to work on other configuration forms.

When you are ready for the changes you have made to this form to take effect, you must shut down the server and restart it.

Figure 88. Certificate Successfully Imported

That having been done, the only thing left was to designate it as a trusted root. For that, we went to the Configuration and Administration Forms again and chose **Key Management**. In the page we got, we keyed in the Key Ring Password and opted to designate a certificate as trusted root, using the radio button.


Key Management - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/admin-bin/sctyin/pwprompt
Instant Message Internet Lookup New&Cool

Key Management

Use the key management forms to manage your keys and certificates. This form shows the current key ring that you'll be working with.

Current key ring: D:\WWW\Bin\keyfile.kyr

Specify the key ring password.

Key Ring Password: *******

Choose the key management task you want to perform for the current key ring.

```
○ Change Password - Change key ring password
○ Manage Keys - Delete keys or make a key the default in this key ring
○ Export Keys - Transfer key pair or certificate to another key ring or computer
○ Import Keys - Transfer key pair or certificate to this key ring
○ Request Certificate - Request certificate for an existing key
● Designate Trusted Root Keys - Designate keys as trusted root keys
○ Remove Trusted Root Keys - Remove trusted root key designation
```

Figure 89. Key Management

We then clicked **Apply**. On the next page, we selected the certificate and clicked on **Apply** again, to designate it to be trusted root. We got a confirmation for the action.

Now we tried accessing the Web server using the `https://` URL, and the request went through. Just a reminder: if you want to go back to the old state, you could go to the Key Management page from the Configuration and Administration Forms home page, and opt to **Remove Trusted Root Keys**. This will lead you to a page containing all the keys that are currently trusted. You could choose to distrust any number of them - one at a time. Also note that in your trusted CAs list will be a series of VeriSign certificates you have not declared as trusted. These have been put there as a default, and you would do well to remove them from the trusted list since a couple of them are for signing trial or demo certificates.

3.2 Examples of Security Using HTTP and SSL

In this section we will illustrate the differences between the two HTTP methods, GET and POST, that are commonly used by forms to handle the information sent to the server. We will also see the security implications of how GET and POST perform over plain HTTP and over SSL.

To do this, we wrote a very simple servlet that welcomes the user, with the entered user name. This servlet is invoked from a simple HTML page, alternately using the GET and the POST methods. The HTML code is as shown:

```
<HTML>

<HEAD>
<TITLE> The Get Method </TITLE>
</HEAD>

<BODY>

<BR>
<CENTER><h2> Please enter your particulars </h2></CENTER>

<FORM Action="/servlet/EchoServlet" Method="GET">
<PRE>
User Name : <INPUT Type="TEXT" Name="userid">
Password  : <INPUT Type="password" Name="passwd">
<INPUT Type="SUBMIT">
</PRE>
</FORM>

</BODY>

</HTML>
```

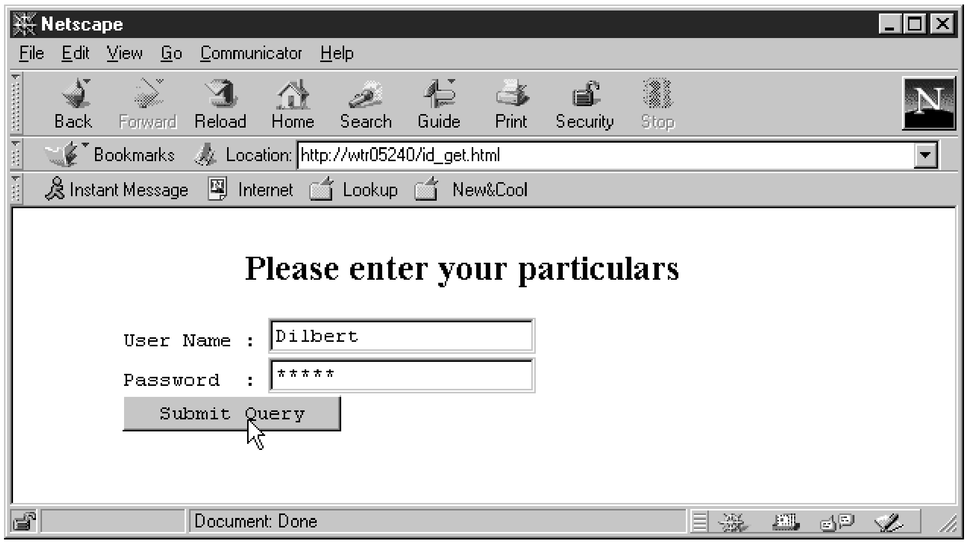
Figure 90. The id_get.html File

When the GET method is used, the filled-in form input variable names and their values are sent to the server by simply appending them to the URL of the next request. We show here the source code of the servlet that receives the form values from the HTML page:

```
import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class EchoServlet extends HttpServlet
{

  public void init(ServletConfig conf) throws ServletException
  {
  super.init(conf);
  log("Echo Server Initialized");
  }

  public void service(HttpServletRequest req, HttpServletResponse res)
    throws IOException, ServletException
  {
    String userid = req.getParameter("userid");
    ServletOutputStream ops = res.getOutputStream();
    res.setContentType("text/html");

    ops.println("<HTML><HEAD><TITLE>Generated by a Servlet</TITLE></HEAD>");
    ops.println("<BODY>");
    ops.println("<CENTER><H1> Hello </H1>");
    ops.println("Welcome, " + userid + " how are you today");
    ops.println("<BR>");
    ops.println("</BODY> </HTML>");
    ops.close();
  }
}
```

Figure 91. The EchoServlet Source Code

The following picture shows the HTML page in a browser, after entering the User Name and Password:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/id_get.html
Instant Message Internet Lookup New&Cool
Document: Done

Please enter your particulars

User Name : Dilbert

Password : *****

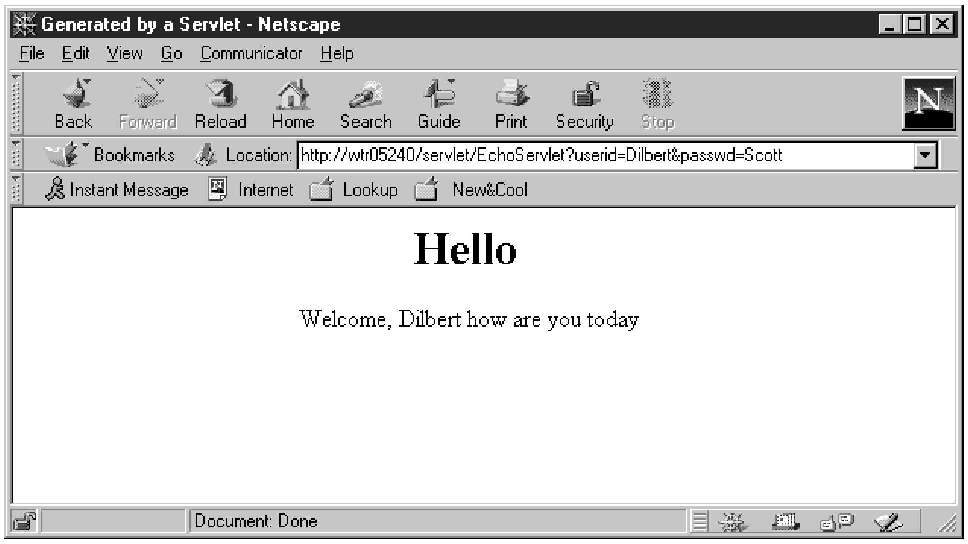
Submit Query

Figure 92. User ID and Password

Note that what you type in the password field appears hidden by a sequence of asterisks, rather than the characters you typed in. This happens to grant privacy and security while you type your password, so that people sitting near you cannot read what you enter. Actually, the asterisks grant only the appearance of privacy and security, as we are going to demonstrate.

If you read through the servlet code, you'd see that the servlet will simply welcome the person using the User Name entered, and will do nothing with the password. We added the password field here to show how confidential information flows between client and server when a form uses GET or POST through HTTP or SSL. The two fields, the User Name and Password, have been used to represent public data (that you wouldn't mind people reading) and private data (confidential data, which you would not like any third-party members reading - such as your passwords or credit card number). We have just shown you how apparently, on the screen of the client machine, confidential information is treated differently from public data.

Now let's see what happens when we click on the **Submit Query** button. The information you keyed in is passed along with the URL, and this is visible in the URL field in your browser. The password `Scott` too is visible as plain text. This means that we have completely lost the privacy that was granted by the sequence of asterisks.


Generated by a Servlet - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location:
Instant Message Internet Lookup New&Cool
Document: Done

http://wtr05240/servlet/EchoServlet?userid=Dilbert&passwd=Scott

Hello

Welcome, Dilbert how are you today

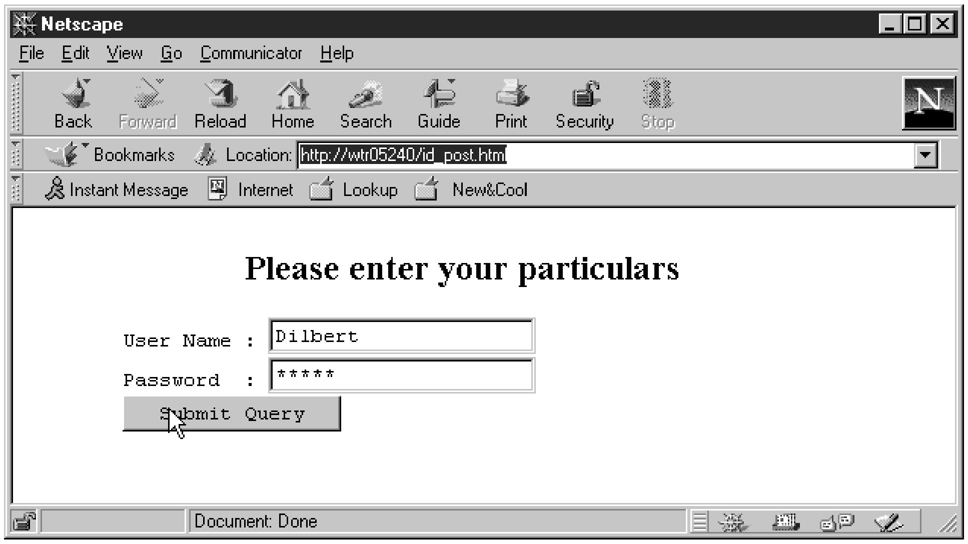
Figure 93. Confidenatial Information Displayed on the URL Field

Now, let's try doing the same with the method POST. The servlet code remains the same. In the HTML page, let us simply change the method to POST. Note that this would not be possible with simple CGI - you would have to make some modifications in the CGI script and recompile, unless you have taken care to put an `if` condition in the script to check what method is used.

```
<HTML>

<HEAD>
<TITLE> The POST method </TITLE>
</HEAD>

<BODY>

<BR>
<CENTER><h2> Please enter your particulars </h2></CENTER>

<FORM Action="/servlet/EchoServlet" Method="POST">
<PRE>
User Name : <INPUT Type="TEXT" Name="userid">
Password  : <INPUT Type="password" Name="passwd">
<INPUT Type="SUBMIT">
</PRE>
</FORM>

</BODY>

</HTML>
```

Figure 94. The id_post.html File

We can key in the same information, and here's what we get.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/id_post.html
Instant Message Internet Lookup New&Cool

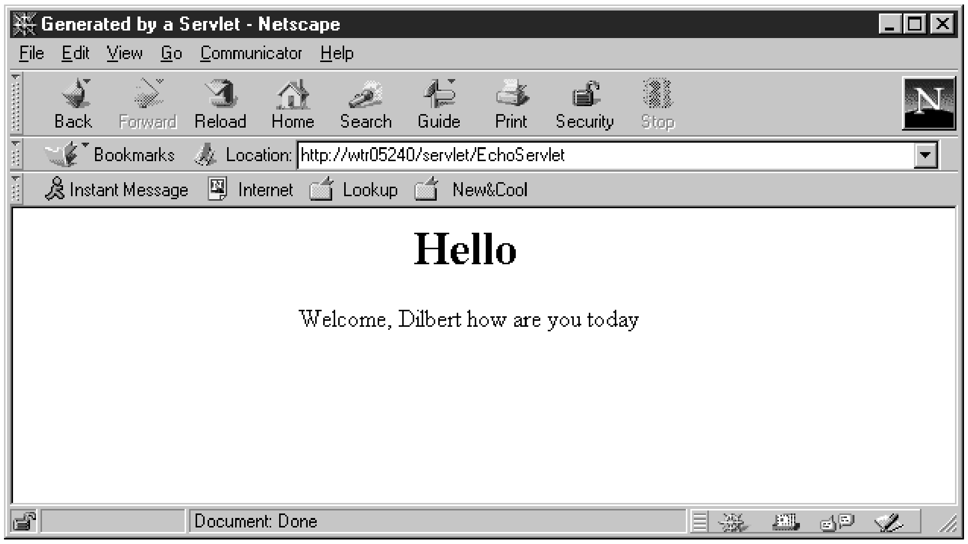
Please enter your particulars

User Name : Dilbert
Password : *****

Submit Query
Document: Done

Figure 95. Using the POST Method

Then we click the **Submit Query** button.


Generated by a Servlet - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/servlet/EchoServlet
Instant Message Internet Lookup New&Cool
Document: Done

Hello

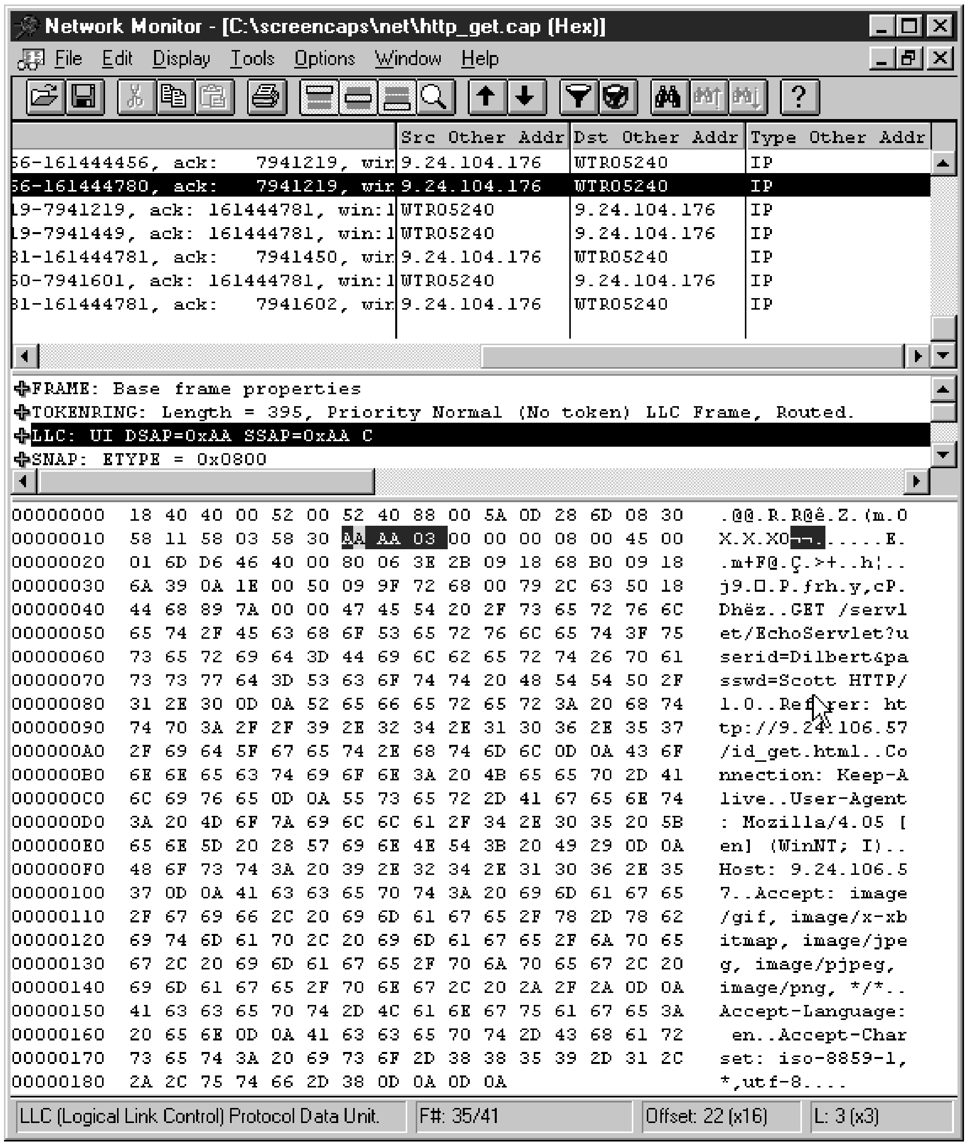
Welcome, Dilbert how are you today

Figure 96. After Using the POST Method

Notice how the passed fields are *not* visible, this time, as a part of the URL (or anything else). This happens with the POST method because the form variable names and values are sent in the HTTP request body. This implies that they are not shown by the browser as part of the URL, which, instead, is sent in the HTTP request header. If the GET method is used, the URL and the data both go in the HTTP request header. This demonstrates that at least from this point of view the POST method grants more privacy and security.

What if we try to intercept the submitted data when in transit? We tried that too. Unless encrypted (by using, say, SSL) the data can be intercepted in transit irrespective of the method type used for the form.

Figure 97 on page 100 shows the information we obtained from the Microsoft Systems Management Server (SMS) Network Monitor Version 4.00, while the GET method was used. This package, which we installed on a third machine running Windows NT Server 4.0, can capture and display data being transmitted between two other machines within the same LAN segment where the server running SMS Network Monitor is attached.



```
                                         Src Other Addr  Dst Other Addr  Type Other Addr
56-161444456, ack:   7941219, wir 9.24.104.176    WTR05240        IP
56-161444780, ack:   7941219, wir 9.24.104.176    WTR05240        IP
19-7941219, ack: 161444781, win:1 WTR05240        9.24.104.176    IP
19-7941449, ack: 161444781, win:1 WTR05240        9.24.104.176    IP
81-161444781, ack:   7941450, wir 9.24.104.176    WTR05240        IP
50-7941601, ack: 161444781, win:1 WTR05240        9.24.104.176    IP
81-161444781, ack:   7941602, wir 9.24.104.176    WTR05240        IP
```

```
FRAME: Base frame properties
TOKENRING: Length = 395, Priority Normal (No token) LLC Frame, Routed.
LLC: UI DSAP=0xAA SSAP=0xAA C
SNAP: ETYPE = 0x0800
```

```
00000000  18 40 40 00 52 00 52 40 88 00 5A 0D 28 6D 08 30   .@@.R.R@ê.Z.(m.0
00000010  58 11 58 03 58 30 AA AA 03 00 00 00 08 00 45 00   X.X.X0¬¬......E.
00000020  01 6D D6 46 40 00 80 06 3E 2B 09 18 68 B0 09 18   .m+F@.Ç.>+..h¦..
00000030  6A 39 0A 1E 00 50 09 9F 72 68 00 79 2C 63 50 18   j9.□.P.ƒrh.y,cP.
00000040  44 68 89 7A 00 00 47 45 54 20 2F 73 65 72 76 6C   Dhëz..GET /servl
00000050  65 74 2F 45 63 68 6F 53 65 72 76 6C 65 74 3F 75   et/EchoServlet?u
00000060  73 65 72 69 64 3D 44 69 6C 62 65 72 74 26 70 61   serid=Dilbert&pa
00000070  73 73 77 64 3D 53 63 6F 74 74 20 48 54 54 50 2F   sswd=Scott HTTP/
00000080  31 2E 30 0D 0A 52 65 66 65 72 65 72 3A 20 68 74   1.0..Referer: ht
00000090  74 70 3A 2F 2F 39 2E 32 34 2E 31 30 36 2E 35 37   tp://9.24.106.57
000000A0  2F 69 64 5F 67 65 74 2E 68 74 6D 6C 0D 0A 43 6F   /id_get.html..Co
000000B0  6E 6E 65 63 74 69 6F 6E 3A 20 4B 65 65 70 2D 41   nnection: Keep-A
000000C0  6C 69 76 65 0D 0A 55 73 65 72 2D 41 67 65 6E 74   live..User-Agent
000000D0  3A 20 4D 6F 7A 69 6C 6C 61 2F 34 2E 30 35 20 5B   : Mozilla/4.05 [
000000E0  65 6E 5D 20 28 57 69 6E 4E 54 3B 20 49 29 0D 0A   en] (WinNT; I)..
000000F0  48 6F 73 74 3A 20 39 2E 32 34 2E 31 30 36 2E 35   Host: 9.24.106.5
00000100  37 0D 0A 41 63 63 65 70 74 3A 20 69 6D 61 67 65   7..Accept: image
00000110  2F 67 69 66 2C 20 69 6D 61 67 65 2F 78 2D 78 62   /gif, image/x-xb
00000120  69 74 6D 61 70 2C 20 69 6D 61 67 65 2F 6A 70 65   itmap, image/jpe
00000130  67 2C 20 69 6D 61 67 65 2F 70 6A 70 65 67 2C 20   g, image/pjpeg,
00000140  69 6D 61 67 65 2F 70 6E 67 2C 20 2A 2F 2A 0D 0A   image/png, */*..
00000150  41 63 63 65 70 74 2D 4C 61 6E 67 75 61 67 65 3A   Accept-Language:
00000160  20 65 6E 0D 0A 41 63 63 65 70 74 2D 43 68 61 72    en..Accept-Char
00000170  73 65 74 3A 20 69 73 6F 2D 38 38 35 39 2D 31 2C   set: iso-8859-1,
00000180  2A 2C 75 74 66 2D 38 0D 0A 0D 0A                  *,utf-8....
```

```
LLC (Logical Link Control) Protocol Data Unit.   F#: 35/41   Offset: 22 (x16)   L: 3 (x3)
```

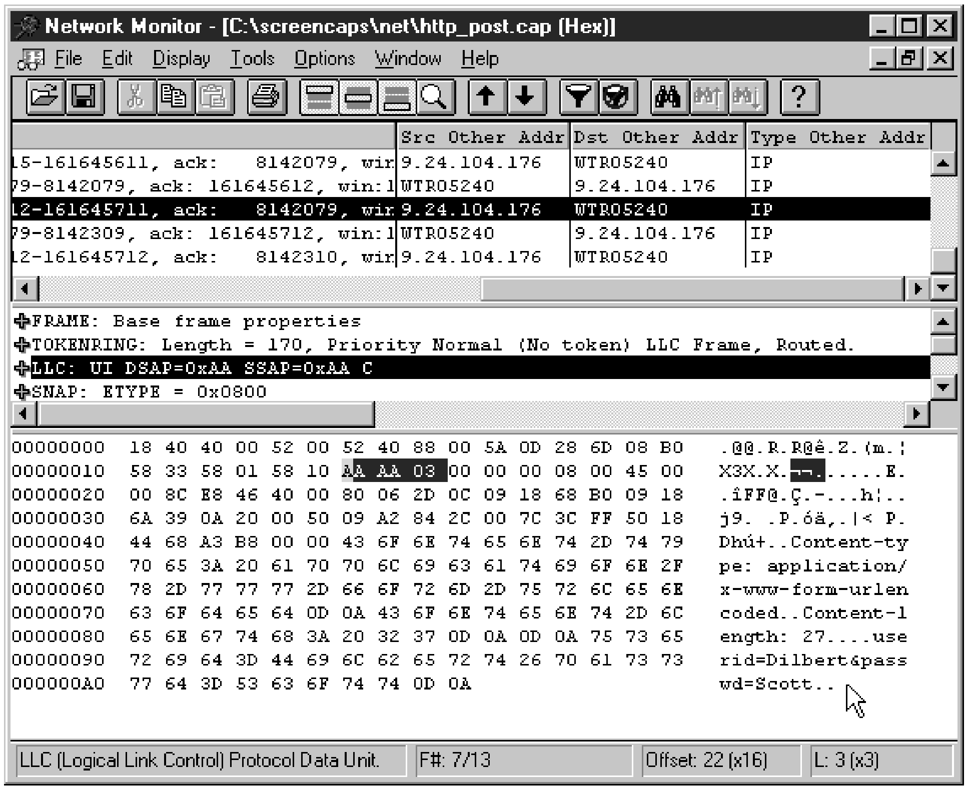
Figure 97. Data on Using the GET Method, at the Network Monitor

Note what the mouse cursor is pointing to. The user ID and password are clearly visible, as will be visible any other form data sent to the server.

Man-In-the-Middle Attacks

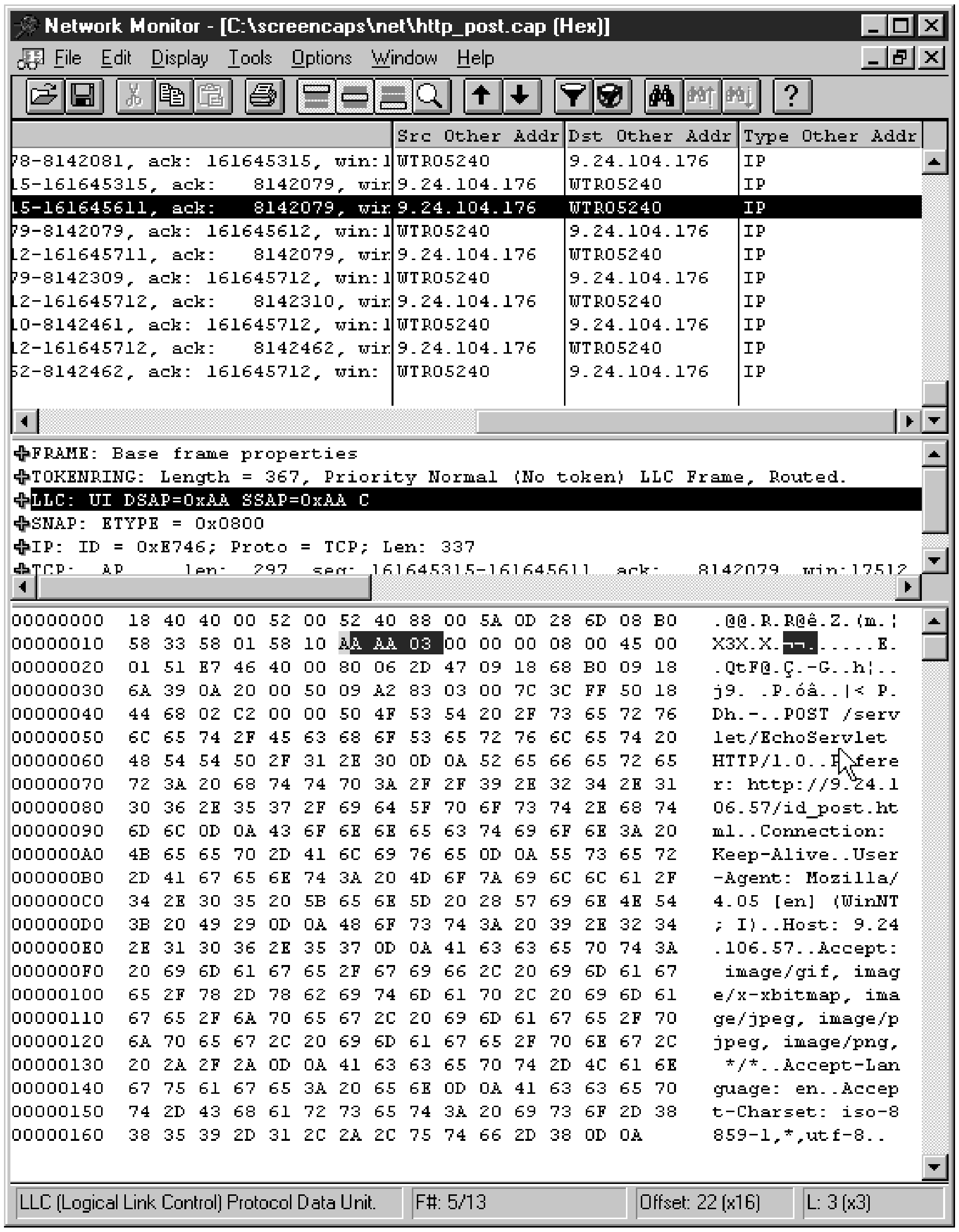
A network entity that intercepts data flowing between two machines is commonly known as Man-In-the-Middle (MIM). Notice that a MIM could have a more active role than just copying frames off the wire. A more dangerous MIM attack could be accomplished by a machine that actively inserts itself in the data flows between two legitimate systems in order to compromise the data flowing between them. To the client, the MIM masquerades as the server and to the server the MIM masquerades as the client. The MIM attack we are showing in this example is less active, since in this case the MIM is merely copying frames in transit, in order to get private information.

We again tried this with the POST method. As expected of an insecure channel, we were able to read inside the packets.


Network Monitor - [C:\screencaps\net\http_post.cap (Hex)]
File Edit Display Tools Options Window Help

```
                                        Src Other Addr Dst Other Addr Type Other Addr
l5-161645611, ack:   8142079, wir 9.24.104.176   WTR05240       IP
79-8142079, ack: 161645612, win:1 WTR05240       9.24.104.176   IP
l2-161645711, ack:   8142079, wir 9.24.104.176   WTR05240       IP
79-8142309, ack: 161645712, win:1 WTR05240       9.24.104.176   IP
l2-161645712, ack:   8142310, wir 9.24.104.176   WTR05240       IP

FRAME: Base frame properties
TOKENRING: Length = 170, Priority Normal (No token) LLC Frame, Routed.
LLC: UI DSAP=0xAA SSAP=0xAA C
SNAP: ETYPE = 0x0800

00000000  18 40 40 00 52 00 52 40 88 00 5A 0D 28 6D 08 B0   .@@.R.R@ê.Z.(m.¦
00000010  58 33 58 01 58 10 AA AA 03 00 00 00 08 00 45 00   X3X.X.¬¬.....E.
00000020  00 8C E8 46 40 00 80 06 2D 0C 09 18 68 B0 09 18   .îFF@.Ç.-...h¦..
00000030  6A 39 0A 20 00 50 09 A2 84 2C 00 7C 3C FF 50 18   j9. .P.óä,.|< P.
00000040  44 68 A3 B8 00 00 43 6F 6E 74 65 6E 74 2D 74 79   Dhú+..Content-ty
00000050  70 65 3A 20 61 70 70 6C 69 63 61 74 69 6F 6E 2F   pe: application/
00000060  78 2D 77 77 77 2D 66 6F 72 6D 2D 75 72 6C 65 6E   x-www-form-urlen
00000070  63 6F 64 65 64 0D 0A 43 6F 6E 74 65 6E 74 2D 6C   coded..Content-l
00000080  65 6E 67 74 68 3A 20 32 37 0D 0A 0D 0A 75 73 65   ength: 27....use
00000090  72 69 64 3D 44 69 6C 62 65 72 74 26 70 61 73 73   rid=Dilbert&pass
000000A0  77 64 3D 53 63 6F 74 74 0D 0A                     wd=Scott..

LLC (Logical Link Control) Protocol Data Unit.   F#: 7/13   Offset: 22 (x16)   L: 3 (x3)
```

Figure 98. Data on Using the POST Method, at the Network Monitor


Network Monitor - [C:\screencaps\net\http_post.cap (Hex)]
File Edit Display Tools Options Window Help

```
                                           Src Other Addr  Dst Other Addr  Type Other Addr
78-8142081, ack: 161645315, win:1 WTR05240        9.24.104.176    IP
15-161645315, ack:   8142079, wir 9.24.104.176    WTR05240        IP
15-161645611, ack:   8142079, wir 9.24.104.176    WTR05240        IP
79-8142079, ack: 161645612, win:1 WTR05240        9.24.104.176    IP
12-161645711, ack:   8142079, wir 9.24.104.176    WTR05240        IP
79-8142309, ack: 161645712, win:1 WTR05240        9.24.104.176    IP
12-161645712, ack:   8142310, wir 9.24.104.176    WTR05240        IP
10-8142461, ack: 161645712, win:1 WTR05240        9.24.104.176    IP
12-161645712, ack:   8142462, wir 9.24.104.176    WTR05240        IP
52-8142462, ack: 161645712, win:  WTR05240        9.24.104.176    IP
```

```
FRAME: Base frame properties
TOKENRING: Length = 367, Priority Normal (No token) LLC Frame, Routed.
LLC: UI DSAP=0xAA SSAP=0xAA C
SNAP: ETYPE = 0x0800
IP: ID = 0xE746; Proto = TCP; Len: 337
TCP: .AP...., len:  297, seq: 161645315-161645611, ack:   8142079, win:17512
```

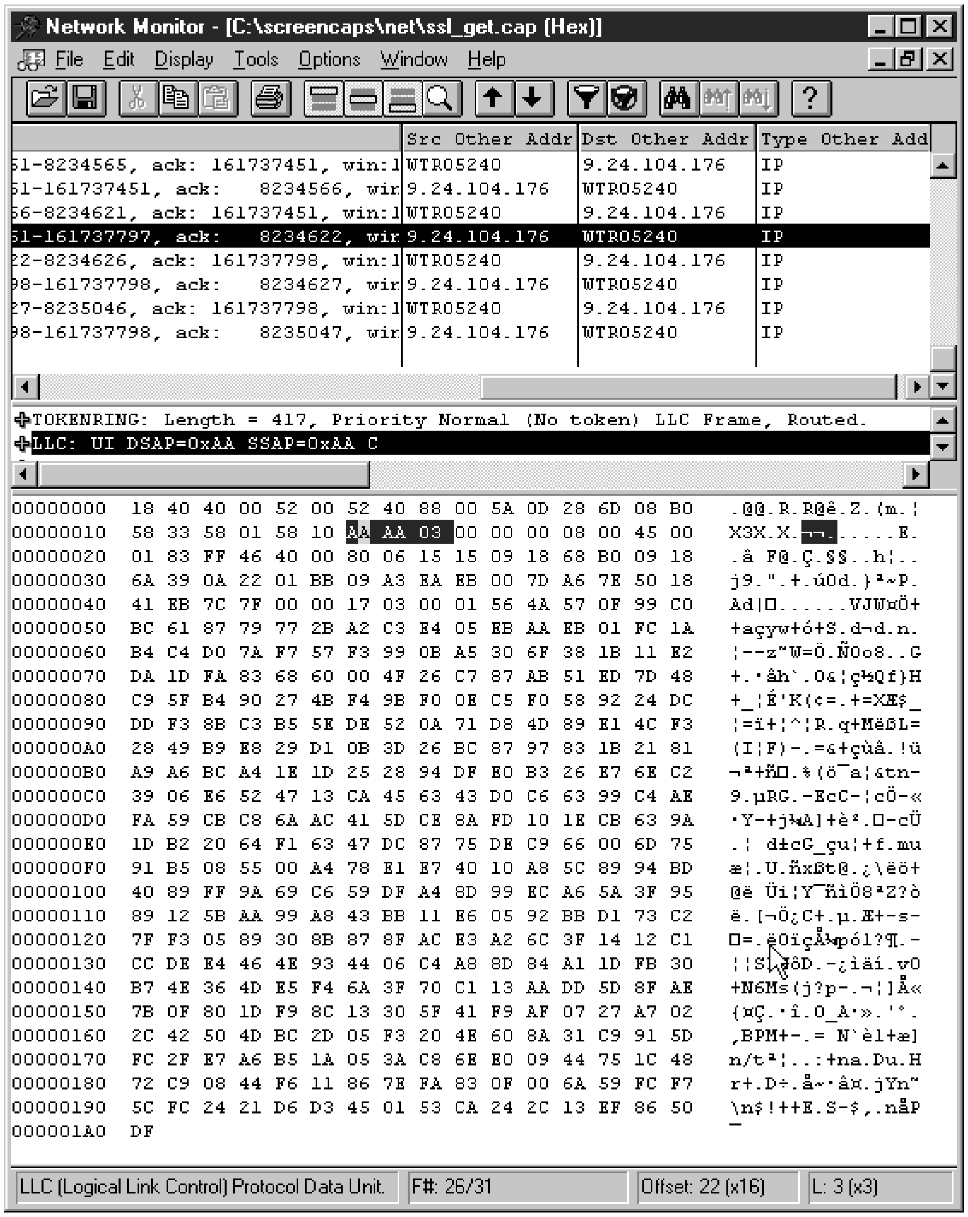
```
00000000  18 40 40 00 52 00 52 40 88 00 5A 0D 28 6D 08 B0   .@@.R.R@ê.Z.(m.¦
00000010  58 33 58 01 58 10 AA AA 03 00 00 00 08 00 45 00   X3X.X.¬¬......E.
00000020  01 51 E7 46 40 00 80 06 2D 47 09 18 68 B0 09 18   .QtF@.Ç.-G..h¦..
00000030  6A 39 0A 20 00 50 09 A2 83 03 00 7C 3C FF 50 18   j9. .P.óâ..|< P.
00000040  44 68 02 C2 00 00 50 4F 53 54 20 2F 73 65 72 76   Dh.-..POST /serv
00000050  6C 65 74 2F 45 63 68 6F 53 65 72 76 6C 65 74 20   let/EchoServlet
00000060  48 54 54 50 2F 31 2E 30 0D 0A 52 65 66 65 72 65   HTTP/1.0..Refere
00000070  72 3A 20 68 74 74 70 3A 2F 2F 39 2E 32 34 2E 31   r: http://9.24.1
00000080  30 36 2E 35 37 2F 69 64 5F 70 6F 73 74 2E 68 74   06.57/id_post.ht
00000090  6D 6C 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20   ml..Connection:
000000A0  4B 65 65 70 2D 41 6C 69 76 65 0D 0A 55 73 65 72   Keep-Alive..User
000000B0  2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F   -Agent: Mozilla/
000000C0  34 2E 30 35 20 5B 65 6E 5D 20 28 57 69 6E 4E 54   4.05 [en] (WinNT
000000D0  3B 20 49 29 0D 0A 48 6F 73 74 3A 20 39 2E 32 34   ; I)..Host: 9.24
000000E0  2E 31 30 36 2E 35 37 0D 0A 41 63 63 65 70 74 3A   .106.57..Accept:
000000F0  20 69 6D 61 67 65 2F 67 69 66 2C 20 69 6D 61 67    image/gif, imag
00000100  65 2F 78 2D 78 62 69 74 6D 61 70 2C 20 69 6D 61   e/x-xbitmap, ima
00000110  67 65 2F 6A 70 65 67 2C 20 69 6D 61 67 65 2F 70   ge/jpeg, image/p
00000120  6A 70 65 67 2C 20 69 6D 61 67 65 2F 70 6E 67 2C   jpeg, image/png,
00000130  20 2A 2F 2A 0D 0A 41 63 63 65 70 74 2D 4C 61 6E    */*..Accept-Lan
00000140  67 75 61 67 65 3A 20 65 6E 0D 0A 41 63 63 65 70   guage: en..Accep
00000150  74 2D 43 68 61 72 73 65 74 3A 20 69 73 6F 2D 38   t-Charset: iso-8
00000160  38 35 39 2D 31 2C 2A 2C 75 74 66 2D 38 0D 0A      859-1,*,utf-8..
```

```
LLC (Logical Link Control) Protocol Data Unit.   F#: 5/13   Offset: 22 (x16)   L: 3 (x3)
```

Figure 99. The URL on Using the POST Method, at the Network Monitor

This confirms that even if the POST method offers more privacy in that at least it does not append information like user ID and password to the URL, a MIM would be able to intercept private data reading inside the packets.

We now know how the form data really flows between the client and the server using the GET and the POST method. Let's now see how it goes on a secure channel. To see this, we accessed the same form, using SSL. We only used server authentication, since enabling client authentication would not have changed the result of our experiment and we wanted to keep things simple. Here's what we found when the GET method was used:



```
                                  Src Other Addr  Dst Other Addr  Type Other Add
51-8234565, ack: 161737451, win:1 WTR05240        9.24.104.176    IP
51-161737451, ack:   8234566, wir 9.24.104.176    WTR05240        IP
56-8234621, ack: 161737451, win:1 WTR05240        9.24.104.176    IP
51-161737797, ack:   8234622, wir 9.24.104.176    WTR05240        IP
22-8234626, ack: 161737798, win:1 WTR05240        9.24.104.176    IP
98-161737798, ack:   8234627, wir 9.24.104.176    WTR05240        IP
27-8235046, ack: 161737798, win:1 WTR05240        9.24.104.176    IP
98-161737798, ack:   8235047, wir 9.24.104.176    WTR05240        IP

TOKENRING: Length = 417, Priority Normal (No token) LLC Frame, Routed.
LLC: UI DSAP=0xAA SSAP=0xAA C

00000000  18 40 40 00 52 00 52 40 88 00 5A 0D 28 6D 08 B0   .@@.R.R@ê.Z.(m.¦
00000010  58 33 58 01 58 10 AA AA 03 00 00 00 08 00 45 00   X3X.X.¬¬......E.
00000020  01 83 FF 46 40 00 80 06 15 15 09 18 68 B0 09 18   .â F@.Ç.§§..h¦..
00000030  6A 39 0A 22 01 BB 09 A3 EA EB 00 7D A6 7E 50 18   j9.".+.úOd.}ª~P.
00000040  41 EB 7C 7F 00 00 17 03 00 01 56 4A 57 0F 99 C0   Ad|□......VJW¤Ö+
00000050  BC 61 87 79 77 2B A2 C3 E4 05 EB AA EB 01 FC 1A   +açyw+ó+S.d¬d.n.
00000060  B4 C4 D0 7A F7 57 F3 99 0B A5 30 6F 38 1B 11 E2   ¦--z˜W=Ö.Ñ0o8..G
00000070  DA 1D FA 83 68 60 00 4F 26 C7 87 AB 51 ED 7D 48   +.·âh`.O&¦ç½Qf}H
00000080  C9 5F B4 90 27 4B F4 9B F0 0E C5 F0 58 92 24 DC   +_¦É'K(¢=.+=XÆ$_
00000090  DD F3 8B C3 B5 5E DE 52 0A 71 D8 4D 89 E1 4C F3   ¦=ï+¦^¦R.q+MëßL=
000000A0  28 49 B9 E8 29 D1 0B 3D 26 BC 87 97 83 1B 21 81   (I¦F)-.=&+çùâ.!ü
000000B0  A9 A6 BC A4 1E 1D 25 28 94 DF E0 B3 26 E7 6E C2   ¬ª+ñ□.%(ö¯a¦&tn-
000000C0  39 06 E6 52 47 13 CA 45 63 43 D0 C6 63 99 C4 AE   9.µRG.-EcC-¦cÖ-«
000000D0  FA 59 CB C8 6A AC 41 5D CE 8A FD 10 1E CB 63 9A   ·Y-+j¼A]+è².□-cÜ
000000E0  1D B2 20 64 F1 63 47 DC 87 75 DE C9 66 00 6D 75   .¦ d±cG_çu¦+f.mu
000000F0  91 B5 08 55 00 A4 78 E1 E7 40 10 A8 5C 89 94 BD   æ¦.U.ñxßt@.¿\ëö+
00000100  40 89 FF 9A 69 C6 59 DF A4 8D 99 EC A6 5A 3F 95   @ë Üi¦Y¯ñiÖ8ªZ?ò
00000110  89 12 5B AA 99 A8 43 BB 11 E6 05 92 BB D1 73 C2   ë.[¬Ö¿C+.µ.Æ+-s-
00000120  7F F3 05 89 30 8B 87 8F AC E3 A2 6C 3F 14 12 C1   □=.ë0ïçÅ¼pól?¶.-
00000130  CC DE E4 46 4E 93 44 06 C4 A8 8D 84 A1 1D FB 30   ¦¦S¦FôD.-¿iäí.v0
00000140  B7 4E 36 4D E5 F4 6A 3F 70 C1 13 AA DD 5D 8F AE   +N6Ms(j?p-.¬¦]Å«
00000150  7B 0F 80 1D F9 8C 13 30 5F 41 F9 AF 07 27 A7 02   {¤Ç.·î.0_A·».'°.
00000160  2C 42 50 4D BC 2D 05 F3 20 4E 60 8A 31 C9 91 5D   ,BPM+-.= N`èl+æ]
00000170  FC 2F E7 A6 B5 1A 05 3A C8 6E E0 09 44 75 1C 48   n/tª¦..:+na.Du.H
00000180  72 C9 08 44 F6 11 86 7E FA 83 0F 00 6A 59 FC F7   r+.D÷.å~·â¤.jYn˜
00000190  5C FC 24 21 D6 D3 45 01 53 CA 24 2C 13 EF 86 50   \n$!++E.S-$,.nåP
000001A0  DF                                                _

LLC (Logical Link Control) Protocol Data Unit.   F#: 26/31   Offset: 22 (x16)   L: 3 (x3)
```

Network Monitor - [C:\screencaps\net\ssl_get.cap (Hex)]
File Edit Display Tools Options Window Help

Figure 100. Data on Using the GET Method with SSL, at the Network Monitor

To be sure, we checked out all the frames and they contained similar unintelligible information. We tried the same thing with the POST method, and we found the Network Monitor output to be similar. This confirms that no sniffing can get to the form data if this is sent over a sufficiently strong secured channel.

Conclusion: we just ascertained in this section that if SSL is used, all information - including URL and form data - is encrypted while being transmitted. Whether the form handling protocol is POST or GET doesn't really matter. The data is encrypted anyway.

Hence, for complete security, the ideal solution would be the POST method over SSL. The GET method is lacking in that the confidential information is visible in the browser Location field and can be seen by anyone behind the user (although the user could always hide the Location field of the browser). Netscape Communicator allows this by selecting **Hide Location Toolbar** from the View menu). Moreover, the URL has also a limitation on length (depending on the browser). This problem is not security-related, but it would prevent the user from sending a very long stream of data (confidential or not) to the server using the GET method.

3.3 Access Control with Lotus Domino Go Webserver

As has been mentioned, Lotus Domino Go Webserver has a host of security features, not the least of which is the SSL support. In Lotus Domino Go Webserver, one can configure several protection setups to restrict access to certain files and directories. If used in tandem with the security ServletExpress has to offer, one can fine-tune the restrictions placed on the usage or access of system resources for clients. In this section, we hope to familiarize you with other security features Domino Go Webserver has to offer.

3.3.1 Creating Users and Groups

On installation of Lotus Domino Go Webserver, a single user is automatically created, with administration privileges and permissions to modify the settings and entries in the httpd.cnf configuration file (see Figure 284 on page 307). If you are running on Windows NT, you could go to the WINNT directory and view the file admin.pwd. We saw the following entry in the file admin.pwd, immediately after installing Lotus Domino Go Webserver and specifying `pistoia` as Administrator ID:

```
narayan:GpwCp8vRSxpqI:Administrator
```

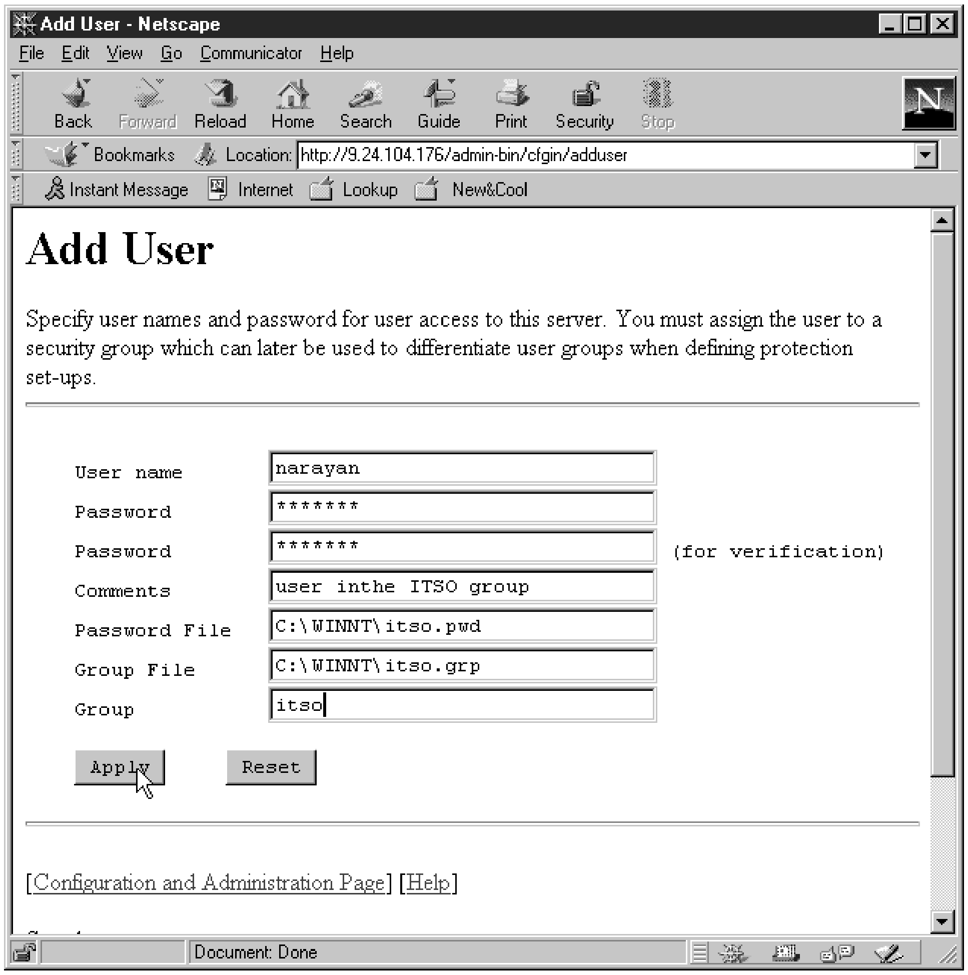
This entry is of the format:

```
name:encrypted_password:comments
```

Interestingly, Domino Go Webserver seems to be encrypting each newly added password by a different key. You can discover this by defining two users having the same password and noticing that the same password is encrypted in two different ways.

This administrator user can create users and groups and grant or revoke permissions to them.

First we created a new user. To do this, we accessed the Administration and Configuration Forms, logging on as the administrator ID, and clicked on **Administration of Users - Add User**. We got the following screen. We entered the values as shown and clicked on **Apply**.


Add User - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://9.24.104.176/admin-bin/cfgin/adduser
Instant Message Internet Lookup New&Cool

Add User

Specify user names and password for user access to this server. You must assign the user to a security group which can later be used to differentiate user groups when defining protection set-ups.

User name	narayan	
Password	*******	
Password	*******	(for verification)
Comments	user inthe ITSO group	
Password File	C:\WINNT\itso.pwd	
Group File	C:\WINNT\itso.grp	
Group	itso	

[Configuration and Administration Page] [Help]

Figure 101. Create a New User

When we did this, we had *not* created any file called itso.pwd or itso.grp, but, as expected, after submitting the form and receiving the confirmation message, we could see that the mentioned files had been created. After adding two more users in the same way, the file C:\WINNT\itso.pwd had the following entries:

```
narayan:Fy5W4wtxCm8x2:user in the ITSO group
bill:g6Z688NFPuAKo:user in the ITSO group
marc:X/l/NRPT4NWVY:yet another user
```

Looking inside the file C:\WINNT\itso.grp, we found one single entry:

```
itso: narayan, bill, marc
```

So that's about creating a new user in a new group. In the same page, you can find options to delete a user, change a user's password and check if a user exists or not. Owing to their simplicity, we have skipped them in our explanation.

3.3.2 Creating Protection Setups with Basic Authentication

We would recommend at this stage to go to the httpd.cnf file and read the section on protection. In fact, what we are saying here is an explanation of what is described there.

You could create a Protection Setup that can be later used by either `Protect` or `DefProt` directives. If you do not wish to create a Protection Setup, you could as well define the setup *in-line*. Let's take these terms one by one.

A Protection Setup is written to define how a set of resources is to be protected. A list of URL templates that activate that protection can be created. When the Web server receives a request, it compares the request to the templates. If the request matches one of the templates, the Web server activates the Protection as defined by the Protection Setup associated with the matching template.

If you look at your httpd.cnf file, even before you create any users or groups you will see the following entry for the administrator.

```
Protection  PROT-ADMIN  {
      PasswdFile   C:\WINNT\admin.pwd
      Mask         All@(*)
      PostMask     All@(*)
      PutMask      All@(*)
      GetMask      All@(*)
      AuthType     Basic
      ServerID     Private_Authorization
}
```

Figure 102. The Protection Setup for the Administrator narayan

The `Protect` and `DefProt` directives define the association of a Protection Setup with a set of resources to be protected. We show now how a `Protect` directive can be created. You can see in Figure 102 that the password file is specified, and that everything else is to be masked. Also, the type of authentication is basic - in other words, it is based upon user ID and password (another option could have been SSL). This Protection Setup is used with the `Protect` directive as shown, to effect a protection for all the resources mentioned.

```
Protect /admin-bin/*  PROT-ADMIN
Protect /Usage*       PROT-ADMIN
Protect /servlet/extConfigServlet PROT-ADMIN
Protect /servlet/intConfigServlet PROT-ADMIN
```

Figure 103. Using the Protect Directive along with the Protection Setup

The above directives, which we found to be installed by default, tell the server to protect the URL template `/admin-bin/*` using the Protection named PROT-ADMIN. If you access one of the pages of the Administration and Configuration Forms (see for example Figure 101 on page 105), you can see that after the Web server name or IP address, you have `/admin-bin/` in the URL field. One of the purposes of the `Protect` directives shown in Figure 103 is just to apply the Protection Setup PROT-ADMIN to the Administration and Configuration Forms. When you enter user ID and password for the administrator, you can log in only if what you typed matches what the system finds in the password file C:\WINNT\admin.pwd.

As we said, the directives shown in Figure 103 on page 106 come with the default installation of Lotus Domino Go Webserver. Note that two `Protect` directives are referred to external and internal servlets. The reason for this is that when we installed this copy of Lotus Domino Go Webserver we had also enabled the Java servlet support. You wouldn't find those two directives if you had not installed the Java servlet support or if you had disabled it, for example to install the servlet engine provided by ServletExpress.

We are going to show you how to create similar Protection Setups. Instead of accessing the configuration file indirectly, through the graphical Configuration and Administration Forms, the technique we use here is editing the configuration file, and writing our own Protection Setups manually. In this way we can also show you how Protection Setups can be stored in separate files, and how these files must then be specified in the configuration file httpd.cnf. However, in other examples, we will use the GUI to configure the Web server, and so you will be able to have a complete understanding. Here is how we did it.

We created a Protection called itso, and stored it in the *access file* D:\WWW\setup_itso.acc. The file contained the following.

```
Protection  itso  {
      PasswdFile   C:\WINNT\itso.pwd
      Mask         All@(*)
      PostMask     All@(*)
      PutMask      All@(*)
      GetMask      All@(*)
      AuthType     Basic
      ServerID     Private_Authorization
}
```

Figure 104. The Contents of the File D:\WWW\setup_itso.acc

You can see that it is not very original, except for the fact that we have stored it in a file, and will be referring to it by giving the file name in the configuration file. So, in the configuration file, we added only one line:

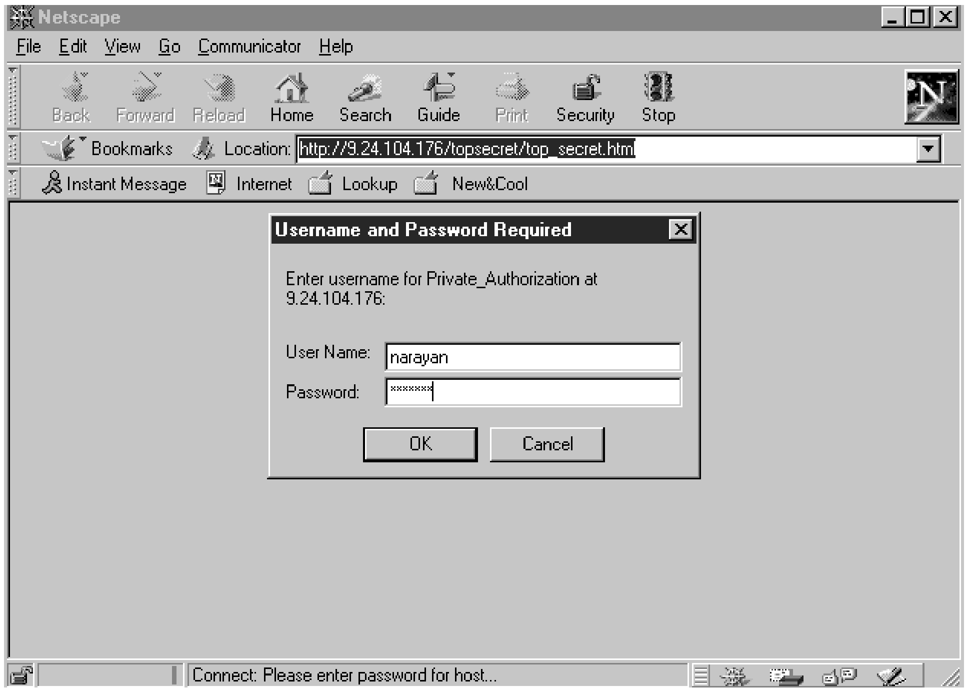
```
Protect /topsecret/* D:\WWW\setup_itso.acc
```

Note that it was not really necessary to create a separate file; we could just as well have entered the Protection Setup directly inside the httpd.cnf file, and then refer to it by its name, for instance

```
Protect /topsecret/* itso
```

We created a separate file to show you another possibility.

We then saved the configuration file, restarted the server, and accessed the page http://9.24.104.176/topsecret/top_secret.html with Netscape Navigator. We got a nice pop-up window asking for user ID and password.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location:
Instant Message Internet Lookup New&Cool
OK
Cancel

Location: http://9.24.104.176/topsecret/top_secret.html

Username and Password Required

Enter username for Private_Authorization at 9.24.104.176:

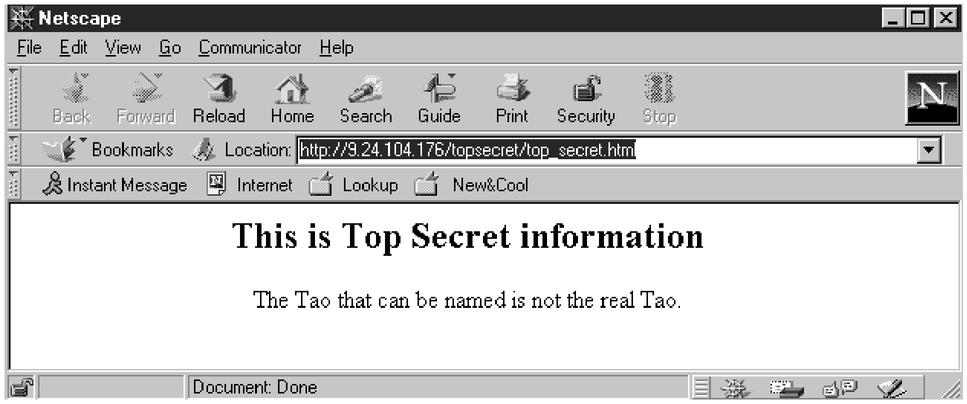
User Name: narayan

Password: *******

Connect: Please enter password for host...

Figure 105. Asking for User ID and Password

By entering user ID and password and clicking **OK** we could get the protected page top_secret.html.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location:
Instant Message Internet Lookup New&Cool

Location: http://9.24.104.176/topsecret/top_secret.html

This is Top Secret information

The Tao that can be named is not the real Tao.

Document: Done

Figure 106. The Protected Page.

The rest of the things behave pretty much as is expected with the normal password authentication. On entering an incorrect user ID or password, you will be prompted with yet another screen asking you if you want to retry. On cancelling it, you will be denied access.

Now let's see how the `DefProt` directives can be used. You could use them to have a default Protection for a certain template over which certain other special cases ought to be dealt with using another Protection Setup. Notice that the `DefProt` directive only defines a Protection Setup that should be used by default, but does not apply this. We will explain this with an example.

We used the `DefProt` directive entering the following three lines in the httpd.cnf file:

```
DefProt /topsecret/* D:\WWW\setup_itso.acc
Protect /topsecret/ccinfo/*
Protect /topsecret/purchaseinfo/* D:\WWW\setup_pur.acc
```

Note the statements:

1. The `DefProt` directive defines what the default Protection will be for all the files and directories starting with the template `/topsecret/`.
2. The next line:

   `Protect / topsecret/ccinfo/*`

   really demonstrates how the `DefProt` directive can be used. In fact you can see that we simply use the `Protect` directive, without specifying anything else but the target name. Note that we have not specified any access file or any Protection Setup names in this second line. In this case the default is taken to be what has been used in the `DefProt` statement.
3. The third directive specifies both a target and an access file. Note that this also begins with the same template `/topsecret/`, but the Protection specified in the file setup_pur.acc will be applied to this *overriding* the default Protection specified in the file setup_itso.acc.

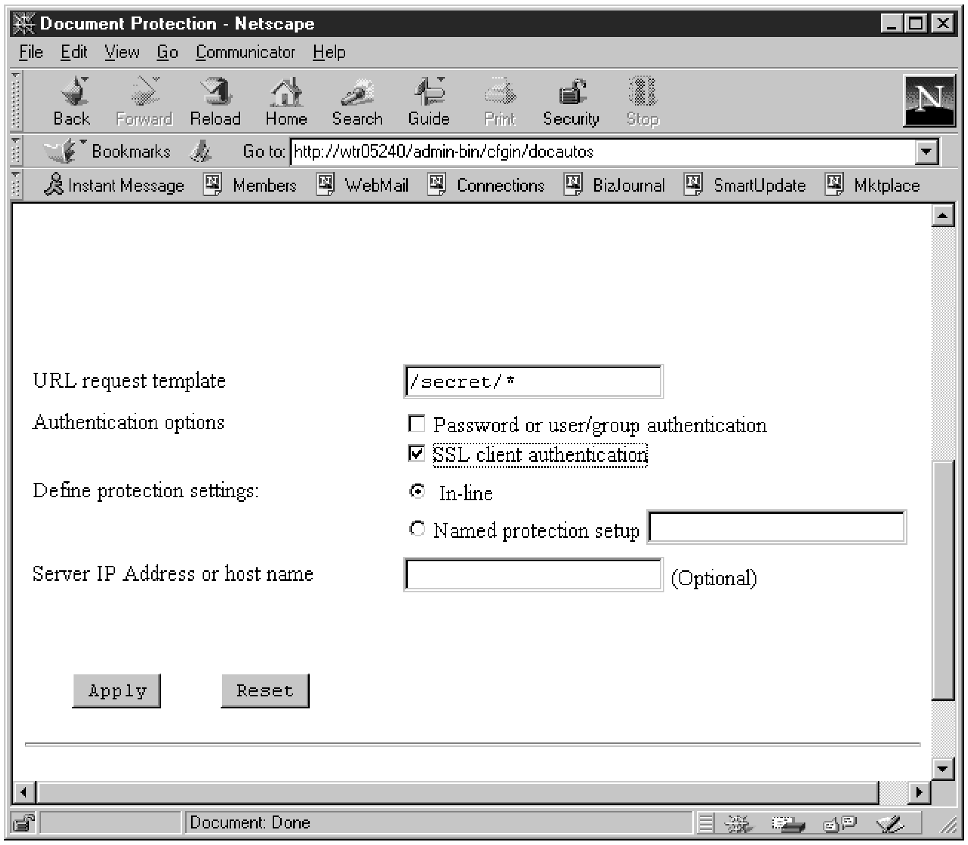
We added this in the configuration file, saved it, and restarted the Web server to get the following results:

1. To access any page in the directory WWW\HTML\topsecret\ we did not have to give any password, since, if you remember, the DefProt only defines the Protection Setup, but does not apply it.
2. To access any information in the directory WWW\HTML\topsecret\ccinfo\ we had to use one of the user IDs and passwords according to the access file setup_itso.acc. Note that the `Protect` directive for /topsecret/ccinfo/ does not contain any Protection Setup, and this is the reason why the DefProt for that template is taken as the default.
3. To access any of the files in WWW\HTML\topsecret\purchaseinfo\ we had to use any of the IDs as specified by the file setup_pur.acc. In this case the default Protection specified in the file setup_itso.acc has been overwritten.

3.3.3 Document Protection with SSL Authentication

All that has been discussed so far has been about basic authentication, since this is the simplest to set up. If you are studying the security setup of Lotus Domino Go Webserver for some intranet situation, you would probably find it easier to try things out with user IDs and passwords than with certificates. However, all that we have discussed so far about the basic authentication is also valid for certificates. Notice that, to try these things out, you first need to enable SSL client authentication, and include the CA who signed the client certificate in your working key ring (as explained in 3.1.4, “SSL Client Authentication” on page 84). Once that is done, you could either go through the configuration pages, or hand code the directives yourselves. This time we chose to go the configuration page way. Here's how.

We went to the Administration and Configuration Forms, and clicked on **Document Protection**. On the page we obtained, we filled in the fields as shown.


Document Protection - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Go to: http://wtr05240/admin-bin/cfgin/docautos
Instant Message Members WebMail Connections BizJournal SmartUpdate Mktplace
Apply Reset
Document: Done

URL request template: /secret/*

Authentication options: [ ] Password or user/group authentication [✓ SSL client authentication]

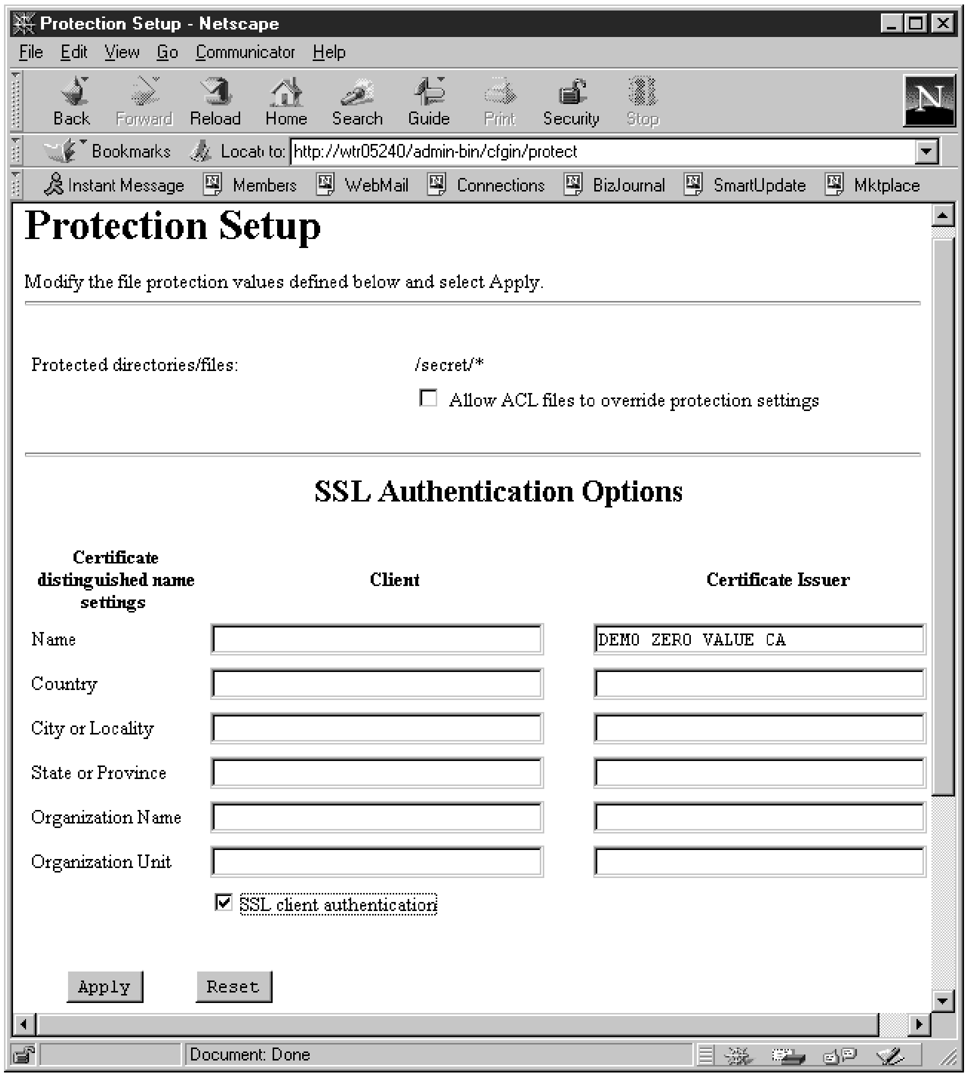
Define protection settings: [✓ In-line] Named protection setup

Server IP Address or host name: (Optional)

Figure 107. Choosing a Document to Protect, and Opting SSL.

Note that we have chosen **In-line**, rather than creating a separated Protection file or Protection Setup. We will see later how the httpd.cnf file will reflect this choice.

This led us to the page shown in Figure 108 on page 111 and we entered the name of the Certificate Issuer.


Protection Setup - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/admin-bin/cfgin/protect
Instant Message Members WebMail Connections BizJournal SmartUpdate Mktplace
Document: Done

Protection Setup

Modify the file protection values defined below and select Apply.

Protected directories/files: /secret/*

☐ Allow ACL files to override protection settings

SSL Authentication Options

Certificate distinguished name settings	Client	Certificate Issuer
Name		DEMO ZERO VALUE CA
Country		
City or Locality		
State or Province		
Organization Name		
Organization Unit		

☑ SSL client authentication

Apply Reset

Figure 108. Specifying the Client Certificate Particulars

Note here, that you could choose to enter any number of fields, and leave the rest empty. What we have chosen is to trust all clients certified by DEMO ZERO VALUE CA. Also, we have chosen not to let the ACLs override this. We will discuss more about this when we talk about ACLs.

The entry that automatically resulted in the configuration file httpd.cnf as a result of what we just did is shown below:

```
Protect /secret/* {
IssuerName "DEMO ZERO VALUE CA"
SSL_ClientAuth client
ACLOverride Off
Mask  Anybody@(*)
AuthType None
}
```

Note that, when we defined the Protection settings, we chose **In-line** and hence, no Protection Setup or Protection file has been created. Also note that `ACLOverride` is set to `Off`, as we chose.

We had a bit of trouble with the 30-day trial version of Domino Go with this functionality. However, we verified that it works fine in the final, purchased version.

When we did all this, restarted the Web server and accessed the page starting the URL with `http://` instead of `https://`, we were given the following error message:

```
Error 403 - Access Forbidden by rule
```

However, when we accessed the page starting the URL with `https://`, it first presented the server certificate, then asked for the client certificate, and upon verifying the certificate, presented the page to us. Note that for this method to be effective, the Web server administrator has to install the *correct* CA certificate in the key ring, and designate it as a trusted root. Otherwise, this will not work. The CA Certificate is needed in the key ring and needs to be designated as trusted for the server to be able to verify the issued certificate.

For more information about client certificates and client authentication, see 3.1.4, “SSL Client Authentication” on page 84, where you will also find how a CA certificate can be designated as trusted.

3.3.4 Using Document Protection with IP Address of Client

The document protection can be modified with the optional IP address tag, as shown below:

```
DefProt /topsecret/* itso 9.24.106.57
DefProt /topsecret/* PROT-ADMIN 9.24.104.176
```

This directives, added in the configuration file httpd.cnf, will accept the user ID and password in the Protection Setups itso (see Figure 104 on page 107) or PROT-ADMIN (see Figure 102 on page 106) depending on the client IP address.

Note that wildcard characters do not work here. If you type the following:

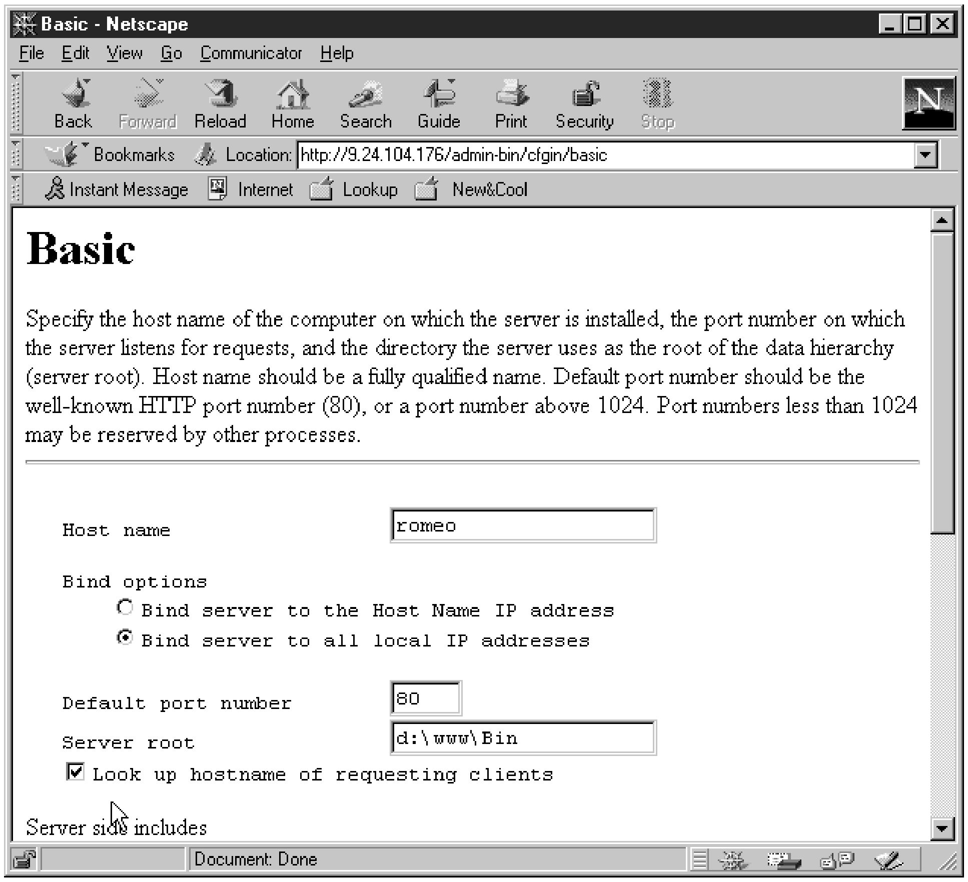
```
DefProt /topsecret/* itso 9.24.106.*
DefProt /topsecret/* PROT-ADMIN 9.24.104.*
```

the system will only accept the user ID and password corresponding to PROT-ADMIN, for *all* IP addresses, the same as if we had entered `*.*.*.*` as the client IP address. This indicates that you cannot add wildcard characters for this directive. Also, it reminds us that directives are overwritten - the last value of the directive is the one that is active. In fact, the second directive overwrote the first one.

Remember that every modification to the configuration file httpd.cnf requires that the Web server is restarted to take effect.

3.3.5 Using Document Protection with Domain Name of Client

Another possibility to protect a document is using the domain name of the client. To do this, you must first select **Look up host names of requesting clients** by marking the corresponding check box to be found in the Basic configuration page. The Basic configuration page can be reached by clicking on **Basic** from the Configuration and Administration Forms home page.


Basic - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://9.24.104.176/admin-bin/cfgin/basic
Instant Message Internet Lookup New&Cool
Document: Done

Basic

Specify the host name of the computer on which the server is installed, the port number on which the server listens for requests, and the directory the server uses as the root of the data hierarchy (server root). Host name should be a fully qualified name. Default port number should be the well-known HTTP port number (80), or a port number above 1024. Port numbers less than 1024 may be reserved by other processes.

Host name: romeo

Bind options
- Bind server to the Host Name IP address
- [✓ Bind server to all local IP addresses]

Default port number: 80

Server root: d:\www\Bin

[✓] Look up hostname of requesting clients

Server side includes

Figure 109. Enabling the Server to Look up Domain Name of Requesting Clients

Once this is done, you can use this facility by adding directives like the following to the configuration file httpd.cnf.

```
DefProt /topsecret/* itso cary.ibm.com
DefProt /topsecret/* PROT-ADMIN raleigh.ibm.com
```

The above directives when added to the configuration file will take the user IDs and passwords specified by the Protection Setup itso (see Figure 104 on page 107) if the client has the domain name cary.ibm.com, and that specified in the Protection Setup PROT-ADMIN (see Figure 102 on page 106) if the client is from raleigh.ibm.com. This is the only way you will be able to discriminate between users from different domain names, since IP addresses do not take wildcard characters.

3.3.6 Lotus Domino Go Webserver Access Control Lists

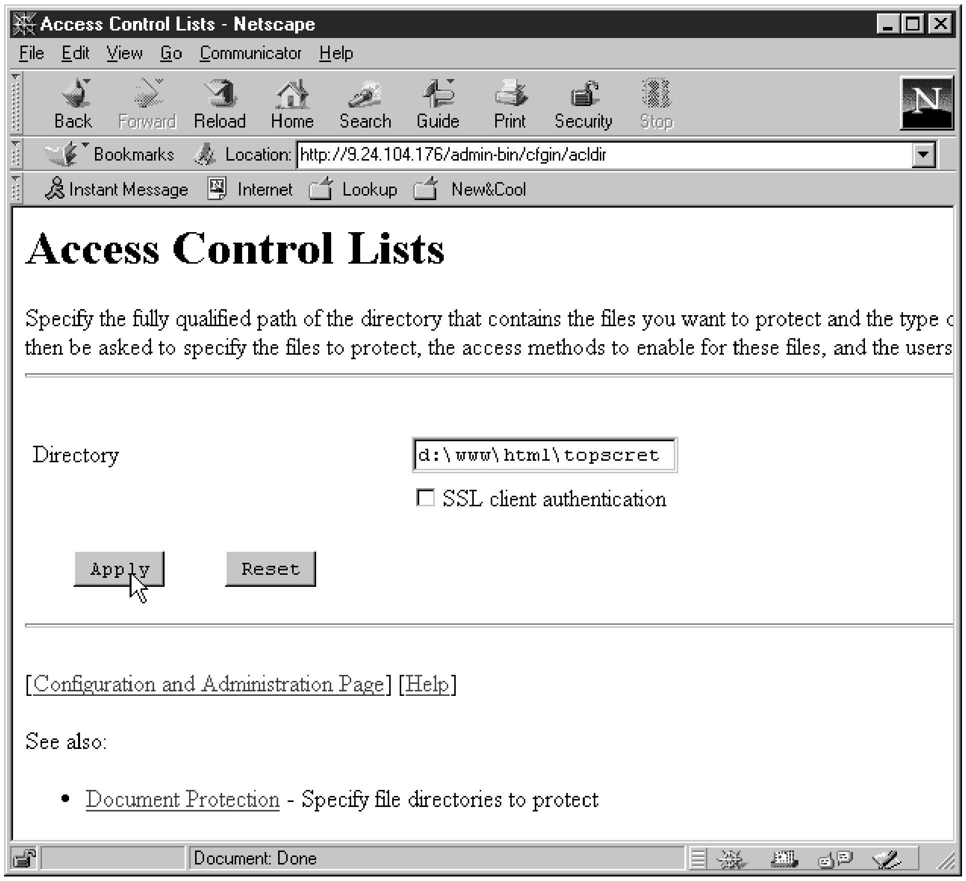
Access Control Lists (ACLs) could typically be used in tandem with Protection Setups, in the sense that a Protection Setup is usually utilized to restrict access to directories, and ACLs to specific files within a directory already protected by a Protection Setup. For this reason, we could say that a Protection Setup is used to define the first level of access control and then an ACL further limits access. However, if you want all the control to come from an ACL, you can override a Protection Setup by marking the **Allow ACL files to override protection settings** check box on the Protection Setup form (see Figure 108 on page 111).

On creating an ACL using the Graphical User Interface, you would see an ACL file called .www_acl in the particular directory. Each directory can only have one ACL file, and this file is read from top to bottom by the Web server before serving any files from that directory. We will first create an ACL using the GUI, then we will read the ACL file that is automatically generated to understand how we could create other ACL files by ourselves, hand coding them directly.

Notice that ACLs can be applied to work either with basic authentication (based on user IDs and passwords) or with SSL client authentication (see 3.1.4, “SSL Client Authentication” on page 84).

3.3.7 Using ACLs with Basic Authentication

First go to the link **Access Control Lists** in the Administration and Configuration Forms and enter the fully qualified directory name in the Directory field. Here is when we entered the directory name `D:\WWW\HTML\topsecret` in the Access Control Lists page.


Access Control Lists

Specify the fully qualified path of the directory that contains the files you want to protect and the type c
then be asked to specify the files to protect, the access methods to enable for these files, and the users

Directory d:\www\html\topscret

☐ SSL client authentication

[Configuration and Administration Page] [Help]

See also:

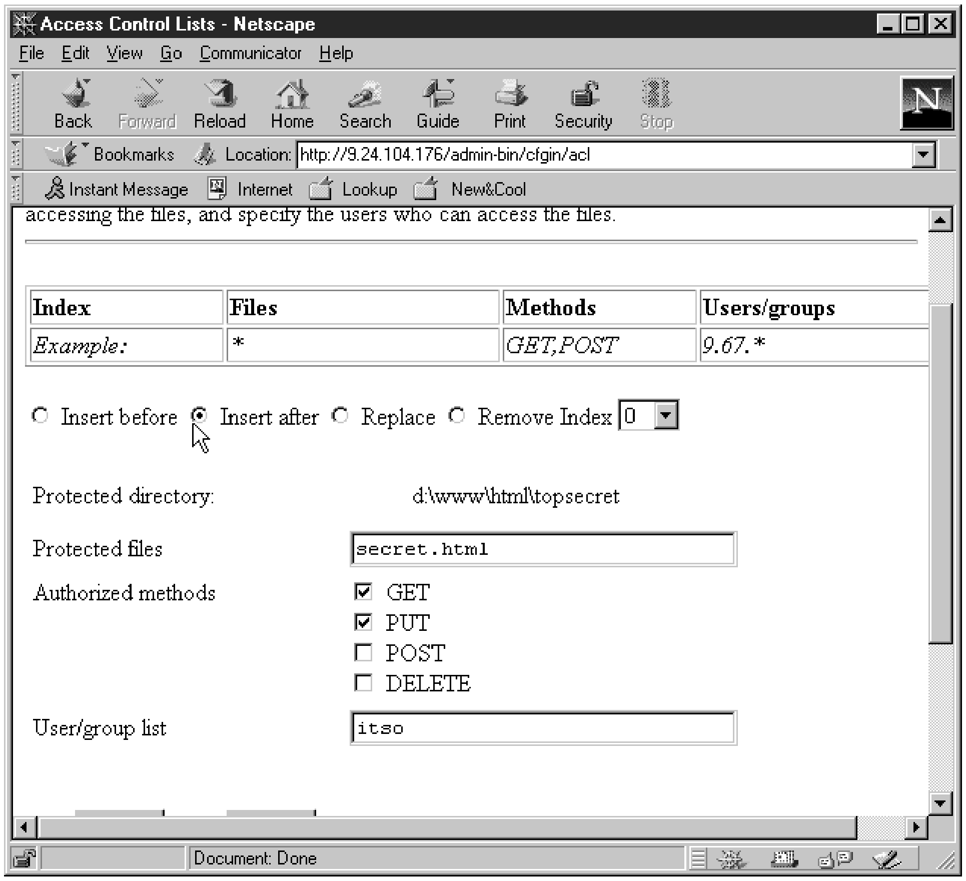
- Document Protection - Specify file directories to protect

Access Control Lists - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://9.24.104.176/admin-bin/cfgin/acldir
Instant Message Internet Lookup New&Cool
Apply
Reset
Document: Done

Figure 110. Setting up ACL - Specifying the Directory

Notice that in the above screen the **SSL client authentication** is not marked, since in this case we are experimenting with the basic authentication.

On the above screen, click **Apply** and then you will be prompted to enter the permissions and the authorized users.


Access Control Lists - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://9.24.104.176/admin-bin/cfgin/acl
Instant Message Internet Lookup New&Cool
Document: Done

accessing the files, and specify the users who can access the files.

Index	Files	Methods	Users/groups
Example:	*	*GET,POST*	*9.67.**

Insert before [✓ Insert after] Replace Remove Index 0

Protected directory: d:\www\html\topsecret

Protected files: secret.html

Authorized methods: [✓ GET] [✓ PUT] POST DELETE

User/group list: itso

Figure 111. Setting Up ACL - Defining Permissions

Remember to select **Insert after** for the first time, or it will give you an error.

When you have set up the ACL, you can verify if it works by going to the particular page, secret.html, that we entered in the form, and verifying if the access is indeed restricted. As we mentioned before, this technique can be used in tandem with the Protection Setup mechanisms described in the earlier chapter for an optimum situation. However, we verified that anyone in the group itso (that we had created in 3.3.1, “Creating Users and Groups” on page 104) got access to that page, and anyone who wasn't, didn't. To see if the ACL really works, we disabled all of the directory protection built with Protection Setup mechanisms and associated with the directory D:\WWW\HTML\topsecret before doing so.

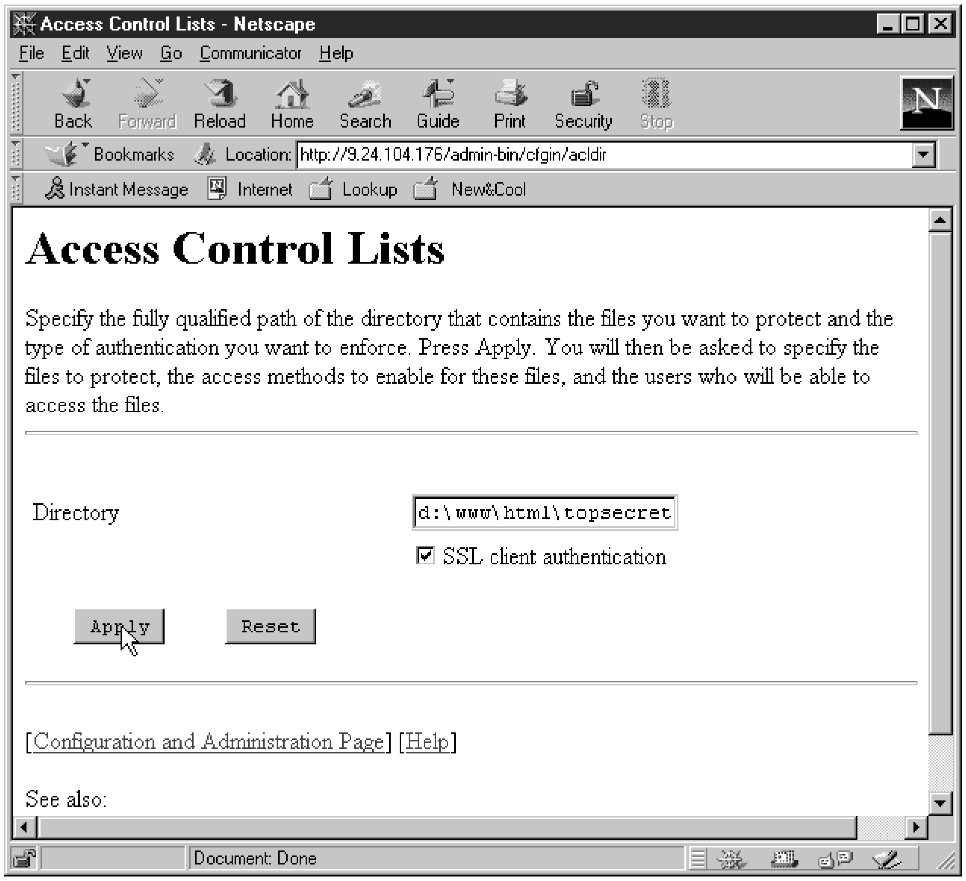
Let's now see what an ACL file looks like, so that we can edit other ACL files directly without bothering to go through the GUI pages every time. We opened the file .www_acl, automatically created by the system in the directory D:\WWW\HTML\topsecret, and we found it to contain only one line:

```
secret.html : GET,PUT : itso
```

Now, isn't that simple to extend on? The first entry is the file name, the second, the permitted HTTP methods, and the third, the group (or, alternately, the user) that is permitted to access it. Note that it is not necessary to specify the directory where the protected file is located, since it is the same directory where the file .www_acl is generated.

3.3.8 Using ACLs with SSL Client Authentication

ACLs can be configured to work with SSL client authentication rather similarly to ALCs with basic authentication. First go to the **Access Control Lists** link from the Administration and Configuration Forms. Enter the directory name, but remember this time to check the **SSL client authentication** check box.


Access Control Lists - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://9.24.104.176/admin-bin/cfgin/acldir
Instant Message Internet Lookup New&Cool
Apply
Reset
[Configuration and Administration Page] [Help]
See also:
Document: Done

Access Control Lists

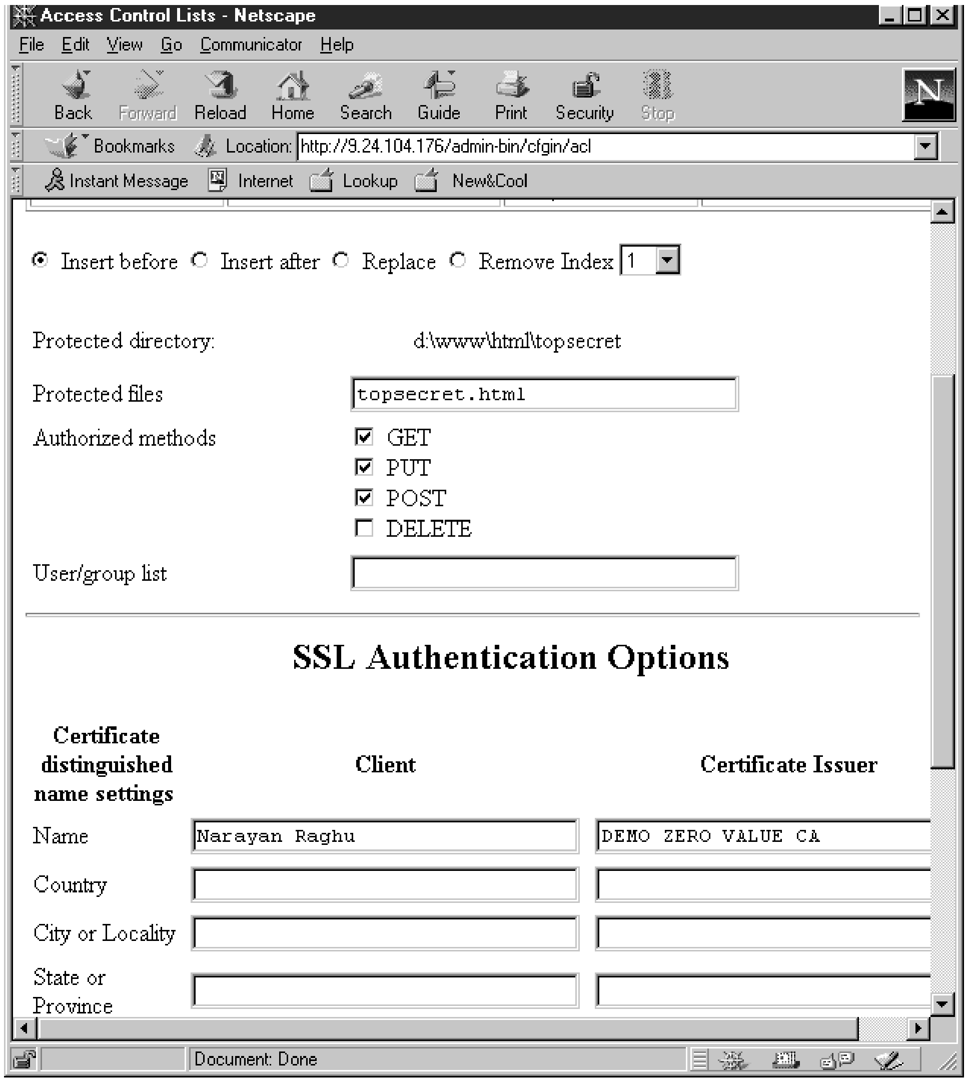
Specify the fully qualified path of the directory that contains the files you want to protect and the type of authentication you want to enforce. Press Apply. You will then be asked to specify the files to protect, the access methods to enable for these files, and the users who will be able to access the files.

Directory: d:\www\html\topsecret

[✓] SSL client authentication

Figure 112. ACL with SSL Client Authentication

The page you will receive is pretty similar to the one you would have seen before in Figure 108 on page 111, where you would have to enter the valid client certificate fields. As before, you can choose to leave any of the fields empty, and fill all that you like. Only, be sure to include the CA certificate in the working key ring, and set it to be trusted, else there is no way for the server to authenticate the client certificate. How to accomplish this operation is explained in 3.1.4, "SSL Client Authentication" on page 84.



Access Control Lists - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://9.24.104.176/admin-bin/cfgin/acl
Instant Message Internet Lookup New&Cool
Document: Done

[✓ Insert before] Insert after Replace Remove Index 1

Protected directory: d:\www\html\topsecret

Protected files: topsecret.html

Authorized methods: [✓ GET] [✓ PUT] [✓ POST] DELETE

User/group list:

SSL Authentication Options

Certificate distinguished name settings	Client	Certificate Issuer
Name	Narayan Raghu	DEMO ZERO VALUE CA
Country		
City or Locality		
State or Province		

Figure 113. ACL with SSL - Giving the Certificate Fields

Once you do this, you will be able to access the particular file that has been protected, only by presenting a client certificate that has been signed by the DEMO ZERO VALUE CA, which has the client common name as Narayan Raghu. This is of course, subject to condition that the DEMO ZERO VALUE CA's certificate has been designated as trusted root in the working key ring.

Let's now see what the ACL file looks like this time.

```
topsecret.html : GET,PUT,POST : !CommonName="Narayan Raghu",
!IssuerCommonName="DEMO ZERO VALUE CA"
secret.html : GET,PUT : itso
```

Notice that the first two lines will really be in the same line. Hence, ACLs can thus be used with directory permissions to get a system that has complete control over the files and who accesses them.

3.3.9 The file Servlet in ServletExpress

Having said all this, we'd like to mention that if you have ServletExpress installed, there is a way of getting around all this. Of course, there is a means of plugging that leak, using the access control functions of ServletExpress.

If you already have ServletExpress installed, and have restricted access to the file *server_root*\HTML\secret\topsecret.html, try accessing it by pointing your browser to http://*server_name*/servlet/file/secret/topsecret.html and don't be surprised if the servlet normally serves the file to you without asking for authentication.

This behavior depends on the particular servlet, named file, that is shipped with ServletExpress and has the ability to invoke and serve any HTML page that is appended to the URL, *without using the HTTP daemon*. The complete class file name for the file servlet is com.sun.server.http.FileServlet and it is automatically copied on your computer by the ServletExpress installation.

We mention methods to disable this servlet in 4.1.5, “Resources” on page 146.

Chapter 4. IBM WebSphere Application Server Security

In this chapter we give a comprehensive treatment of IBM WebSphere Application Server security.

IBM WebSphere Application Server consists of a Java-based servlet engine that is independent on both the Web server on which it is installed and the underlying operating system. This way, the goal write once, run everywhere becomes available also for servlet development.

The WebSphere Application Server offers a choice of server plug-ins that are compatible with the most popular server Application Programming Interfaces (APIs):

- Lotus Domino Go Webserver Version 4.6.1 or higher
- Apache Server Version 1.2.x
- Netscape Enterprise Server Version 2.01
- Netscape FastTrack Server Version 2.01
- Microsoft Internet Information Server Version 2.x, 3.x or 4.0

It supports the following operating systems:

- Microsoft Windows NT Version 4.0
- AIX Version 4.1.5 or higher
- Sun Solaris Version 2.5.1 with the Native Threads Package

In addition to a servlet engine and plug-ins, WebSphere application server provides the following components:

- Implementations of the JavaSoft Java Servlet API, plus extensions of and additions to the API (see 7.3, "IBM ServletExpress 1.0" on page 311).
- Sample applications that demonstrate how to use the basic classes and the extensions (see 7.3.5, "How to Add Servlets into ServletExpress" on page 327, Chapter 8, "Three-Tier Applications in Firewall-Protected Network Environments" on page 361 and Chapter 9, "IIOP in Firewall-Protected Network Environments" on page 435).
- The Application Server Manager, a Graphical User Interface (GUI) that makes it easy to set options for loading local and remote servlets, set initialization parameters, specify servlet aliases, create servlet chains and filters, monitor resources used by the Application Server, monitor loaded servlets and active servlet sessions, log servlet messages and perform other servlet management tasks. This feature will be used several times in this chapter.
- A connection management feature that caches and reuses connections to your Java Database Connectivity (JDBC)-compliant databases. When a servlet needs database connections, it can go to the pool of available connections. This eliminates the overhead required to open a new connection each time.
- Additional Java classes, coded to the JavaBeans specifications, that allow programmers to access JDBC-compliant databases. These data access beans provide enhanced function while hiding the complexity of dealing with relational databases. They can be used in a visual manner in an integrated development environment.

- Support for a new technology for dynamic page content called JavaServer Pages (JSP). JSP files can include any combination of HTML tags, `<SERVLET>` tags, `<INSERT>` tags, `<BEAN>` tags and NCSA tags (special tags that were the first method of implementing the server-side includes).
- CORBA Support: an Object Request Broker (ORB) and a set of services that are compliant with the Common Object Request Broker Architecture (CORBA). An extensive use of the CORBA Support is shown in Chapter 9, “IIOP in Firewall-Protected Network Environments” on page 435.

Note: At the time this book went to print, IBM WebSphere Application Server was still in its beta 3 version and was still named ServletExpress. Hence, if you have the final version, you might find some of the functionalities or the GUI to be different. However, we stayed in touch with the development team all through the writing of this book, and were not advised of any major changes (except a modification in the directory structure), and so you can safely refer to what we have written here about ServletExpress 1.0 beta 3 even if you have the final version 1.0 for IBM WebSphere Application Server.

Moreover, at the time of this project, the Java Development Kit 1.2 was available only in beta version. For this reason IBM WebSphere Application Server did not support the new JDK 1.2 yet. To install WebSphere Application Server, JDK 1.1.4 or 1.1.6 is required (JDK 1.1.5 has a memory leak problem and is not recommended for any platform).

To see all the steps we followed to install and configure ServletExpress correctly, refer to 7.3, “IBM ServletExpress 1.0” on page 311.

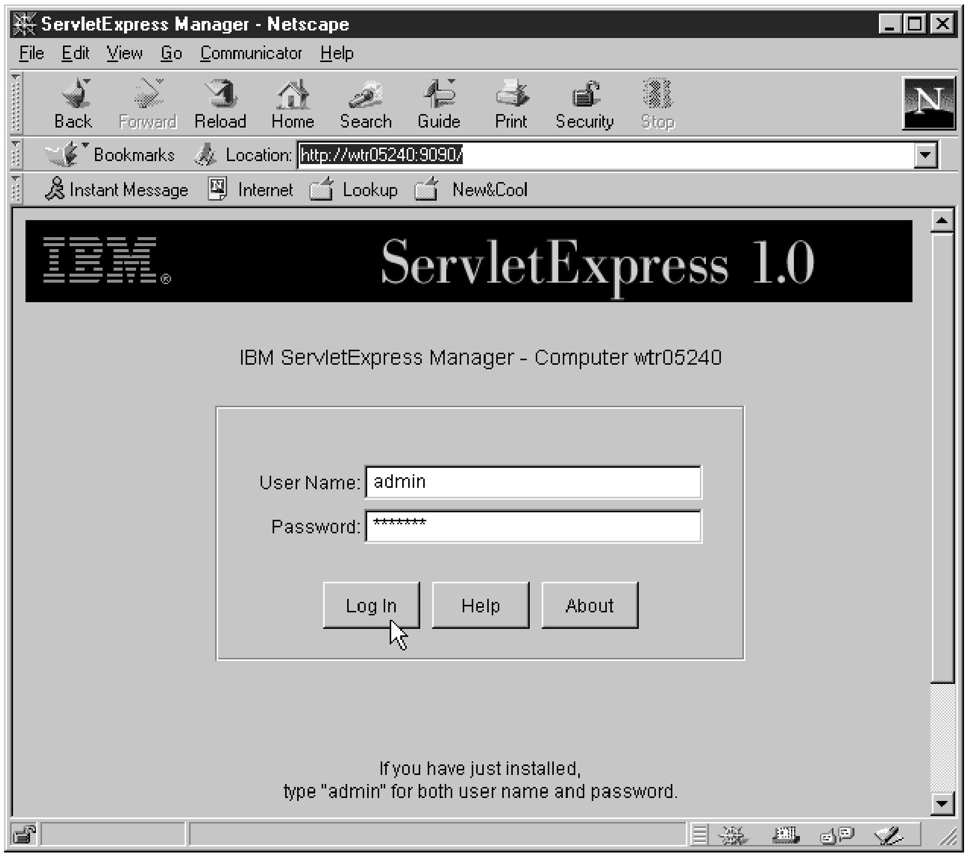
4.1 ServletExpress Security Management

In this section, we consider the security features that ServletExpress has to offer.

Basically, from a security point of view, ServletExpress permits the administrator to restrict access to some or all of the resources that have been installed on the Web server and that have been registered in ServletExpress. Access can be allowed to some of the users based on certificates or passwords. The administrator can create users and groups within realms, and add users to one or more groups. Positive permissions (permit) can be set to users and groups, in the sense that, with the permission model implemented by ServletExpress, the ServletExpress administrator can specify who can access a given resource. As we show later there are certain simple rules as to how conflicting permissions between users and groups are handled.

Here's how we set up Access Control Lists (ACLs). Our considerations are based upon our tests performed with Lotus Domino Go Webserver, installed without its servlet support, but using the servlet engine provided by ServletExpress.

We went to the ServletExpress Manager page at http://*servername*:9090, we keyed in the admin password, and logged into the system.



ServletExpress Manager - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location:
Instant Message Internet Lookup New&Cool
ServletExpress 1.0
Log In
Help
About

http://wtr05240:9090/

IBM ServletExpress Manager - Computer wtr05240

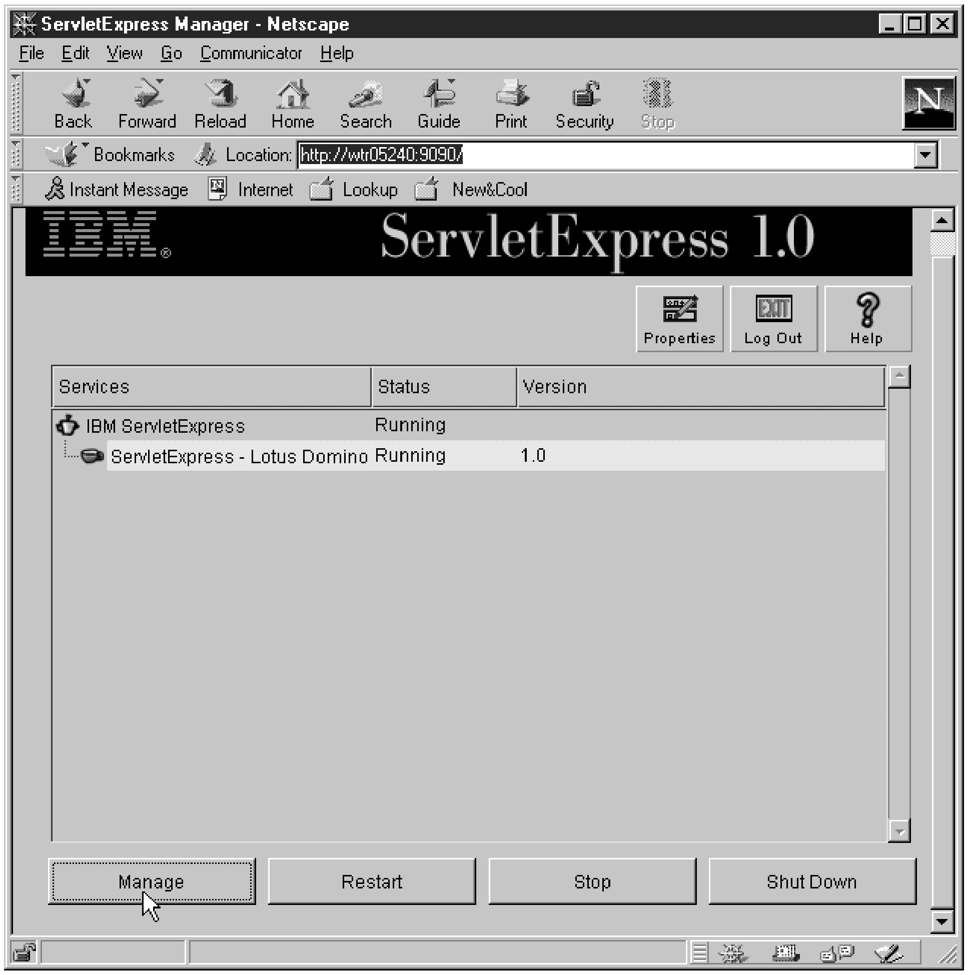
User Name: admin

Password: *******

If you have just installed,
type "admin" for both user name and password.

Figure 114. Login Page for ServletExpress

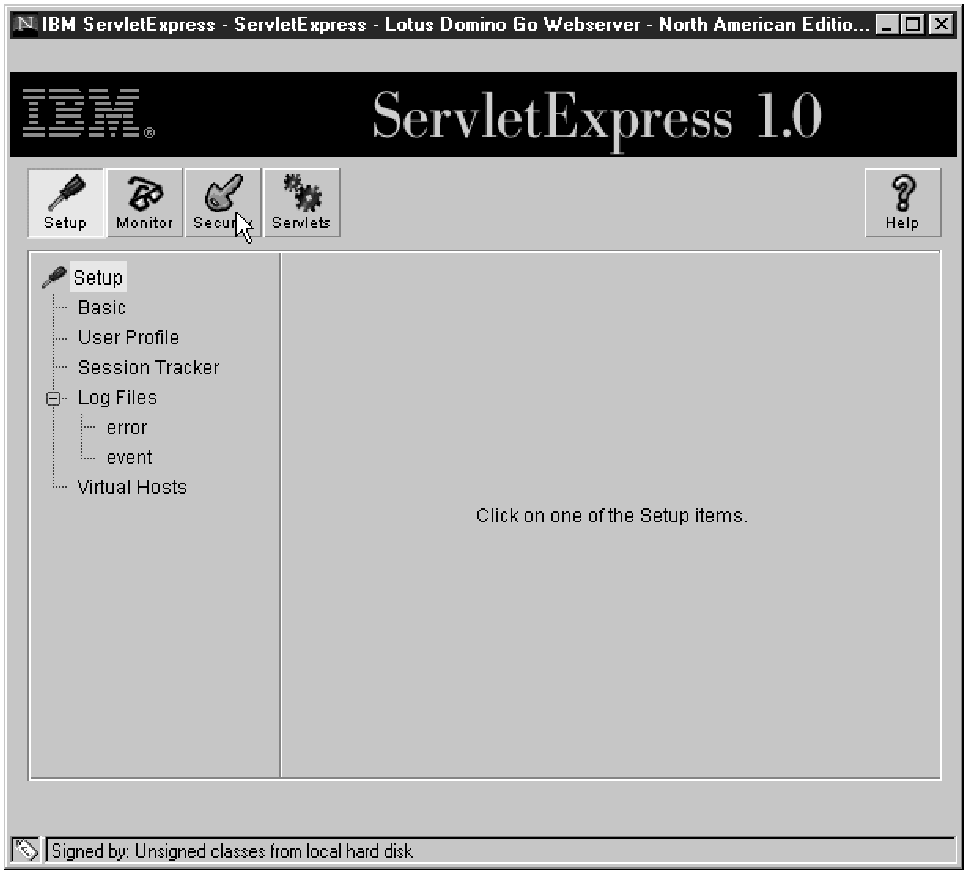
We got the following screen:


ServletExpress Manager - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240:9090/
Instant Message Internet Lookup New&Cool
IBM ServletExpress 1.0
Properties Log Out Help
Manage Restart Stop Shut Down

Services	Status	Version
IBM ServletExpress	Running	
ServletExpress - Lotus Domino	Running	1.0

Figure 115. Login Page for ServletExpress

When we clicked **Manage** a separate window was brought up and there we clicked the **Security** button:



Click on one of the Setup items.

Signed by: Unsigned classes from local hard disk

Figure 116. Administering ServletExpress

We have now logged in, and reached the Security page. Let us now understand how the system really functions.

4.1.1 Realms

Realms are used to organize users, groups and ACLs in a structured way to protect Web resources. In the context of ServletExpress, realms are in particular used for two different purposes: to authenticate a client and to decide which remote servlets to trust. In this section, we hope to familiarize you with how these two things are accomplished using the concept of realm.

The system has three realms built into it, in the Windows NT version:

1. defaultRealm
2. servletMgrRealm
3. NT

In the AIX version, the NT realm is replaced by the UNIX realm.

Here is what we can do with these realms:

1. *defaultRealm*

   The users registered in this realm are given permissions to execute certain servlets. Typically, the system administrator would include, in this realm, the *users* who are expected and permitted to access the servlets on the system, and the *resources* that are to be protected. The administrator can also add groups, and ACLs to facilitate the handling of these.

2. *servletMgrRealm*

   This realm contains a list of servlet-signers. The system administrator would add into this realm certificates of those signers whose signed servlets would be trusted to run on the server.

3. *NT* or *UNIX* realm.

   This realm is used to give the users of the system who already have IDs on the server access to the servlets through the Web. The system administrator cannot add or delete any users from this realm. The users in this realm will gain access to the servlets pretty much in the same way as those in defaultRealm, with keying in similar user IDs and passwords. This has been added to save the system administrator the trouble of creating duplicate IDs - one for the NT or UNIX, and the other to run servlets. The system administrator would add to this realm those resources that the users who have a login into the server can access. Apart from the fact that the system administrator cannot modify the user list, this realm functions pretty much in the same way as the defaultRealm.

We went browsing through the directories to get some specific information on where data is stored about realms. Our ServletExpress was installed in D:\ServletExpress, and we went to D:\ServletExpress\realms, where there was one file corresponding to each realm. We found these files to be text files, capable of being opened by notepad, containing information about the Java class name and the directory associated with this realm. We also saw that the defaultRealm, the servletMgrRealm and the NT or UNIX realms, which we will be working with extensively, have different classes associated with them.

The file name and content for the defaultRealm are given below.

```
# @(#)defaultRealm 1.4 97/09/10
#
# Configuration for the "default" realm, a low-security shared-password
# realm used for demo purposes.
#

classname=com.sun.server.realm.sharedpassword.SharedPasswordRealm
directory=realms/data/defaultRealm
```

Figure 117. D:\ServletExpress\realms\defaultRealm

The defaultRealm is implemented using the class com.sun.server.realm.sharedpassword.SharedPasswordRealm, which, according to Sun's documentation, implements a very simple authentication database that stores user passwords in a text file. We confirmed this by going to the data\defaultRealm subdirectory as indicated in the file and opening the file named keyfile. We saw all the users we had created in this realm, and their passwords.

```
Narayan::c3cxNTA0cg==
Tintin::c3cxNTA0cg==
jeeves::amVldmVz
Wooster::c3cxNTA0cg==
Popye::c3cxNTA0cg==
admin::YWRtaW4=
Asterix::c3cxNTA0cg==
pistoia::c3cxNTA0cg==
```

The entries on the left are the user names created in the defaultRealm and the entries on the right are their passwords, encoded, not encrypted, in base64 format. Since the passwords can be easily decoded by any base64-to-binary converter, access to this file should be restricted using the options available in the operating system. Also, we tried hacking into the system, by adding a user name and a base64-encoded password into this file, and then accessing the system as the newly created user. It worked. Hence, this file ought to be protected at all costs by any means available in the operating system. The consolation, however, is that this file is in no way accessible from the Web, and any hacker must find a way to access this file from the intranet, by logging on to the machine as a user.

The file name and content for the servletMgrRealm are given below.

```
# @(#)servletMgrRealm 1.5 97/09/10
#
# Configuration for the "servletMgr" realm, used to control the privileges
# assigned through the server sandbox.
#

classname=com.sun.server.realm.certificate.CertificateRealm
certclassname=sun.security.x509.X509Cert
directory=realms/data/servletMgrRealm
```

Figure 118. D:\ServletExpress\realms\servletMgrRealm

The servletMgrRealm is implemented using the class com.sun.server.realm.certificate.CertificateRealm, with the certificate class name sun.security.x509.X509Cert. This, according to Sun's documentation, will provide access to users based on their certificate, to be enrolled with it. Hence, users in this realm will be identified by their certificates, and can thus access their resources. *Users*, in this case, really means servlet-signers.

The file name and content for the NT realm are given below.

```
# @(#)NT 1.1 97/06/10
#
# Configuration for the "NT" realm, providing access to user accounts
# available through the NT NetUser family of calls.  Uses a local
# directory to store ACLs;
#

classname=com.sun.server.realm.nt.NTRealm
directory=realms/data/NT
```

Figure 119. D:\ServletExpress\realms\NT

The NT or the UNIX realms contain, as mentioned before, users that are already present in the system - in other words, those who have logins in the NT or UNIX machine in question. These same logins and passwords can be used to access the services offered by ServletExpress remotely.

Also, if you open the realms directory, you would see yet another realm, called the adminRealm, although this does not show up on the GUI. This is used to store information pertaining to the administrator's user ID and password. We opened the file D:\ServletExpress\realms\data\adminRealm\keyfile and we found only one line in this file:

```
admin::Y3M0ODAxcg==
```

This line represents the administrator's user ID, admin, followed by the password encoded in base64 format.

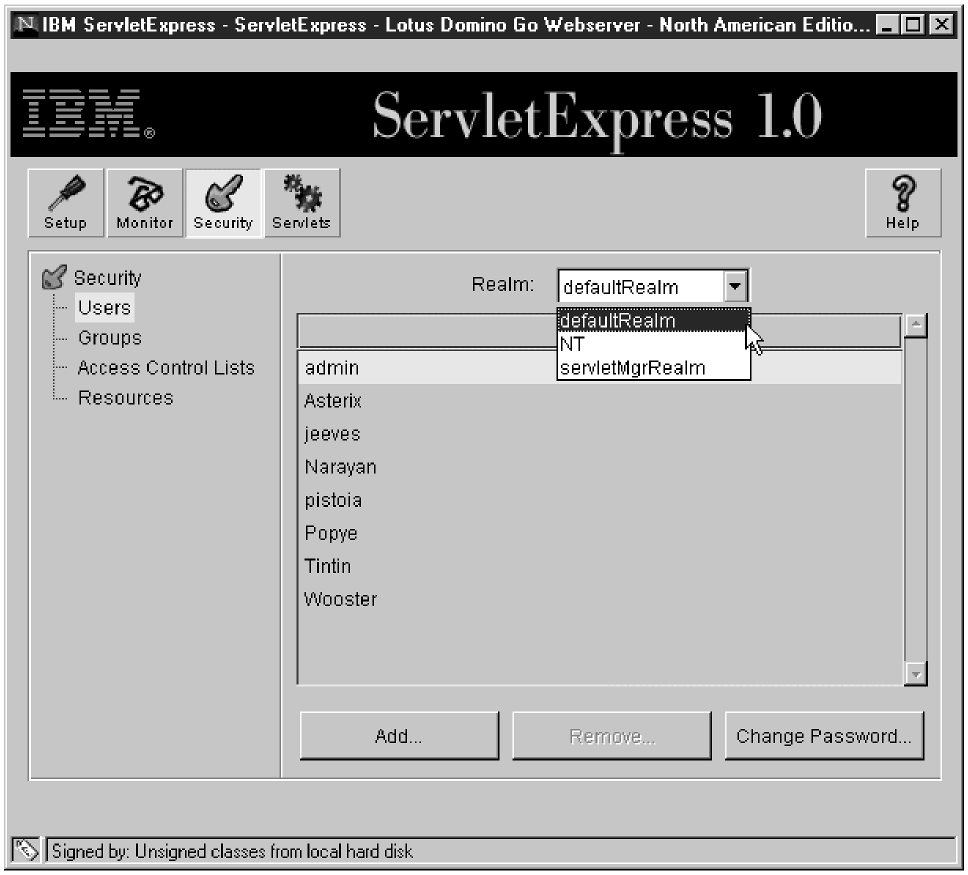
4.1.2 Users

The ServletExpress administrator, named admin, can create any number of users in the system. These users are to be created under defaultRealm or servletMgrRealm, but not under NT or UNIX, for the reason mentioned earlier. It is not possible, using the ServletExpress Manager GUI, to add a new user under the adminRealm, so no new administrators can be created.

If the ServletExpress administrator is spoofed, by adding another user ID and password in the file *servletexpress_root*\realms\data\adminRealm\keyfile, the newly added user can also create a user. We made this experiment: we created another administrator by manually adding a new user name in the keyfile below the adminRealm directory, followed by a password that we encoded in base64 format using again the binary-to-base64 converter. When we logged in the ServletExpress Manager as the new administrator, we were really able to add new users under the defaultRealm and the servletMgrRealm. However, the ServletExpress development team strictly warned us against accessing or modifying these files directly, since these might have unpredictable effects on the functioning of the system.

The following is how to create a new user in the defaultRealm. If a user is created under the defaultRealm, the administrator sets the password for the user, which cannot be changed by the user. This could have some security implications, since it is preferable to change passwords frequently. However, the only way the users can be permitted to change their passwords directly is by having a servlet do the password changing, and this will mean that the file storing the passwords is accessible from the net. If you so desire, you could write a servlet that takes the user ID and password, takes the new password, encodes it in base64 format, and replaces the old password with the new password in the file *servletexpress_root*\realms\data\defaultRealm\keyfile. This would obviate the necessity for an administrator's intervention to change the password for those using basic authentication.

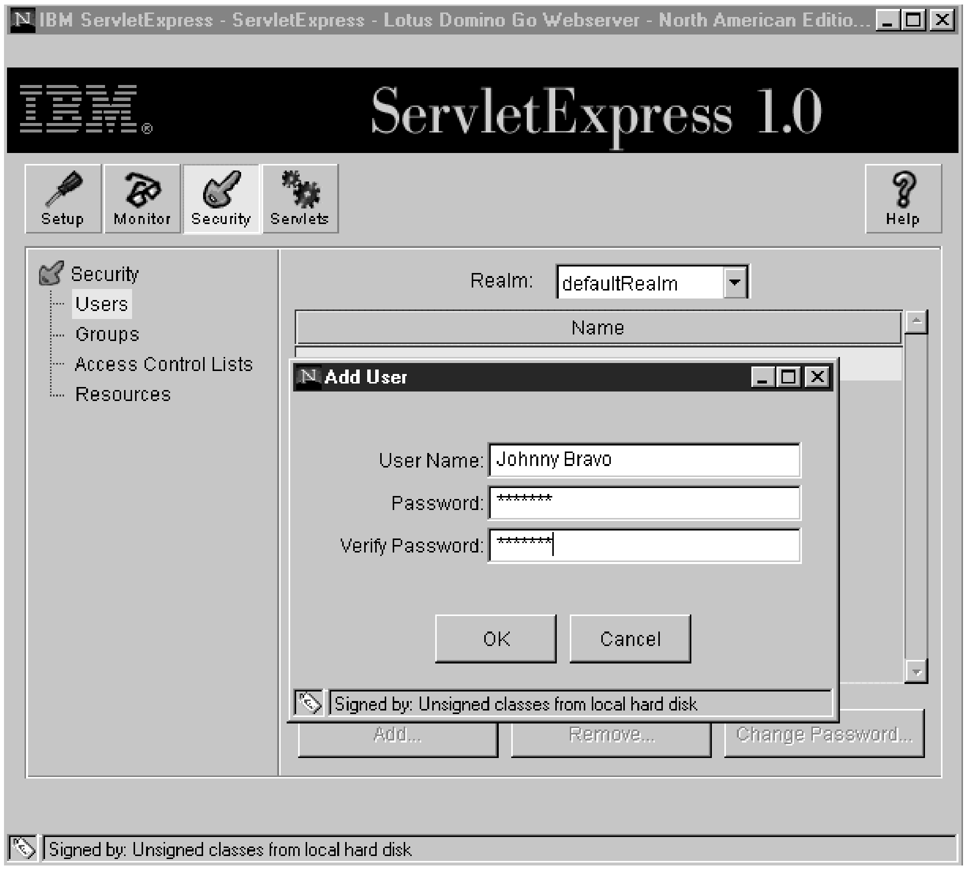
Let us now go through the process of creating a user, the way we did it. From the Security options, choose **Users** and from the drop down Realm list box, choose **defaultRealm**.



admin
Asterix
jeeves
Narayan
pistoia
Popye
Tintin
Wooster

Figure 120. Users

You can see that we have already created an interesting group of users. The way we did it was to simply click on the **Add...** button at the bottom, fill in the form that is automatically brought up, and press **OK**.



Realm: defaultRealm

User Name: Johnny Bravo
Password: *******
Verify Password: *******

Signed by: Unsigned classes from local hard disk

Signed by: Unsigned classes from local hard disk

Figure 121. Create User

Creating a user in the servletMgrRealm implies creating a user whose signature on servlets is valid. In the later sections, we will be explaining how servlets can be loaded remotely, and there you will see that it is possible to control access permissions to remotely loaded servlets based on who signed it. For this feature, it is important to enter all the signers in one place first. The users in servletMgrRealm are those whose signatures on the servlet ServletExpress will recognize.

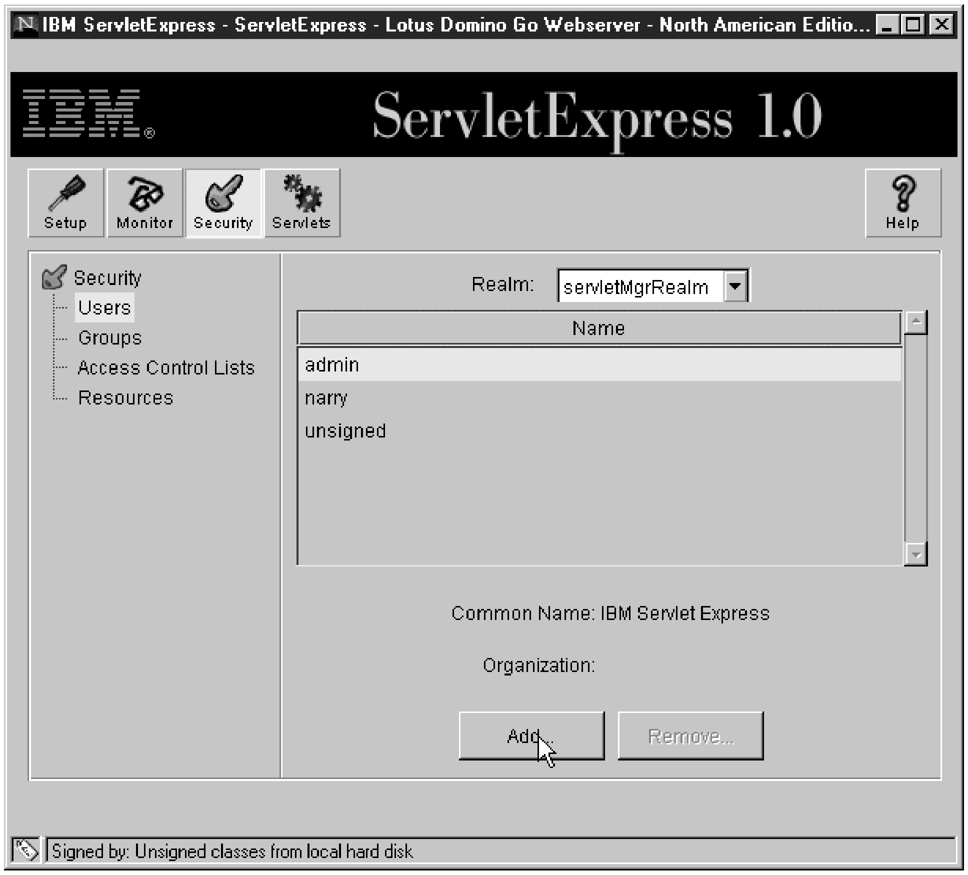
Based on who signed the JAR file containing the servlet, the servlet will be given permissions to perform various actions, such as reading from a file, writing to a file or opening a remote socket. The good thing about this is that the permissions can be controlled finely, in the sense that the administrator could decide the servlets signed by which servletMgrRealm user to be given permissions to read system files, and which ones are to be permitted to write to them, and which ones to be permitted to open remote sockets, etc. Note however, that *all servlets signed by a particular user will have the same permissions*.

Servlet-Signer

Throughout this chapter, we use the terms *servlet-signer* and user in servletMgrRealm interchangeably. These two refer to the same thing, really, since a user in the servletMgrRealm is actually a signer of servlets, as we have explained earlier.

To create a user in the servletMgrRealm, here's what has to be done. Note that for saying to ServletExpress to trust all servlets signed by so-and-so to such an extent, we first have to give ServletExpress the public key of the entity to verify the signatures. To register a servlet-signer, you would first have to get the signer to sign a JAR file, put it on a Web server, and give the URL of that JAR file at the time of registering the user. Notice that we didn't say that the signed JAR file must contain a servlet class. In fact we were also able to add a new user under the servletMgrRealm by presenting to ServletExpress a signed JAR file containing a simple text file.

We show you now how to create a new servlet-signer, step by step. First, we went to the Security page under servletMgrRealm.



Realm: servletMgrRealm

admin
narry
unsigned

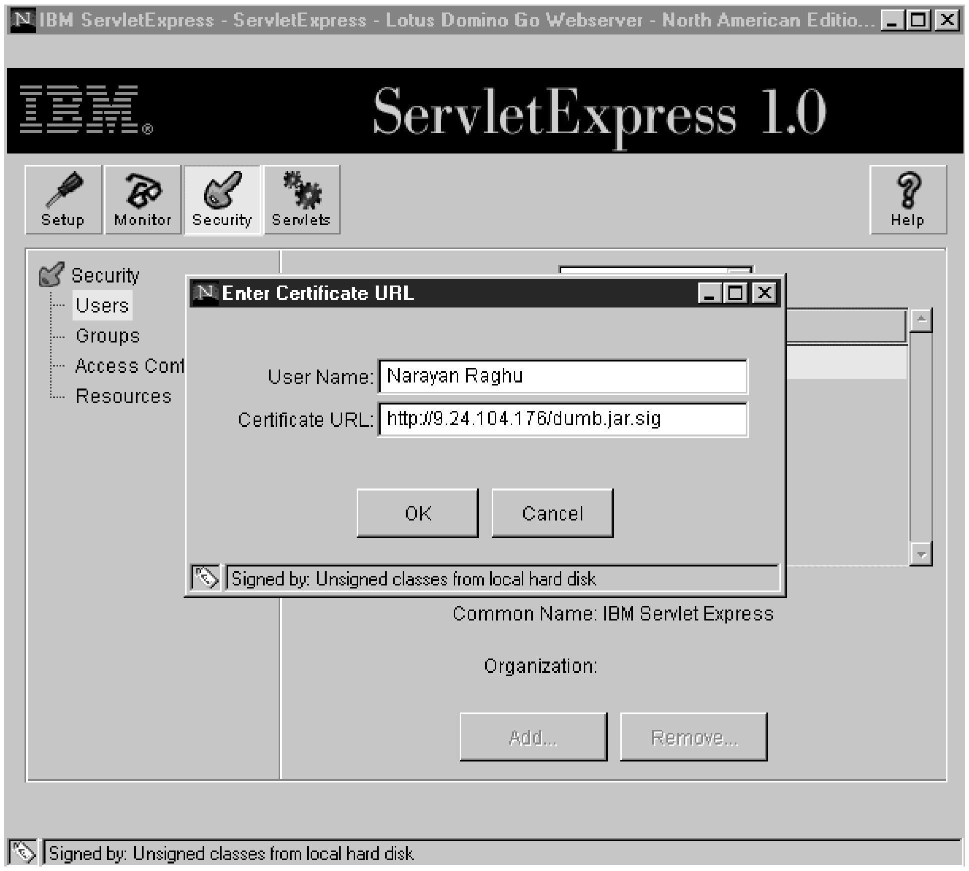
Common Name: IBM Servlet Express

Organization:

Signed by: Unsigned classes from local hard disk

Figure 122. Adding a User in the servletMgrRealm

We clicked on **Add...**. Then we filled in the particulars as shown.



User Name: Narayan Raghu
Certificate URL: http://9.24.104.176/dumb.jar.sig

Signed by: Unsigned classes from local hard disk

Common Name: IBM Servlet Express

Organization:

Signed by: Unsigned classes from local hard disk

Figure 123. Users

The URL `http://9.24.104.176/dumb.jar.sig` that we entered pointed to the location containing the JAR file that had been signed by the particular signer we wanted registered. Three important points must be considered:

1. When you enter the User Name in the above panel, you don't have to type the name that is registered in the certificate. The User Name that you enter here is simply the name of the servlet-signer that ServletExpress identifies as the owner of the certificate.
2. Even if the above panel has a field named Certificate URL, what ServletExpress really expects that you type there is the URL of a JAR file signed by the trusted user you want to add.
3. The signed JAR file is not supposed to contain a servlet class. Actually, you could jar whatever file, even a text file, sign it and use that file to add a user under the servletMgrRealm.

This is enough to add a user under the servletMgrRealm, but now we want to show to you how we created the signed JAR file dumb.jar.sig to which the Certificate URL points.

To produce the JAR file dumb.jar from the class file DumbServlet.class, we used the command:

```
jar cvf dumb.jar DumbServlet.class
```

JAR Files and Signed JAR Files

A discussion on JAR files in JDK 1.2 is found at 2.5.1, "The jar Utility" on page 29. Notice, however, that even if the Java support that our Web server used was still 1.1 (JDK 1.2 was still beta at the time of the project), no differences have been noticed between the `jar` utility of JDK 1.1 and the `jar` utility of JDK 1.2. You can therefore apply the same considerations.

However, the technique to sign a JAR file changes in JDK 1.2 from what it was in JDK 1.1. When Lotus Domino Go Webserver and ServletExpress will support JDK 1.2, you will have to use the new security features and tools provided by JDK 1.2. The new technique to sign JAR files in JDK 1.2 is discussed in 2.5.3, "The jarsigner Utility" on page 34.

In order to add a servlet-signer to the servletMgrRealm, we needed to use the `javakey` command line tool. The javakey.exe file comes with the JDK in the bin directory below JAVA_HOME. It is a command line utility that can create entities, designate them trusted or untrusted, create DSA key pairs, sign certificates, etc. Again, type `javakey` to get all the possible options. We recommend the JavaSoft Web site http://www.javasoft.com/security/usingJavakey.html, which deals with how `javakey` is to be used. However, you can see how we did it now.

First of all, we created the entity narry through the following command:

```
javakey -cs "narry" true
```

We then generated a key pair for the entity:

```
javakey -gk "narry" DSA 512 narry_pub narry_priv
```

Then we created a certificate directive file as shown, and stored it in the file cert.direc:

```
issuer.name=narry
issuer.cert=1
subject.name=narry
subject.real.name=Narayan Raghu
subject.org.unit=ITSO
subject.org=IBM
subject.country=US
start.date=02 Jun 1998
end.date=01 Jun 1999
serial.number=1500
out.file=narry.x509
```

Then we created a certificate using:

```
javakey -gc cert.direc
```

We had already created a JAR file, named dumb.jar, from the class file DumbServlet.class. We then created a signature directive file like the one shown, and stored it in sign.direc:

```
signer=narry
cert=1
chain=0
signature.file=narrySig
```

Then, finally, we signed the JAR file dumb.jar, using:

```
javakey -gs sign.direc dumb.jar
```

to obtain the file dumb.jar.sig. Notice that a signed JAR file in JDK 1.1 carries the double extension .jar.sig, but in JDK 1.2 JAR files keep the extension .jar even when they are signed. Keep this difference in mind for when Lotus Domino Go Webserver and ServletExpress will have a full JDK 1.2 support. You can see 2.5.3, “The jarsigner Utility” on page 34 for further details.

Note that we used the DSA algorithm, which ships free with JDK. The common standard for X.509 certificates is, however, RSA (though this is not *required* by the standard).

In this way we have just shown you the full process necessary to create a new user, or serlvet-signer, under the servletMgrRealm. As you can see, if you want to add a user to servletMgrRealm, you have no other choice than to generate a signed JAR file and then pointing to its URL, even if it is not required that the signed JAR file contains a servlet class. As we said, a signed JAR file containing a text file works fine anyway.

What about new users in the NT or UNIX realms? These two realms are used to give access to users who already have an ID on the UNIX or NT machine, to the servlet resources on the server. You will not be able to create users here, since ServletExpress picks up information about the UNIX or NT users from the operating system it's running on. We'll talk more about this when we talk about ACLs.

At this stage, the only form of authentication of users supported is the basic authentication, with user ID and password, and the only realm that can be used to store user profiles, such as which servlets the user is permitted to run, and which of these servlets can access system resources, is the defaultRealm, unless you wouldn't mind programming on ServletExpress using the JDK APIs.

The *user* that you can create in the servletMgrRealm is used only for signing servlets that are to be remotely loaded. In more unambiguous terms, the *user* in the defaultRealm is *not* the same as, or even similar in functionality to the *user* in the servletMgrRealm, the former being a client accessing the resources the server and the servlets have to offer, and the latter, a person whose signatures on JAR files containing servlets the ServletExpress has been instructed to trust. At this stage, we'd like to mention that for obvious reasons, it is not possible to create users in the NT or UNIX realm. These users are read off the underlying operating system.

4.1.3 Groups

Let's now see how to create groups. To begin with, let's stick to defaultRealm. Click on the **Groups** link, and it'll take you to the groups page, as shown.

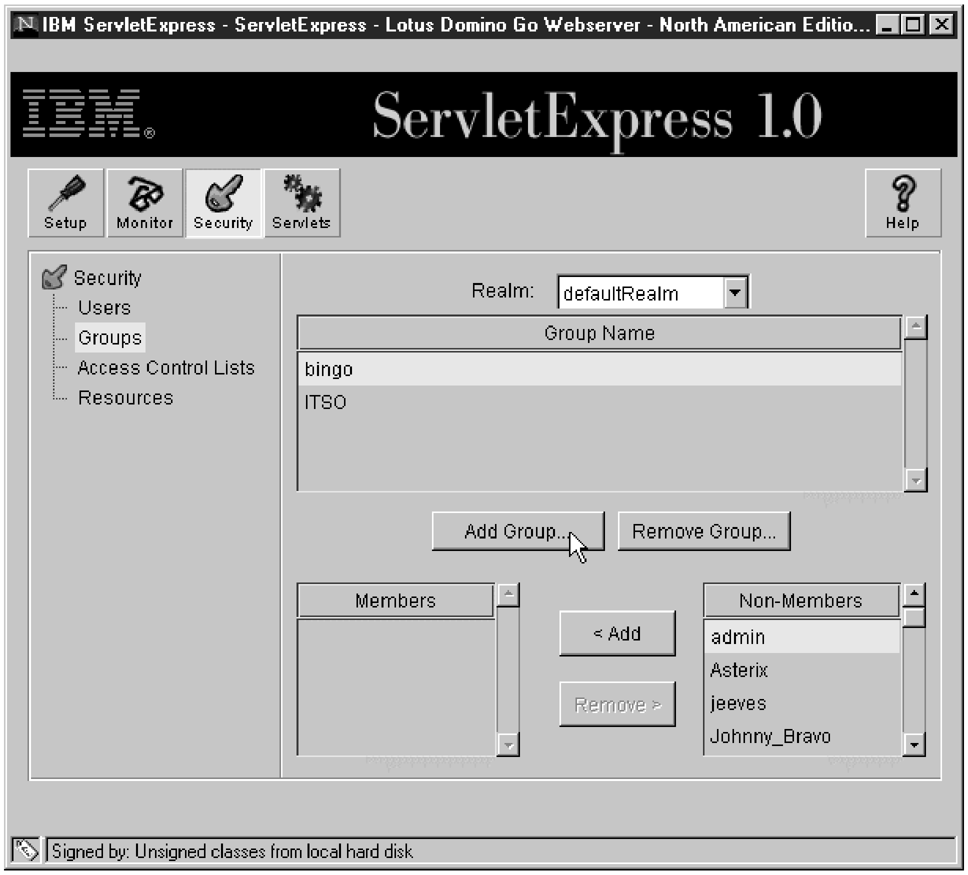


Figure 124. Groups

As you can see, two groups have been created. You can create another one by clicking on the **Add Group...** button. You can enter the name in the message box, and press **Add**, as shown.

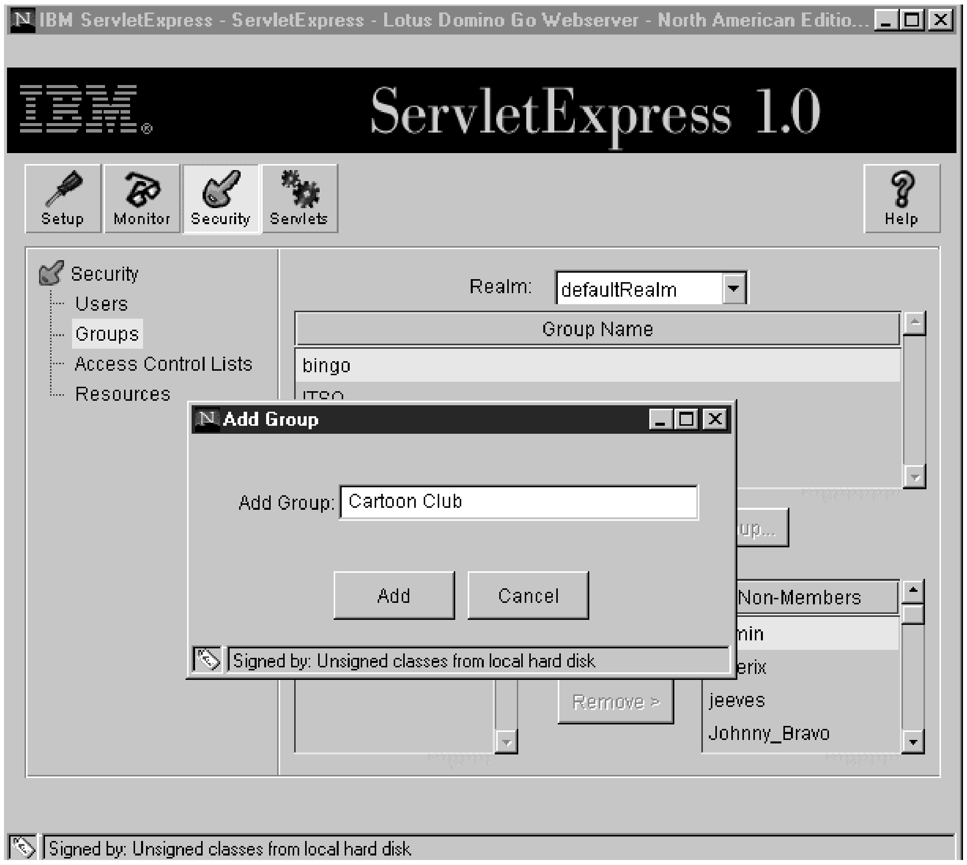
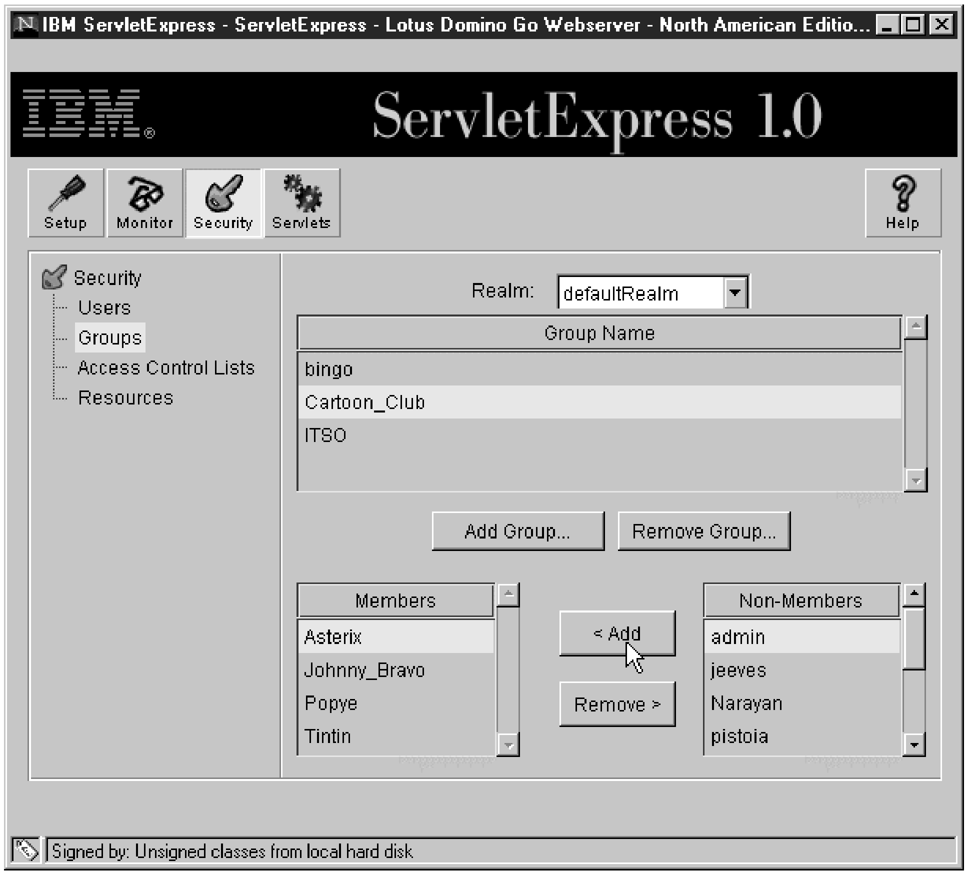


Figure 125. Add Groups

Groups are not really entities in themselves. They are used to make it easier for the administrator to give or revoke permissions to a number of people at the same time. Hence, we now have to add users into groups. Note that a user can belong to more than one group, and a group can have any number of users. In the beginning, all the users within the realm are non-members of a newly created group, meaning that they appear in the Non-Members list. The admin can choose to add any number into the group by clicking on the **Add** button, and remove any number by clicking on the **Remove** button. Note that the union of the users in the Members list and the users in the Non-Members list is the list of users in the realm. Also note that no user can be a member and a non-member at the same time - in other words, the intersection of the two sets is always a null set.



Realm: defaultRealm

Group Name
bingo
Cartoon_Club
ITSO

Members
Asterix
Johnny_Bravo
Popye
Tintin

Non-Members
admin
jeeves
Narayan
pistoia

Signed by: Unsigned classes from local hard disk

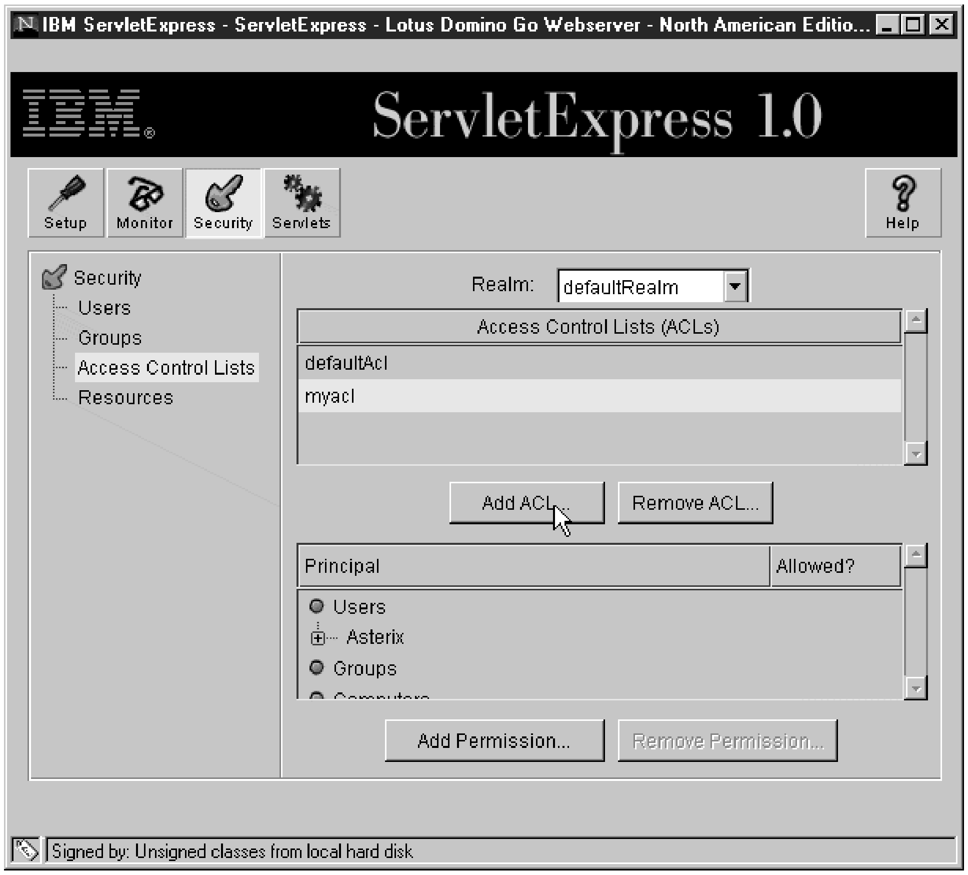
Figure 126. Add to and Remove Users from Groups

Now, if you are as inquisitive as we were, and want to go to the file and check out how this information is stored, go to the directory *servletexpress_root*\realm\data\defaultRealm and type out the file with the same name as the newly created group. We typed the file D:\ServletExpress\realm\data\defaultRealm\Cartoon_Club.grp and we got a listing of the user names we just added into the group. Notice that this file cannot be opened by double clicking on its icon displayed by Windows NT Explorer, since .grp is the extension of Windows Program Manager Group files, and obviously, this file will not be in the format of a Program Manager Group file. You need to open it with a text editor directly or, alternately, you could rename this file's extension, double click on the icon, and be sure to change back the name before starting the server.

You will notice that a new file is created for each group, and each new user is added to the file named keyfile in the appropriate directory. You could try to edit these files yourself, and try crashing the system, but we really didn't have the inclination to do that, and so cannot predict the result. Note that groups cannot be created in the servletMrgRealm.

4.1.4 Access Control Lists

Now, let's tackle one of the more important parts of the ServletExpress setup - the ACLs. The administrator can add ACLs or delete existing ACLs in a given realm. The addition of an ACL is pretty similar to the addition of users and groups. We now show one just for completeness. You select **Access Control List** from the Security tree and then click on **Add ACL...**:



Realm: defaultRealm

Access Control Lists (ACLs)
defaultAcl
myacl

Principal	Allowed?
Users	
Asterix	
Groups	

Signed by: Unsigned classes from local hard disk

Figure 127. Security - Access Control Lists Page

You can type the name in the message box and then press **Add**.

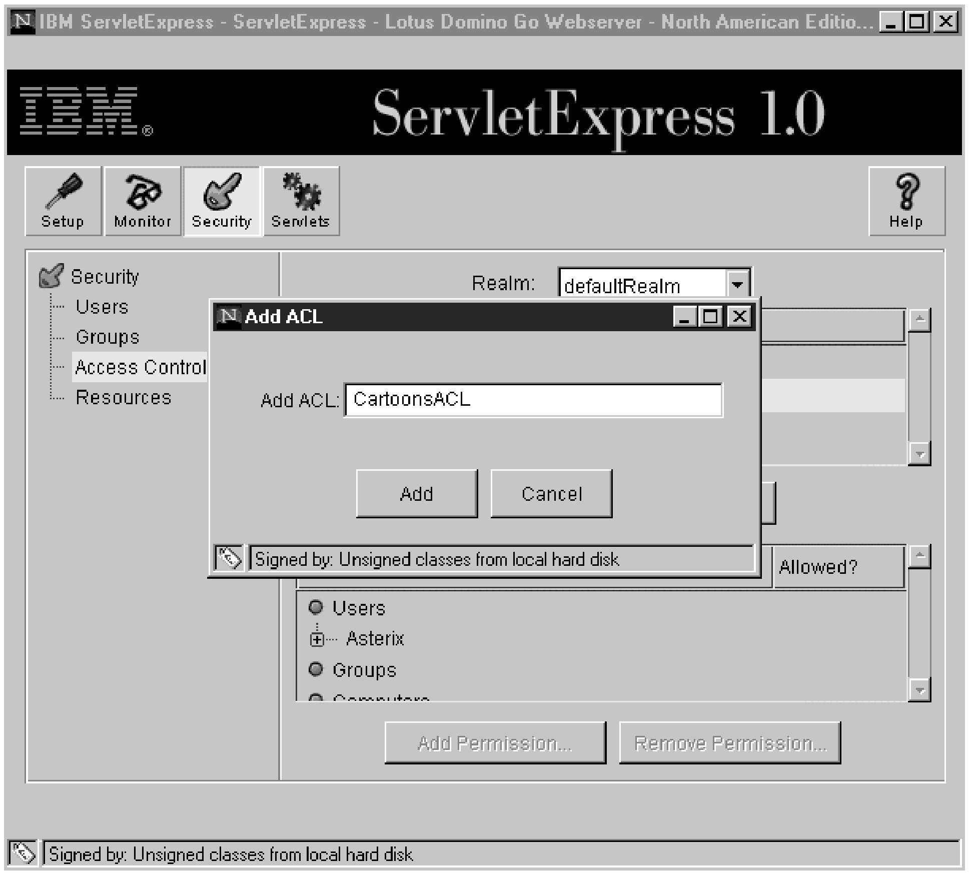
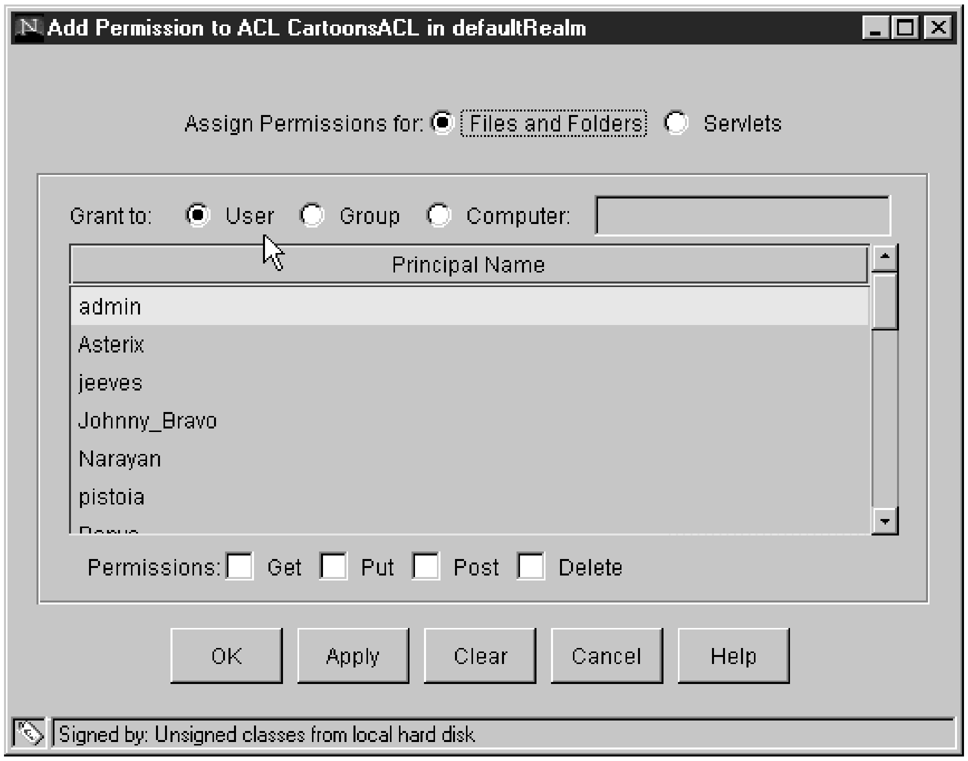


Figure 128. Addition of ACLs

Now, what does one *do* with an ACL? One simply adds or removes permissions for certain users and groups for certain resources in the ACL. Note that all this happens *only* within the realm, and it is not possible to add a user registered under a different realm to an ACL in this one. Let's now try to add some permissions. Click on the button **Add Permission...**. You should see the following three screens, depending on whether you have marked **User**, **Group** or **Computer**:



Assign Permissions for: [✓ Files and Folders] Servlets

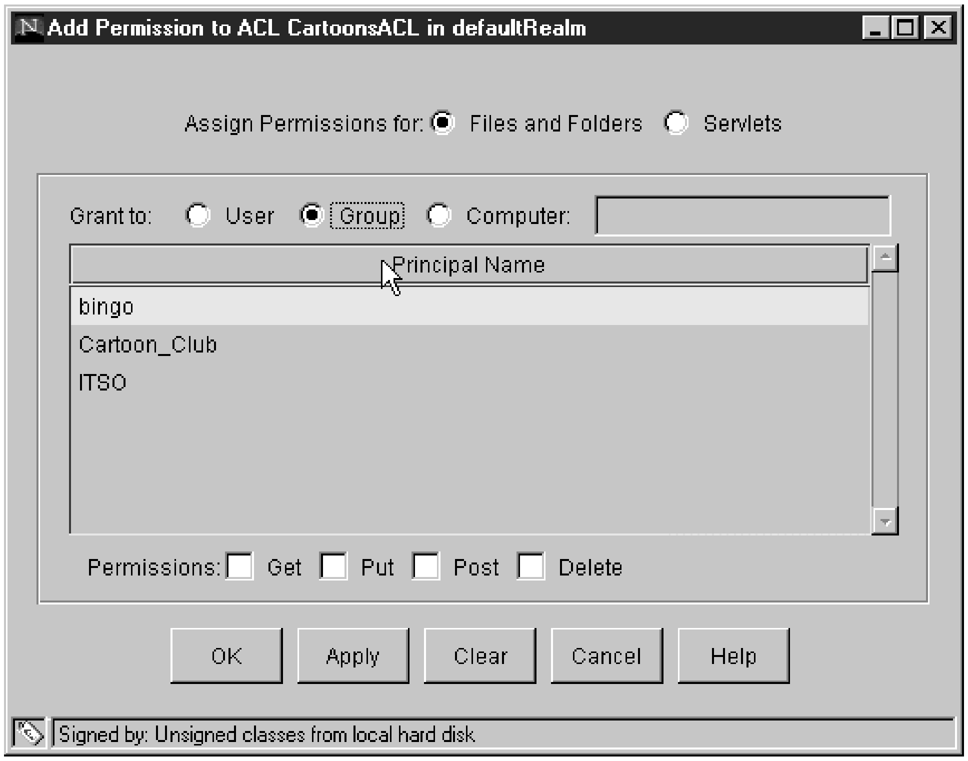
Grant to: [✓ User] Group Computer:

Principal Name
admin
Asterix
jeeves
Johnny_Bravo
Narayan
pistoia

Permissions: ☐ Get ☐ Put ☐ Post ☐ Delete

Signed by: Unsigned classes from local hard disk

Figure 129. Adding File Access Permissions to Users



Assign Permissions for: [✓ Files and Folders] Servlets

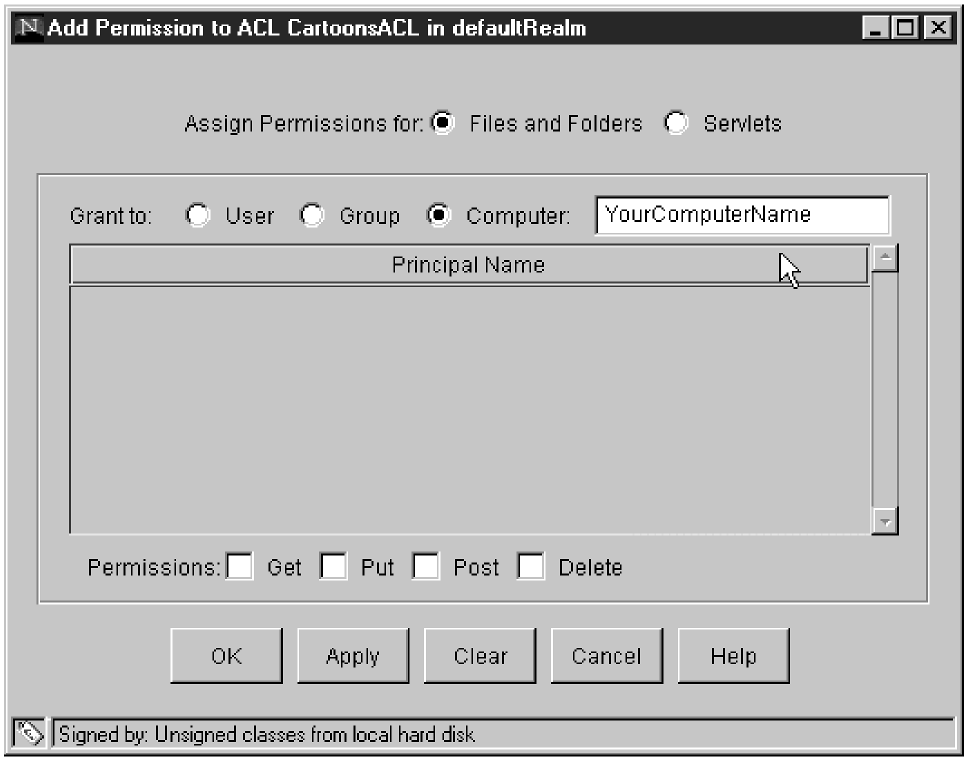
Grant to: User [✓ Group] Computer:

Principal Name
bingo
Cartoon_Club
ITSO

Permissions: ☐ Get ☐ Put ☐ Post ☐ Delete

Signed by: Unsigned classes from local hard disk

Figure 130. Adding File Access Permissions to Groups


Add Permission to ACL CartoonsACL in defaultRealm

Assign Permissions for: [✓ Files and Folders] Servlets

Grant to: User Group [✓ Computer:] YourComputerName

Principal Name

Permissions: Get Put Post Delete

Signed by: Unsigned classes from local hard disk

Figure 131. Adding File Access Permissions a Specific Computer

Note the file access permissions you can give to users, groups, and specified computers on the network. The file permissions are the same as the HTTP methods of the form handling protocol:

1. GET

   This method is used for sending form data. The data is sent to the server from the client by appending the data in the URL field. The same stream is used for the requested URL and the form data.

2. PUT

   This is used to put files on the server. You might not have an access to this method from a normal browser. However, most HTML authoring tools use this method to put an edited HTML page onto the server.

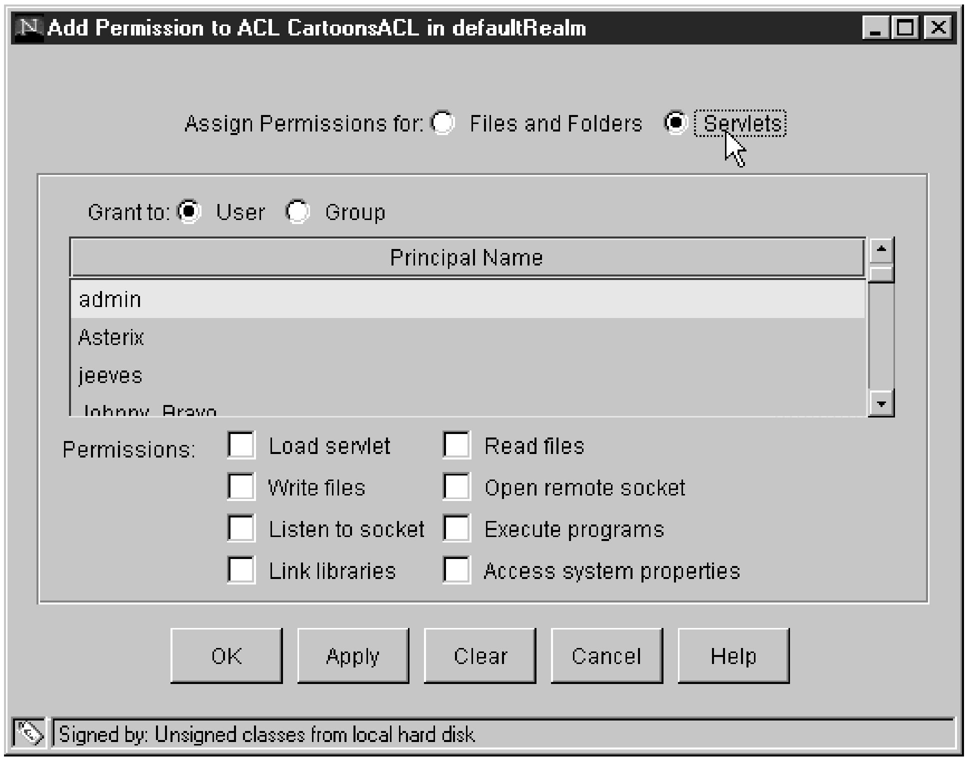
3. POST

   As the submitted form data got larger, it was found that the URL field was inadequate to handle this data. Further, it is really not very graceful to clutter up the URL window. The POST method is used to send data from the client to the server. It is different from the GET method in that another stream is used to send the data.

4. DELETE

   This option also, like the PUT method, might not be available from your normal client. Moreover, though specified in the HTTP protocol (see http://www.w3.org/Protocols/), most servers do not grant permission for this action. This is used to delete a file from a server. This method might be supported by some Web authoring clients, but, as mentioned before, might not be supported by your server.

Let's now add some permissions both to users and groups, and see if they work. But before we verify if they work or not, we have to associate a resource with them. So, let's go ahead with granting permissions first. The following screens are those of us granting permissions to users, groups and computers. Since these are pretty much self explanatory, you probably wouldn't like redundant sentences between them. However, in the rest of the chapter you will see how these permissions really work.


Add Permission to ACL CartoonsACL in defaultRealm

Assign Permissions for: ( ) Files and Folders (●) Servlets

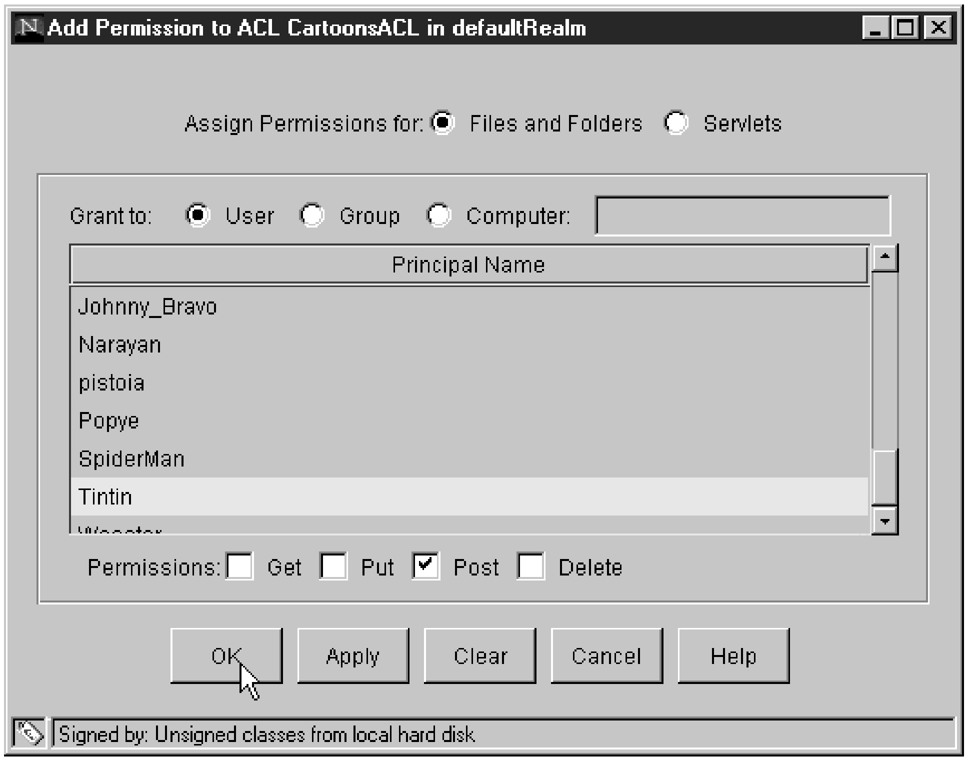
Grant to: (●) User ( ) Group

Principal Name
admin
Asterix
jeeves
Johnny Bravo

Permissions: [ ] Load servlet [ ] Read files [ ] Write files [ ] Open remote socket [ ] Listen to socket [ ] Execute programs [ ] Link libraries [ ] Access system properties

Signed by: Unsigned classes from local hard disk

Figure 132. Adding No Servlet Permission to User admin


Add Permission to ACL CartoonsACL in defaultRealm

Assign Permissions for: [✓ Files and Folders] Servlets

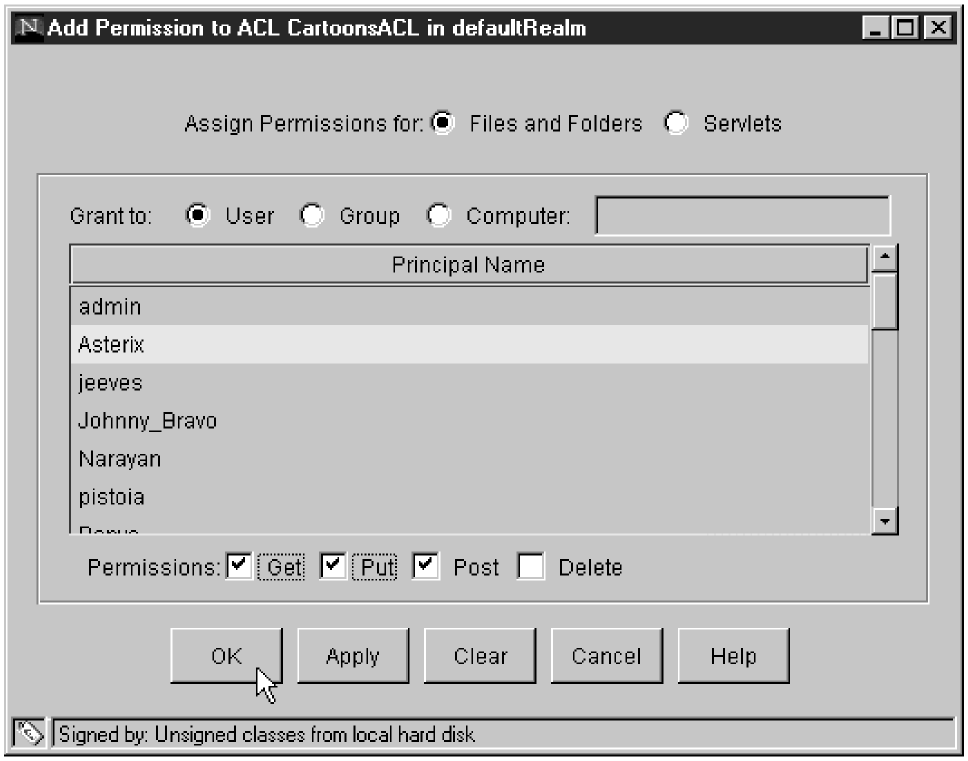
Grant to: [✓ User] Group Computer:

Principal Name
Johnny_Bravo
Narayan
pistoia
Popye
SpiderMan
Tintin

Permissions: Get Put [✓ Post] Delete

Signed by: Unsigned classes from local hard disk

Figure 133. Adding File Post Permission to User Tintin


Add Permission to ACL CartoonsACL in defaultRealm

Assign Permissions for: [✓ Files and Folders] Servlets

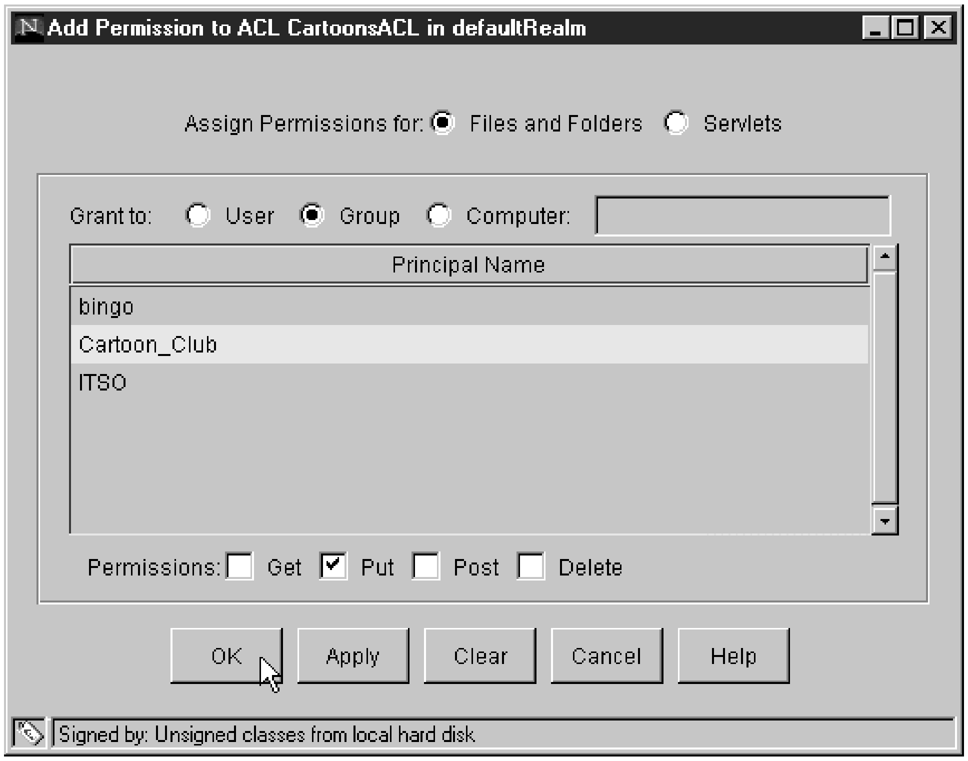
Grant to: [✓ User] Group Computer:

Principal Name
admin
Asterix
jeeves
Johnny_Bravo
Narayan
pistoia

Permissions: [✓ Get] [✓ Put] [✓ Post] Delete

Signed by: Unsigned classes from local hard disk

Figure 134. Adding File Permissions Get, Put and Post to User Asterix


Add Permission to ACL CartoonsACL in defaultRealm

Assign Permissions for: [✓ Files and Folders] Servlets

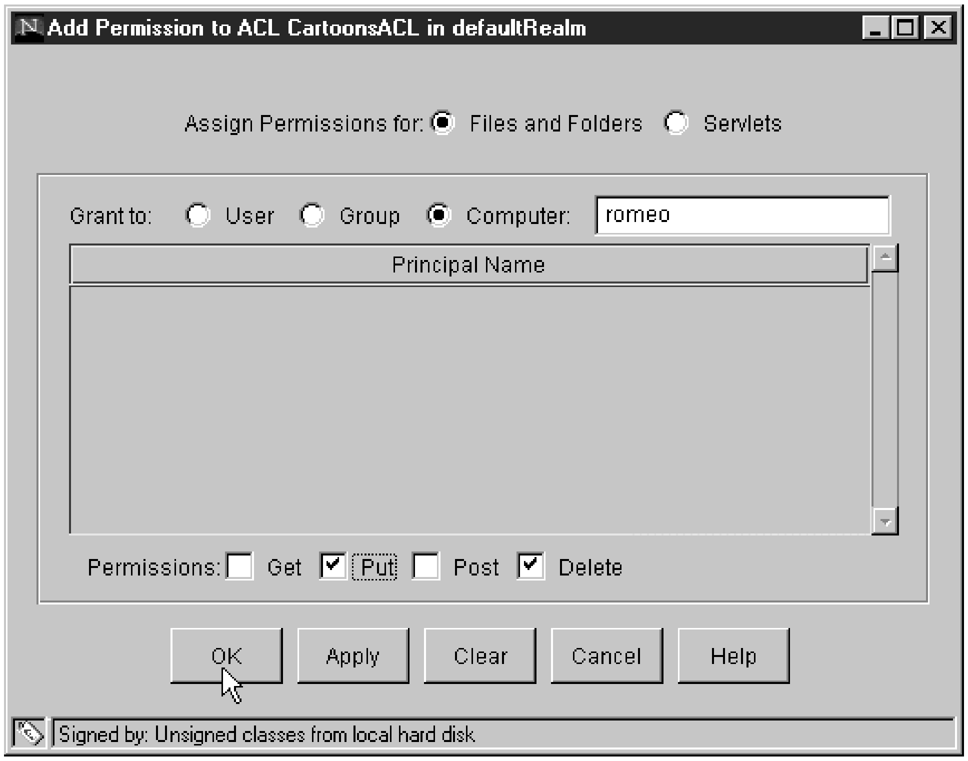
Grant to: User [✓ Group] Computer:

Principal Name
bingo
Cartoon_Club
ITSO

Permissions: ☐ Get ☑ Put ☐ Post ☐ Delete

Signed by: Unsigned classes from local hard disk

Figure 135. Adding File Put Permission to Group Cartoon_Club


Add Permission to ACL CartoonsACL in defaultRealm

Assign Permissions for: [✓ Files and Folders] Servlets

Grant to: User Group [✓ Computer:] romeo

Principal Name

Permissions: ☐ Get ☑ Put ☐ Post ☑ Delete

Signed by: Unsigned classes from local hard disk

Figure 136. Adding File Permissions Put and Delete to Computer romeo

Note that all the permissions were added in one particular ACL, the CartoonsACL that we had defined. Now, if you come over to the main screen and look at the bottom window, you can get complete information about how the access controls are set. Users, groups and computers for which permissions were set appear in the Principal tree.

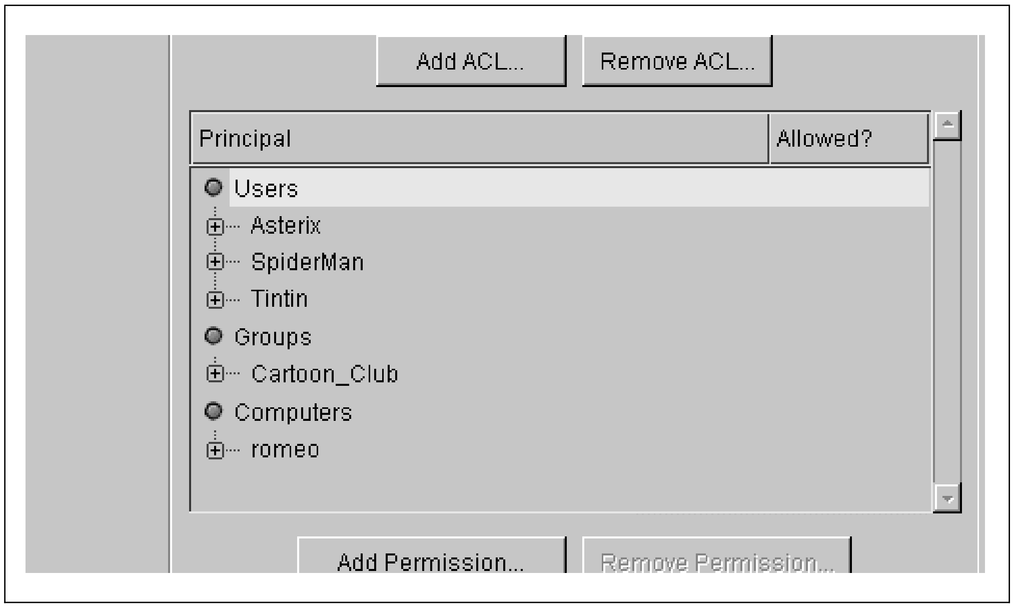
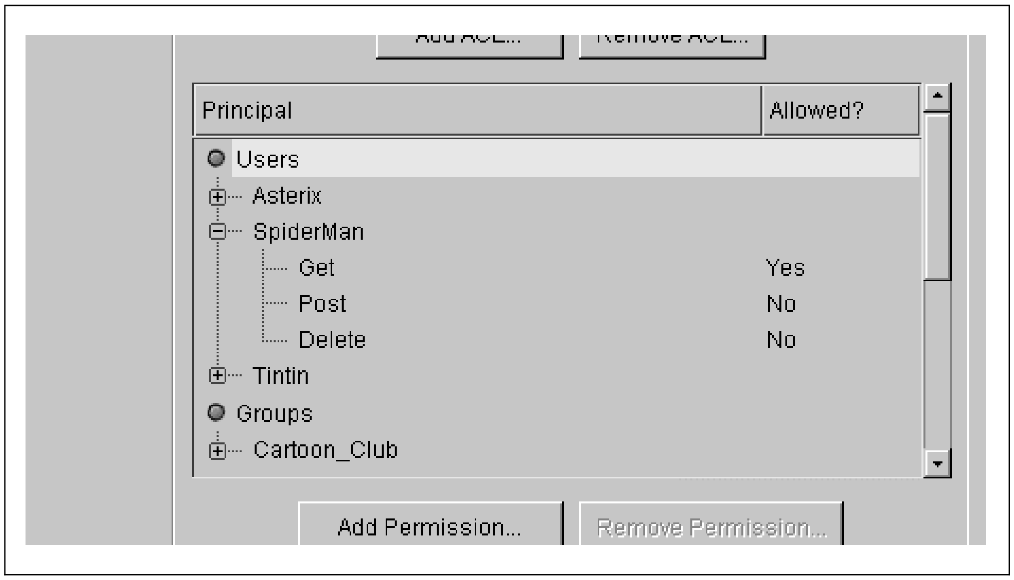


Figure 137. Principal Tree

You must click on the plus sign (**+**) beside each item on the list to see the specific permissions.



Principal	Allowed?
Users	
Asterix	
SpiderMan	
Get	Yes
Post	No
Delete	No
Tintin	
Groups	
Cartoon_Club	

Figure 138. Checking Specific Permissions

Notice that Figure 136 on page 144 shows the particular case when you want to allow access only from specific computers. In the Computer field, you can enter the name of the host either as a host name or as an IP address. You can use the wildcard character `*` when entering a host name, for example `*.com`. Requests that originate from hosts other than the specified one will be denied.

Access Controls in Case of Conflict

We know now that a user can belong to more than one group. This brings about a possibility that by the virtue of belonging to one group, the user might get a particular privilege, and by the virtue of belonging to another, he might be denied the same permission. So how do we solve this paradox? There is a specific method of calculating permissions, and this is discussed under 4.1.5, "Resources" on page 146.

Note that you cannot add ACLs in the servletMgrRealm. However, you can add permissions to either servlets or files in the servletACL. The servletACL is the only ACL that is available in the servletMgrRealm. This is created automatically, and cannot be deleted. Here is where you would have to indicate what permissions you would like to give to servlets signed by a particular user in the servletMgrRealm. We will deal more with this in later sections.

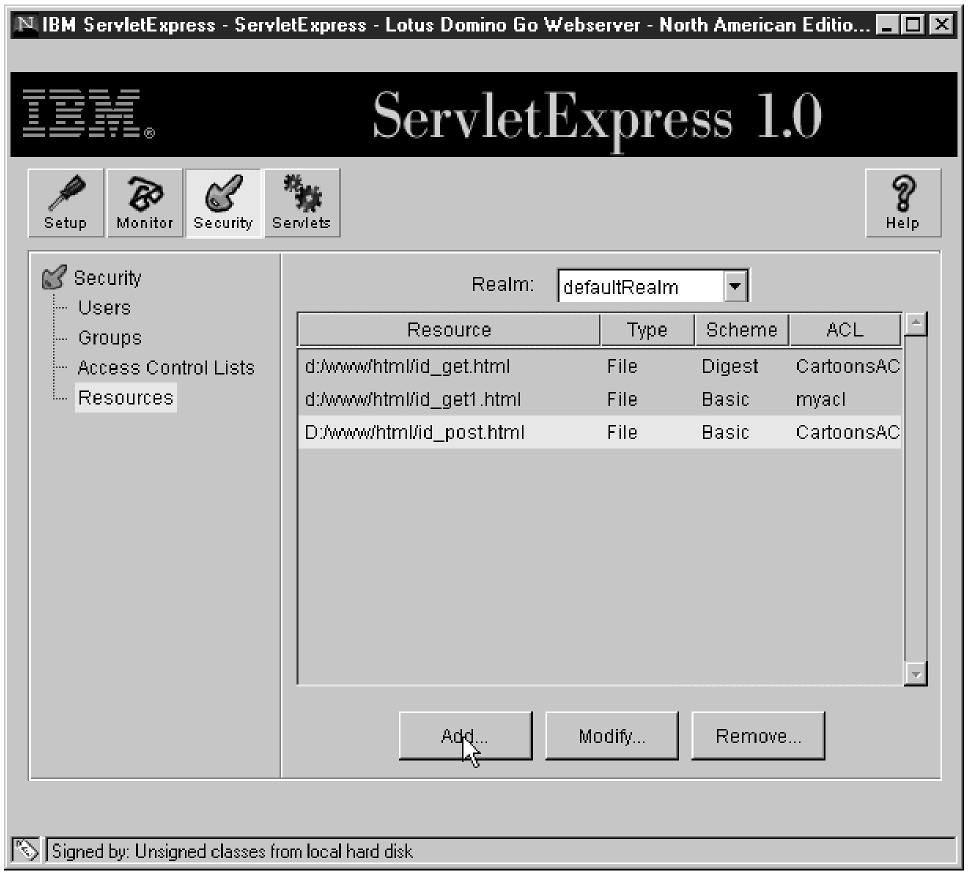
The NT or UNIX realms, however, behave pretty much like the defaultRealm, except for the fact that their user list cannot be modified. Notice that ACLs can be added to NT and UNIX realms.

4.1.5 Resources

A *resource*, in this context, is essentially a service the Web server has to offer, which needs to be protected. HTML pages and servlets that needn't be protected are not considered *resources* in this context. In general, to restrict access to static HTML pages, you would have to use the facilities provided by Domino Go Web server, or any other Web server you might be using, and you will use ServletExpress to control access to servlets. However ServletExpress, via the servletMgrRealm, allows the administrator to protect also the resources that servlets can access, such as files and socket connections. This could create a conflict with the underlying Web server, in that access to a Web page could be denied by the Web server and allowed to a servlet by ServletExpress. In general, the relationship between the Web server and ServletExpress about conflicting permissions is as follows. If the Web server first denies access to a resource, then access is denied to ServletExpress too. If the Web server permits the access, then it is up to ServletExpress to protect the resource. In other words, it is the Web server that takes precedence.

Put briefly, access control in the ServletExpress security model is all about how ServletExpress gives *users* and *groups* access to several of the *resources* using predefined *ACLs*.

The administrator of ServletExpress can add new resources, in the sense that he or she can declare more of the existing resources to be protected. We tried this out, and to do so, we first went to the Resources screen, which for the defaultRealm looked like the following figure, and clicked on **Add...**.

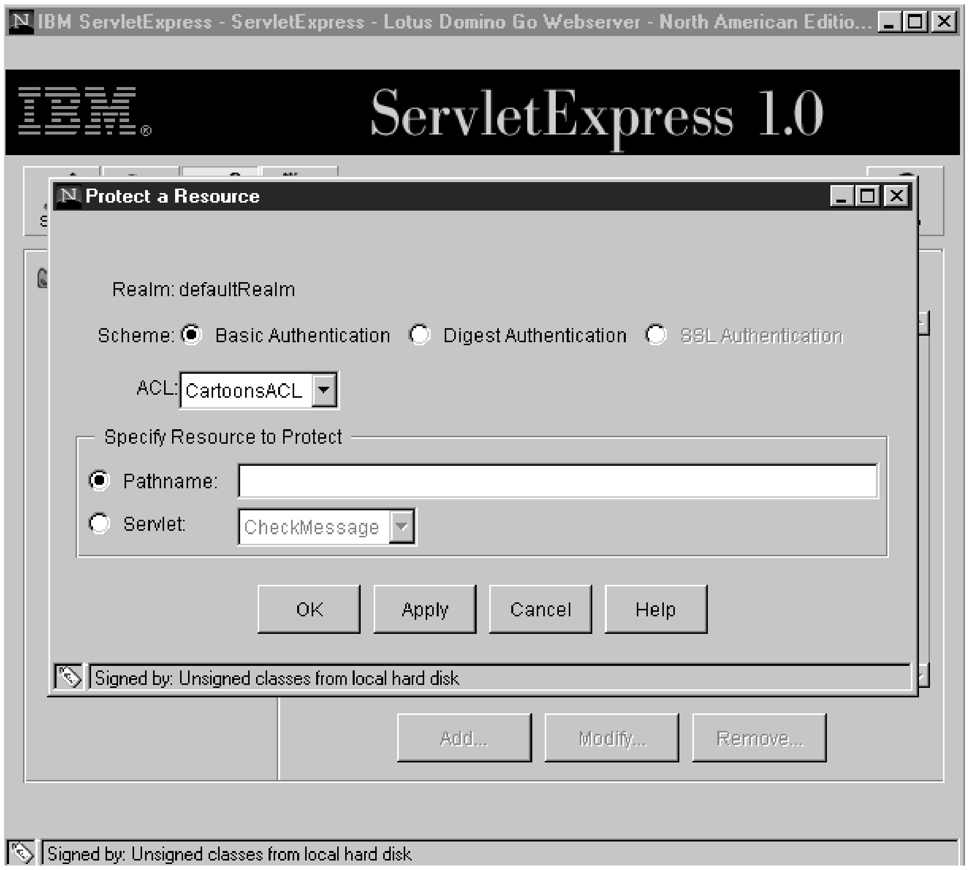

IBM ServletExpress - ServletExpress - Lotus Domino Go Webserver - North American Editio...
IBM
ServletExpress 1.0
Setup
Monitor
Security
Servlets
Help
Security
Users
Groups
Access Control Lists
Resources
Add...
Modify...
Remove...
Signed by: Unsigned classes from local hard disk

Realm: defaultRealm

Resource	Type	Scheme	ACL
d:/www/html/id_get.html	File	Digest	CartoonsAC
d:/www/html/id_get1.html	File	Basic	myacl
D:/www/html/id_post.html	File	Basic	CartoonsAC

Figure 139. ServletExpress Resources

The second screen shows the kind of resource you can add to an ACL.


IBM ServletExpress - ServletExpress - Lotus Domino Go Webserver - North American Editio...
IBM
ServletExpress 1.0
Protect a Resource
OK
Apply
Cancel
Help
Add...
Modify...
Remove...

Realm: defaultRealm

Scheme: [✓ Basic Authentication] Digest Authentication SSL Authentication

ACL: CartoonsACL

Specify Resource to Protect

[✓ Pathname:]

Servlet: CheckMessage

Signed by: Unsigned classes from local hard disk

Signed by: Unsigned classes from local hard disk

Figure 140. Adding Resources to ACLs

The screen begins by confirming that we are still in the defaultRealm. The next line shows the type of authentication. Basic Authentication will ask the client to key in a user ID and password, which will be sent to the server over the network. The Digest Authentication will do the same, but the information will be sent to the server in an encrypted form over the net. SSL Authentication, which we expect that you will find disabled on your screen, will authenticate the user through their client certificate. However, this is not supported in the simple installation of ServletExpress. You could gain this functionality by writing your own realm, using the JDK APIs.

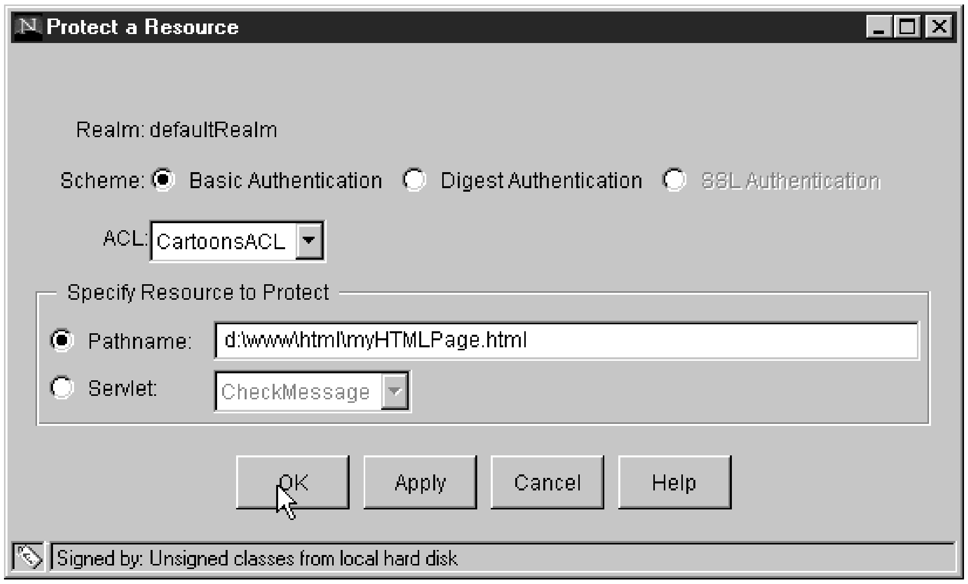
Next is a drop down list box containing a list of all the ACLs in the realm. ServletExpress permits us to add a particular resource to only one ACL, since if the same resource is in more than one ACL, there might be conflicts in permissions. We tried adding a particular resource that already belonged to one ACL, to yet another ACL, and we found that the resource now belonged to the new ACL, but not to the older one. In other words, status of a resource can be overwritten.

ServletExpress and Windows NT

Note that there is a problem with the NT version of ServletExpress. Windows NT is not case sensitive, but ServletExpress is. So if you try to give a file permission, and enter the complete location as `d:\something\nuts.txt`, you could add yet another entry in another ACL as `D:\sOmEtHiNg\NuTs.TxT`. Although this would refer to the same file in NT, it would be considered as two separate resources by ServletExpress. This could cause several problems.

We recommend sticking to one convention while entering the file names in the Resources form, thus ensuring that duplicate entries do not get in by mistake.

Next is the option to choose the resource to protect. The administrator can choose between a servlet and a file. If the resource is a file or a directory containing HTML pages, the administrator is to key in the complete path name.



Realm: defaultRealm

Scheme: [✓ Basic Authentication] Digest Authentication SSL Authentication

ACL: CartoonsACL

Specify Resource to Protect

[✓ Pathname:] d:\www\html\myHTMLPage.html

Servlet: CheckMessage

Signed by: Unsigned classes from local hard disk

Figure 141. Adding a File Resource

If the resource in question is a servlet, it can be selected from the drop down list box.

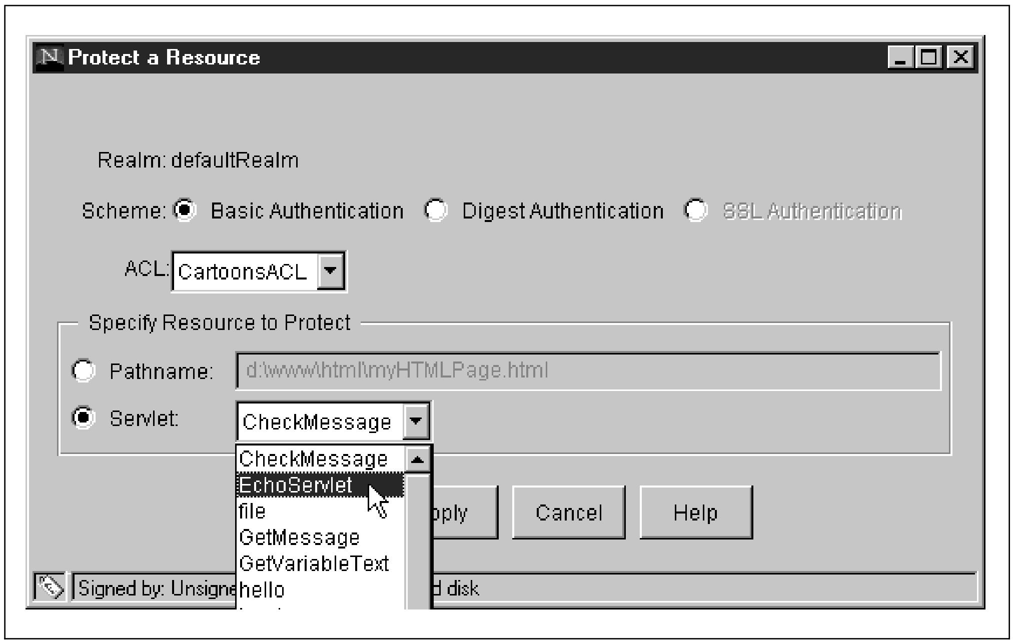
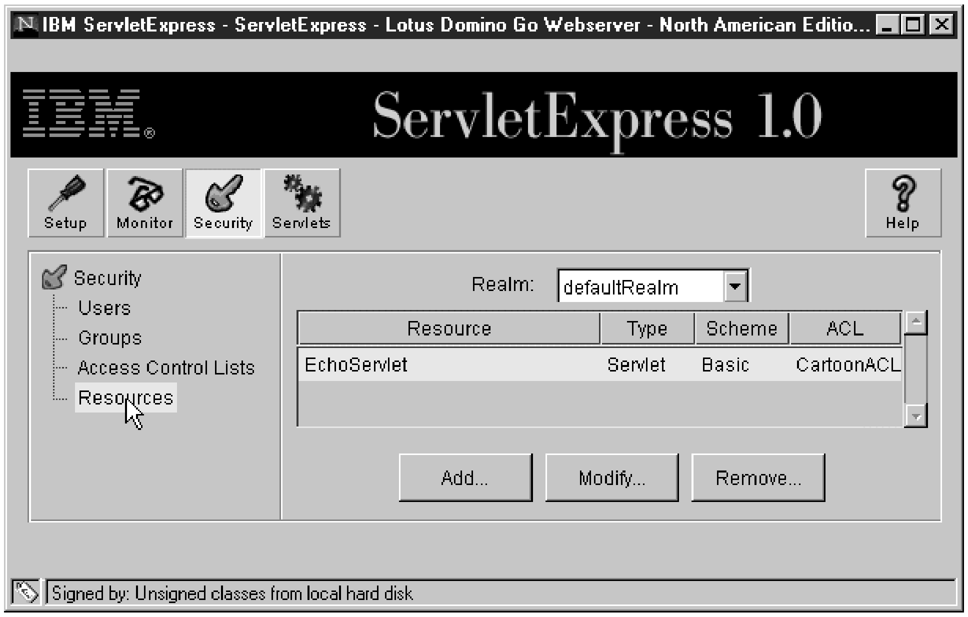

Protect a Resource
Realm: defaultRealm
Scheme: [✓ Basic Authentication] Digest Authentication SSL Authentication
ACL: CartoonsACL
Specify Resource to Protect
Pathname: d:\www\html\myHTMLPage.html
[✓ Servlet:] CheckMessage
CheckMessage
EchoServlet
file
GetMessage
GetVariableText
hello
pply
Cancel
Help
Signed by: Unsigne
d disk

Figure 142. Adding a Servlet Resource

The help link from this section is very meaningful. You can read it by clicking on the **Help** button.

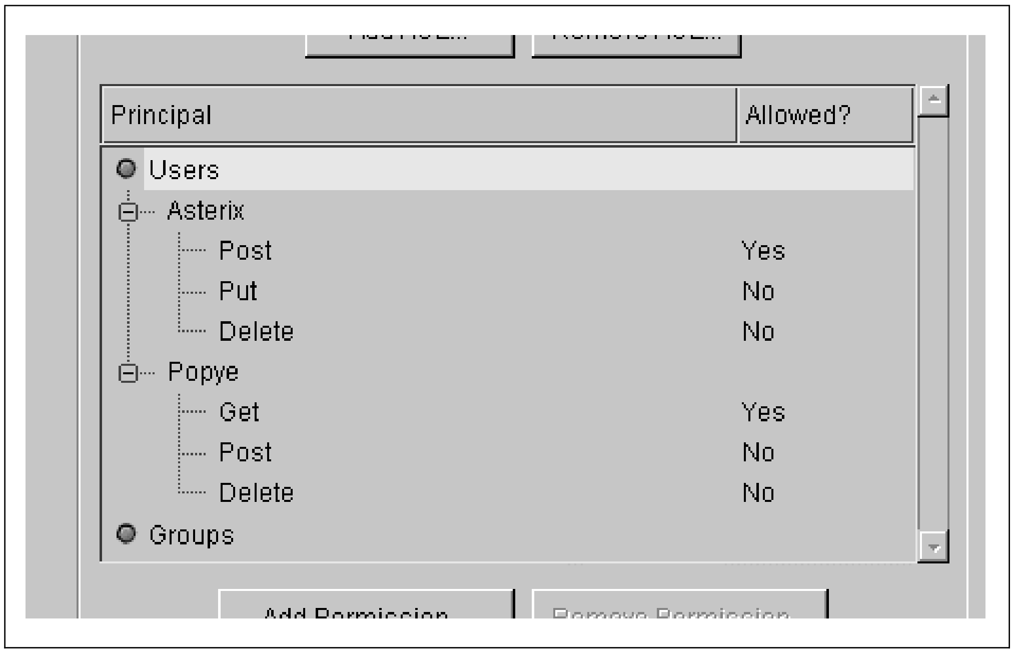
Now let's try out the real stuff. We set an ACL called the CartoonACL, and registered EchoServlet as a resource in the CartoonACL, as shown in Figure 142. The Resources window looks somewhat like this:



Resource	Type	Scheme	ACL
EchoServlet	Servlet	Basic	CartoonACL

Figure 143. The Resources Window

Further, within the CartoonACL, we gave the user Asterix permission *only* to POST, and the user Popye permission *only* to GET. At this stage, the ACL window looks somewhat like this:



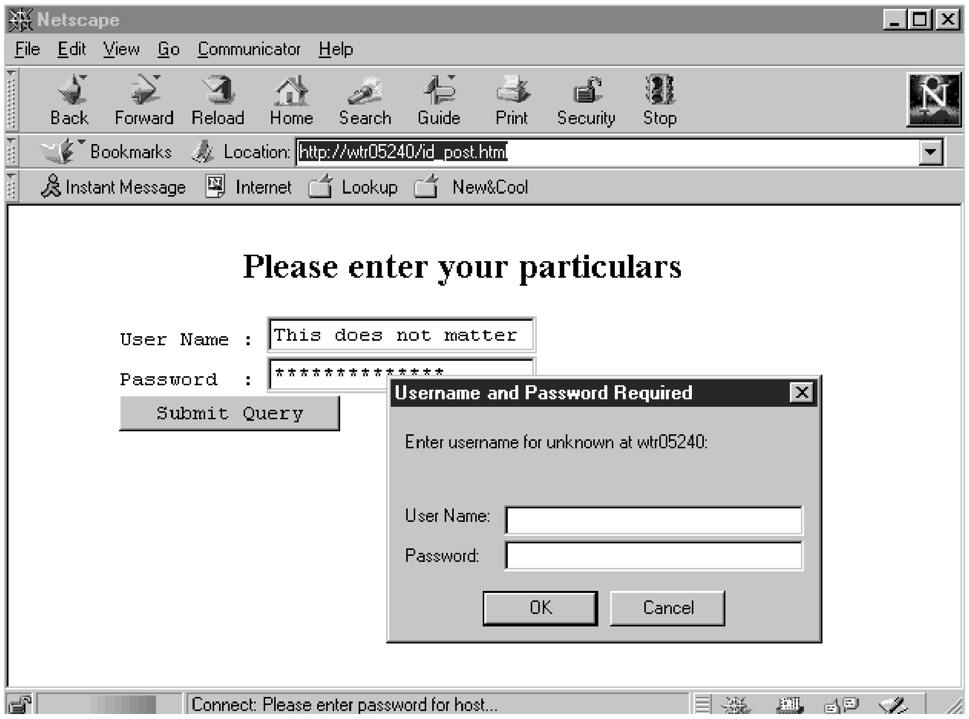
Principal	Allowed?
Users	
Asterix	
Post	Yes
Put	No
Delete	No
Popye	
Get	Yes
Post	No
Delete	No
Groups	

Figure 144. The ACL Window

You would do well at this point to refer back to the source code of EchoServlet, which we wrote earlier in this chapter to test the POST and GET methods (see Figure 90 on page 95, Figure 94 on page 98 and Figure 91 on page 96).

Let's now try to access the EchoServlet, through the pages id_post.html and id_get.html, which use, respectively, the POST and GET methods to communicate with the Web server. We opened up the Netscape window to access the HTML pages, and not surprisingly, there was no trouble getting to the pages themselves (to restrict access to static pages, you'd have to use the ACL functionality supported by Domino Go Webserver).

When we clicked on **Submit** from the page id_post.html, it popped up a window asking for User Name and Password. Note that the page containing the same fields (User Name and Password) is purely coincidental, and has nothing to do with access restrictions at all.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/id_post.html
Instant Message Internet Lookup New&Cool

Please enter your particulars

User Name : This does not matter

Password : **************

Submit Query

Username and Password Required

Enter username for unknown at wtr05240:

User Name:

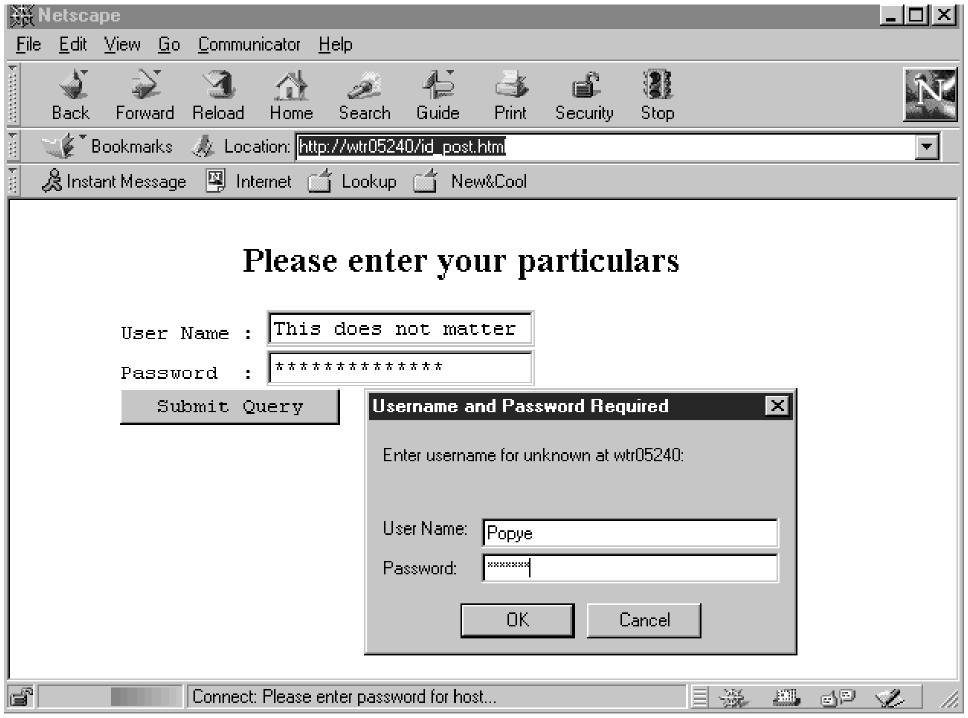
Password:

OK Cancel

Connect: Please enter password for host...

Figure 145. Asking for Authentication

We entered the user ID `Popye` and his password `spinach`. The password appeared hidden by a sequence of asterisks. Remember that Popye had permission to GET and the id_post.html page uses the POST method.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/id_post.html
Instant Message Internet Lookup New&Cool

Please enter your particulars

User Name : This does not matter

Password : **************

Submit Query

Username and Password Required

Enter username for unknown at wtr05240:

User Name: Popye

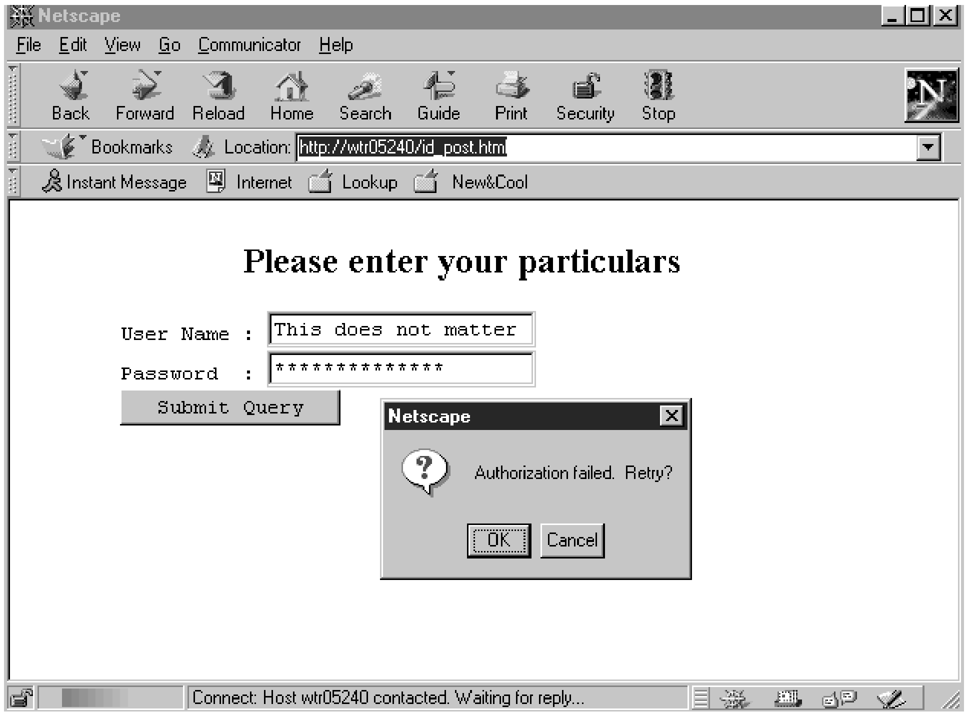
Password: *******

OK Cancel

Connect: Please enter password for host...

Figure 146. Entering Unauthorized User ID and Password

And we got the following screen:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/id_post.html
Instant Message Internet Lookup New&Cool

Please enter your particulars

User Name : This does not matter

Password : ***************

Submit Query

Netscape

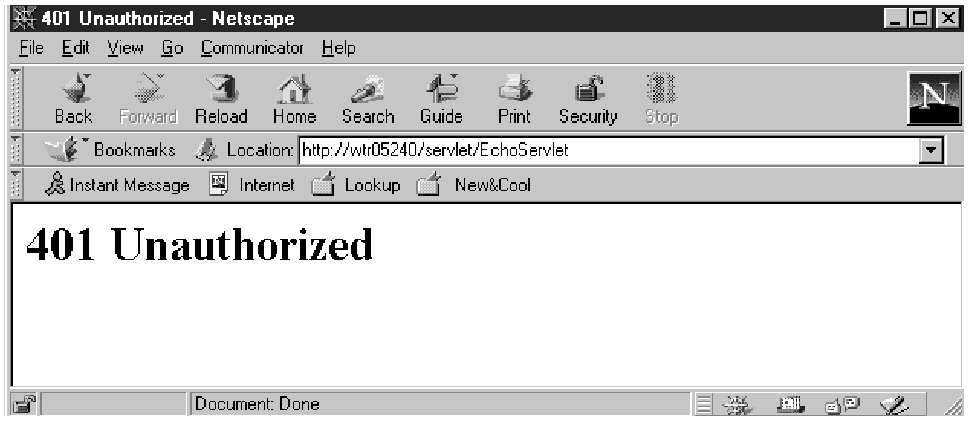
Authorization failed. Retry?

OK Cancel

Connect: Host wtr05240 contacted. Waiting for reply...

Figure 147. Authorization Failed

So we understand now that the user Popye, who has no permission to POST, cannot invoke a servlet protected by ServletExpress using the POST method. Then we tried clicking on the **Cancel** button, and it gave us this result:

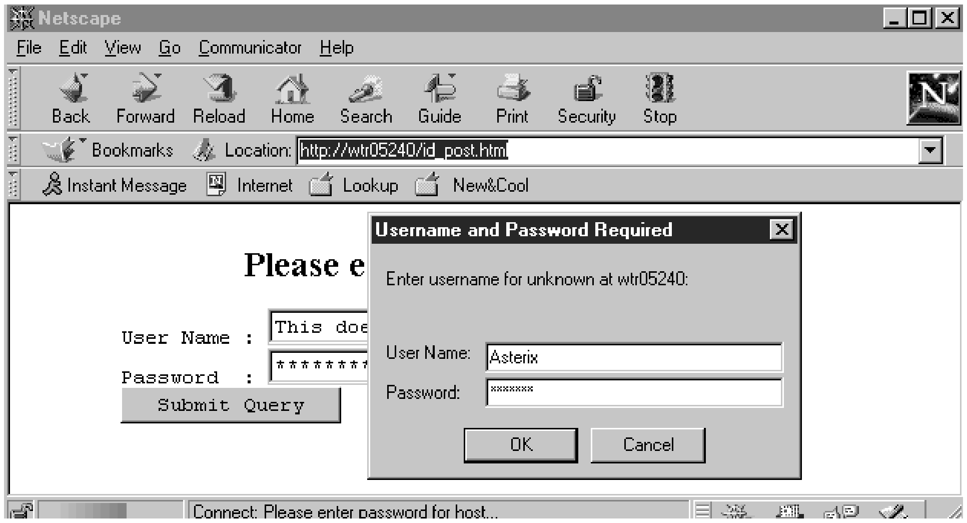

401 Unauthorized - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/servlet/EchoServlet
Instant Message Internet Lookup New&Cool

401 Unauthorized

Document: Done

Figure 148. No Access to the Servlet for an Unauthorized User ID

After that, we reaccessed the id_post.html page and tried posting the information. But this time we gave the user ID `Asterix` and the corresponding password.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/id_post.html
Instant Message Internet Lookup New&Cool

Please e

User Name : This doe

Password : *********

Submit Query

Username and Password Required

Enter username for unknown at wtr05240:

User Name: Asterix

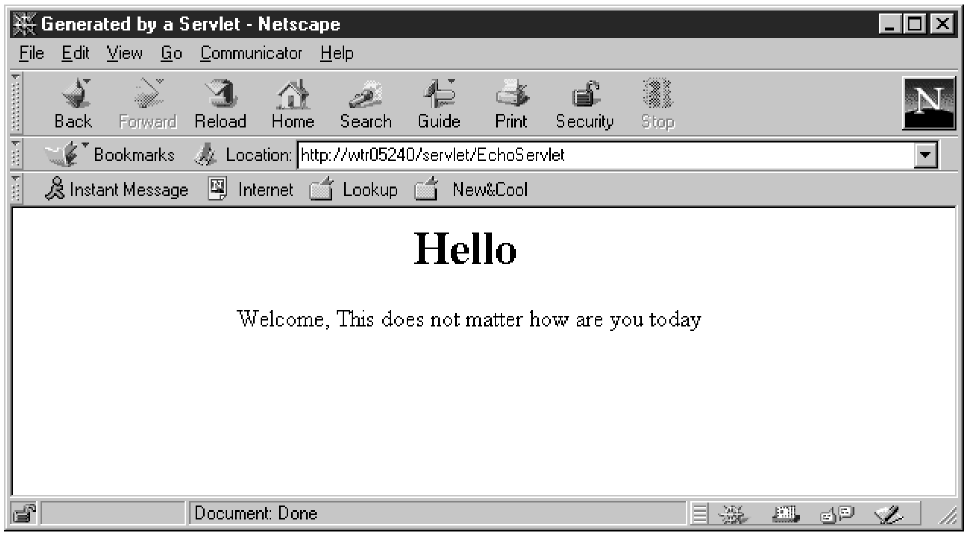
Password: *******

OK Cancel

Connect: Please enter password for host...

Figure 149. Accessing the Servlet with the Proper User ID

And since Asterix was authorized to POST, we got access to the output of the servlet as expected.


Generated by a Servlet - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/servlet/EchoServlet
Instant Message Internet Lookup New&Cool

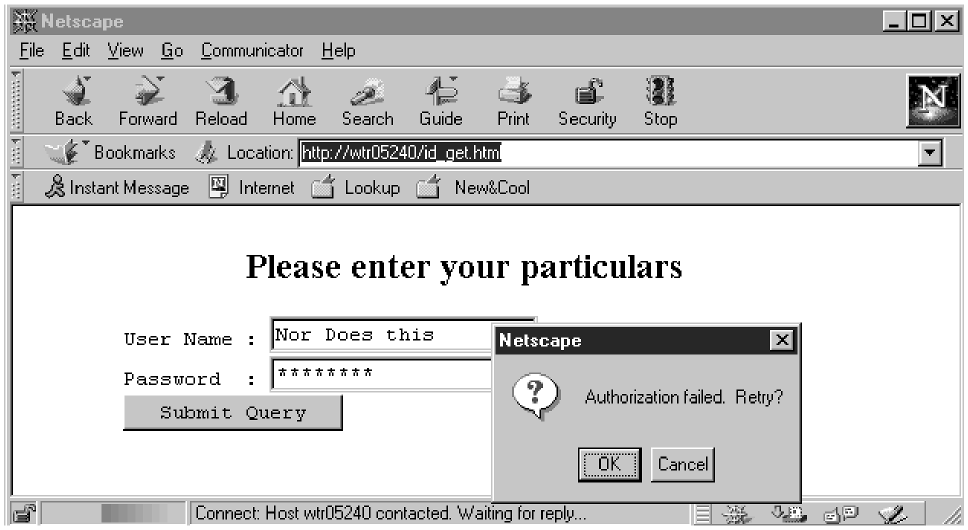
Hello

Welcome, This does not matter how are you today

Document: Done

Figure 150. Got Access to the Servlet

With that confirmed, we tried to use the GET method. We accessed the page id_get.html, and tried submitting the information. Here's what we got.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/id_get.html
Instant Message Internet Lookup New&Cool

Please enter your particulars

User Name : Nor Does this

Password : *********

Submit Query

Netscape

Authorization failed. Retry?

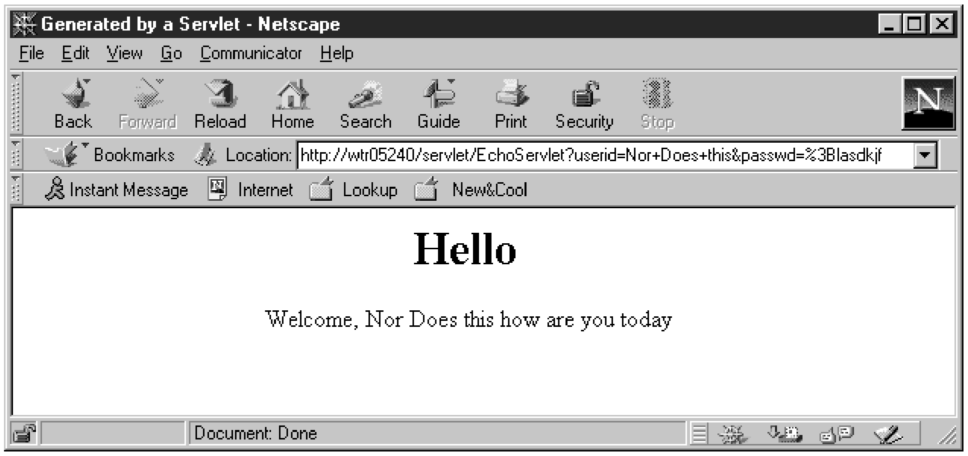
OK Cancel

Connect: Host wtr05240 contacted. Waiting for reply...

Figure 151. Accessing Using the GET Method

Why was that? The browser maintains the user ID and password mapped with a specific URL. This makes it convenient for the user, by not requiring him to key in the user ID and password with each request. The browser, however, after getting the user ID information from the user for the first time, stores it and sends it to the server each time. Hence, the browser sends the same user ID information to the server over and over again, till the server declines to accept it for some reason, upon which it pops up the window in Figure 147 on page 153.

Note here that there is no way of telling the browser that you would like to change the user ID, unless you would consider restarting the browser, upon which the mapping the browser maintains between the user ID - password pair and the URL is reset. So, getting back to using our EchoServlet, we clicked on **OK**, entered `Popye` and his password and got regular access to the output of the servlet.


Generated by a Servlet - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/servlet/EchoServlet?userid=Nor+Does+this&passwd=%3Blasdkjf
Instant Message Internet Lookup New&Cool

Hello

Welcome, Nor Does this how are you today

Document: Done

Figure 152. Access Using the GET Method, with Proper User ID

Everything worked fine with the user Popye, since we had granted to Popye permission to GET and the HTML page id_get.html uses the GET method to access the servlet.

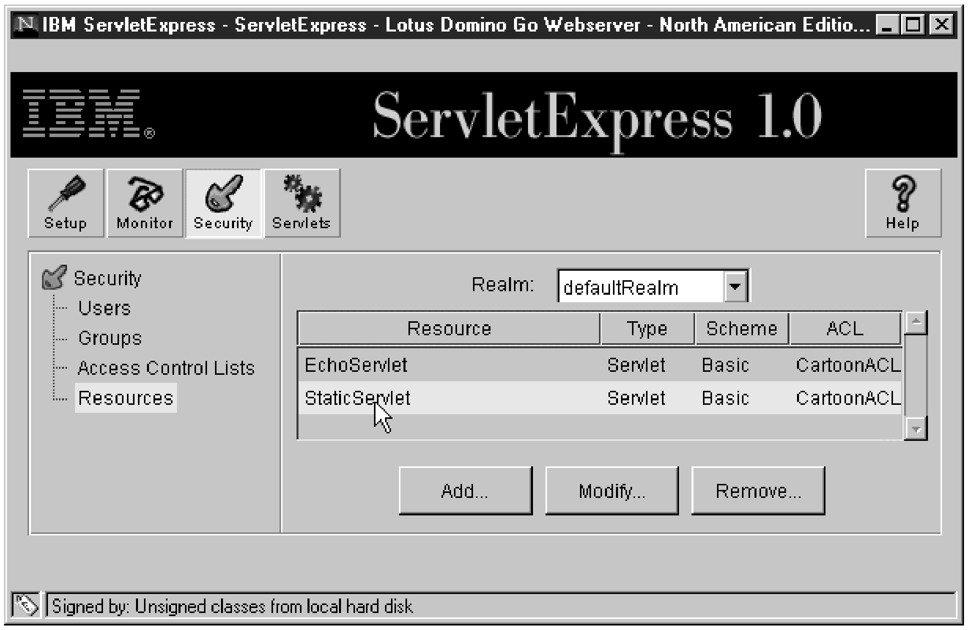
Once again, notice that the dynamic information with the GET method is always appended to the URL in the HTTP request header, and with the POST method it is sent through the HTTP request body (see Figure 150 on page 154).

Now let's consider an interesting case. Suppose we want to run the servlet without the HTML page - perhaps by giving the values at the URL (as in the GET method), or maybe we have a servlet that takes no input at all. One example for this could be a servlet that extracts information from a client certificate and performs certain action such as getting some information from a database or simply welcoming the person by name. We will try to keep our example simple. So let's write a rather simple servlet, that just says `Hello` without taking any inputs. We admit that we could just as well have written a static HTML page for this functionality, but we have mentioned the use of a no GET or POST input servlet. Here's the code of a servlet that really does nothing but give out static information. The name we gave to this servlet was StaticServlet.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class StaticServlet extends HttpServlet
{
  public void init(ServletConfig conf) throws ServletException
  {
    super.init(conf);
    log(" Static Server Initialized ");
  }

  public void service(HttpServletRequest req, HttpServletResponse res)
    throws IOException, ServletException
  {
    ServletOutputStream ops = res.getOutputStream();
    res.setContentType("text/html");
    ops.println("<HTML><HEAD><TITLE>Generated by a Servlet</TITLE></HEAD>");
    ops.println("<BODY>");
    ops.println("<CENTER><H1> Hello </H1>");
    ops.println(" Welcome, partner. How are you today");
    ops.println("<BR>");
    ops.println("</BODY> </HTML>");
    ops.close();
  }
}
```

Figure 153. StaticServlet.java.

What we now need to do is to restrict access to this servlet. So we add this as a resource in the CartoonsACL. For details, refer to 4.1.4, “Access Control Lists” on page 138.


IBM ServletExpress - ServletExpress - Lotus Domino Go Webserver - North American Editio...

ServletExpress 1.0

Realm: defaultRealm

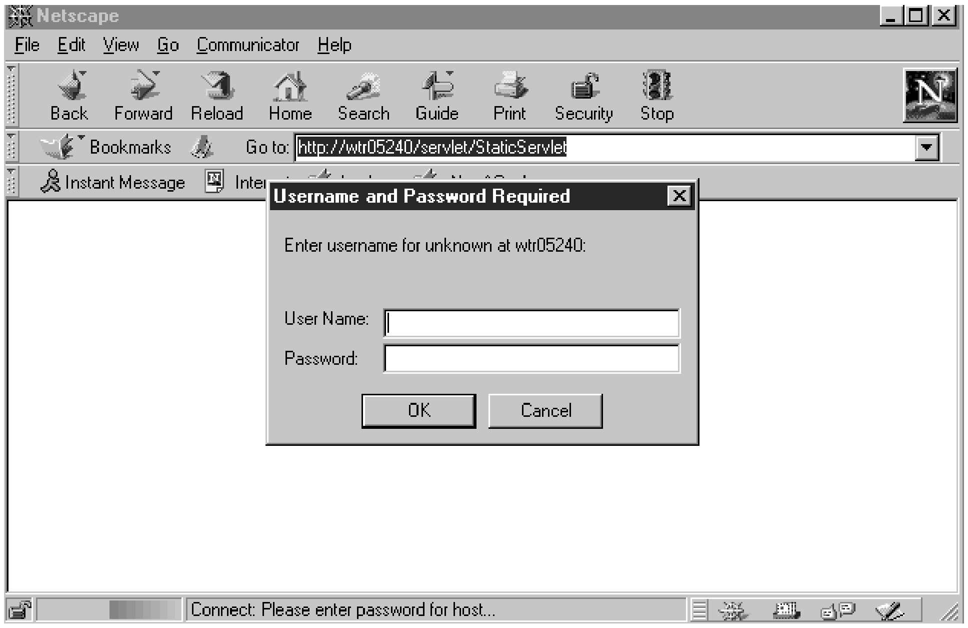
Resource	Type	Scheme	ACL
EchoServlet	Servlet	Basic	CartoonACL
StaticServlet	Servlet	Basic	CartoonACL

Signed by: Unsigned classes from local hard disk

Figure 154. Addition of StaticServlet Resource to the List.

Note that we have not changed the permissions of either users or groups within the ACL. We have just added another resource in the ACL. Asterix still has permission *only* to POST, and Popye still has permission *only* to GET. Now which of these do you think will be able to access the servlet? If you guessed Popye, with the GET permission, you were right. As we have mentioned before, the GET method uses the same stream for the form data as well as the URL. By giving someone the permission to do a GET, you are really giving the permission to connect using the URL stream. Obviously, if the URL stream is not read, the particular file cannot be delivered at all. So, a GET permission is required to read a file. Further, since GET uses the same stream for data and URL, our guess is that the user that has permission to do a GET method can run a servlet from the URL. We verified this, and here are the relevant screens.

After adding StaticServlet as a resource in the CartoonACL, using a browser, we went to the servlet directly pointing to its URL http://wtr05240/servlet/StaticServlet. On keying in the URL, and pressing Enter, we were prompted to give the user ID and password:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Go to: http://wtr05240/servlet/StaticServlet
Instant Message

Username and Password Required

Enter username for unknown at wtr05240:

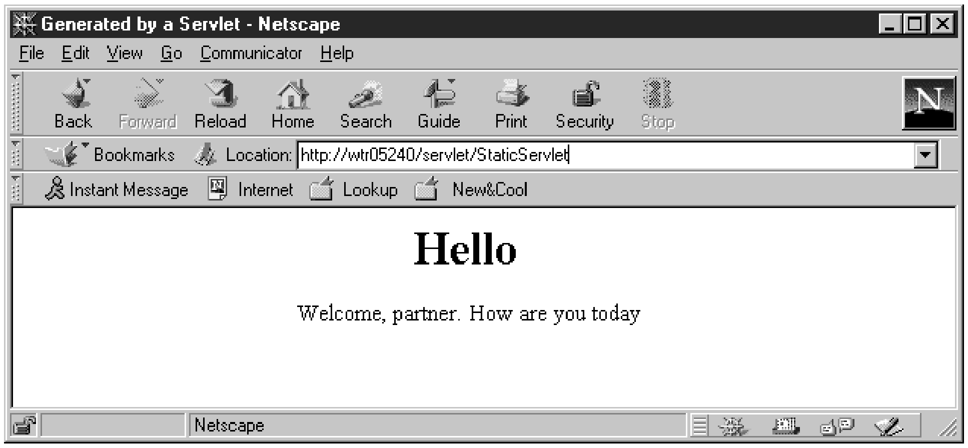
User Name:

Password:

Connect: Please enter password for host...

Figure 155. Prompt for User ID and Password to Run a Servlet

The system behaved pretty much as expected when we clicked on **Cancel**, by giving us the page as in Figure 148 on page 153. Also, when we tried entering the user ID as `Asterix`, since Asterix has the POST permission only, it gave us a message saying `Authorization failed` (as in Figure 151 on page 155). However, when we keyed in the user ID as `Popye` and the correct password, we got the output of the servlet.


Generated by a Servlet - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05240/servlet/StaticServlet
Instant Message Internet Lookup New&Cool

Hello

Welcome, partner. How are you today

Figure 156. The Output of the StaticServlet

Note that all that we found to be applicable to defaultRealm, we also found to be applicable to the NT realm, except for adding users and groups. We could add resources under the NT realm, and create ACLs, add permissions for the users created in the NT system, and get them to use these resources by logging on through the browser.

Restricting the Access to the file Servlet

Now that we have familiarized you with restricting access to a servlet, you can use the same or similar system to restrict access to the servlet named file (mentioned in 3.3.6, “Lotus Domino Go Webserver Access Control Lists” on page 114) by creating a separate ACL, adding no permission to any user in it and adding the servlet there. This *has* to be done on any system that relies extensively on the access control, and directory permission methods of Domino Go, if the system also has a ServletExpress running.

4.2 ServletExpress Advanced Security

When Lotus Domino Go Webserver acts in tandem with ServletExpress, you use the ServletExpress security measures to manage and restrict access to servlets. In fact, in order to plug ServletExpress in the Web server, the servlet engine provided by the Web server must be disabled. So you cannot use the security measures provided by the Web server to restrict access to servlets and the only possibility is to use the more advanced security controls that are offered by ServletExpress. We have just seen the basic security provided by ServletExpress. We go now more in depth.

4.2.1 Calculating Permissions

We had mentioned earlier about the problem generated by having a user in one or more groups, and these groups being given different permissions for an ACL by the administrator. Here's the solution. Let's assume that the user is a member of two groups, A and B. Hence, we have three permissions to take into consideration - that of group A, that of group B and that of the user. If we simply put a boolean notation for:

```
permitted = TRUE
```

and

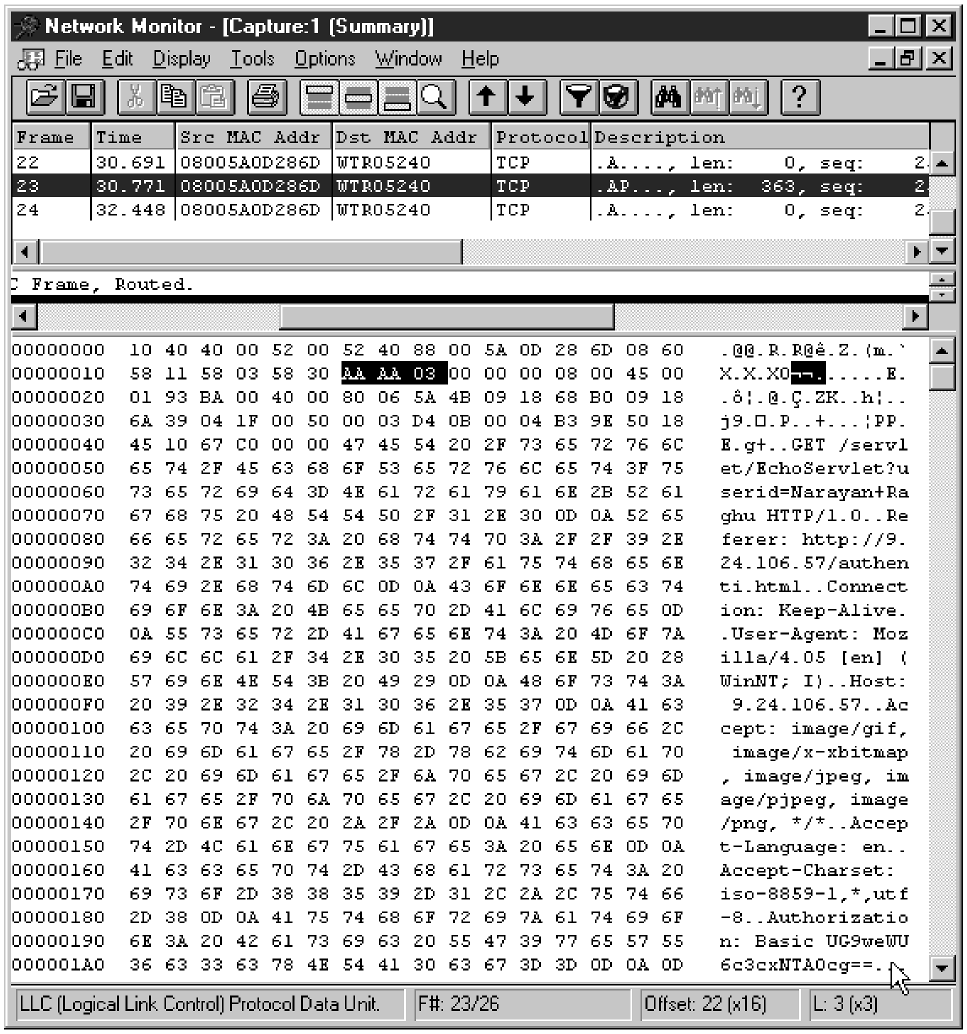
```
not permitted = FALSE
```

the resultant permission, coming out of the three permissions in question, is their logical `OR`. In other words, the user will be permitted to use the resource if either he or she has access to it according to the corresponding ACL, or if *any* of the groups to which he or she belongs has access to it in the corresponding ACL.

4.2.2 Flow of User ID and Password Information

We figured you'd be interested in how exactly the user ID and password information flows from the client to the server. Remember that you can set the type of authentication selecting **Basic Authentication** or **Digest Authentication** in the Protect a Resource screen, as shown in Figure 140 on page 148. We already commented on the fact that **SSL Authentication** appears disabled after the installation of ServletExpress.

The results were not too encouraging with Basic Authentication. The Network Monitor typically gave the following information:


Network Monitor - [Capture:1 (Summary)]
File Edit Display Tools Options Window Help

```
Frame  Time    Src MAC Addr  Dst MAC Addr  Protocol Description
22     30.691  08005A0D286D  WTR05240      TCP      .A...., len:    0, seq:   2
23     30.771  08005A0D286D  WTR05240      TCP      .AP..., len:  363, seq:   2
24     32.448  08005A0D286D  WTR05240      TCP      .A...., len:    0, seq:   2
```

```
C Frame, Routed.
```

```
00000000   10 40 40 00 52 00 52 40 88 00 5A 0D 28 6D 08 60   .@@.R.R@ê.Z.(m.`
00000010   58 11 58 03 58 30 AA AA 03 00 00 00 08 00 45 00   X.X.X0¬¬......E.
00000020   01 93 BA 00 40 00 80 06 5A 4B 09 18 68 B0 09 18   .ô¦.@.Ç.ZK..h¦..
00000030   6A 39 04 1F 00 50 00 03 D4 0B 00 04 B3 9E 50 18   j9.□.P..+...¦PP.
00000040   45 10 67 C0 00 00 47 45 54 20 2F 73 65 72 76 6C   E.g+..GET /servl
00000050   65 74 2F 45 63 68 6F 53 65 72 76 6C 65 74 3F 75   et/EchoServlet?u
00000060   73 65 72 69 64 3D 4E 61 72 61 79 61 6E 2B 52 61   serid=Narayan+Ra
00000070   67 68 75 20 48 54 54 50 2F 31 2E 30 0D 0A 52 65   ghu HTTP/1.0..Re
00000080   66 65 72 65 72 3A 20 68 74 74 70 3A 2F 2F 39 2E   ferer: http://9.
00000090   32 34 2E 31 30 36 2E 35 37 2F 61 75 74 68 65 6E   24.106.57/authen
000000A0   74 69 2E 68 74 6D 6C 0D 0A 43 6F 6E 6E 65 63 74   ti.html..Connect
000000B0   69 6F 6E 3A 20 4B 65 65 70 2D 41 6C 69 76 65 0D   ion: Keep-Alive.
000000C0   0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A   .User-Agent: Moz
000000D0   69 6C 6C 61 2F 34 2E 30 35 20 5B 65 6E 5D 20 28   illa/4.05 [en] (
000000E0   57 69 6E 4E 54 3B 20 49 29 0D 0A 48 6F 73 74 3A   WinNT; I)..Host:
000000F0   20 39 2E 32 34 2E 31 30 36 2E 35 37 0D 0A 41 63    9.24.106.57..Ac
00000100   63 65 70 74 3A 20 69 6D 61 67 65 2F 67 69 66 2C   cept: image/gif,
00000110   20 69 6D 61 67 65 2F 78 2D 78 62 69 74 6D 61 70    image/x-xbitmap
00000120   2C 20 69 6D 61 67 65 2F 6A 70 65 67 2C 20 69 6D   , image/jpeg, im
00000130   61 67 65 2F 70 6A 70 65 67 2C 20 69 6D 61 67 65   age/pjpeg, image
00000140   2F 70 6E 67 2C 20 2A 2F 2A 0D 0A 41 63 63 65 70   /png, */*..Accep
00000150   74 2D 4C 61 6E 67 75 61 67 65 3A 20 65 6E 0D 0A   t-Language: en..
00000160   41 63 63 65 70 74 2D 43 68 61 72 73 65 74 3A 20   Accept-Charset:
00000170   69 73 6F 2D 38 38 35 39 2D 31 2C 2A 2C 75 74 66   iso-8859-1,*,utf
00000180   2D 38 0D 0A 41 75 74 68 6F 72 69 7A 61 74 69 6F   -8..Authorizatio
00000190   6E 3A 20 42 61 73 69 63 20 55 47 39 77 65 57 55   n: Basic UG9weWU
000001A0   36 63 33 63 78 4E 54 41 30 63 67 3D 3D 0D 0A 0D   6c3cxNTA0cg==.
```

```
LLC (Logical Link Control) Protocol Data Unit.   F#: 23/26   Offset: 22 (x16)   L: 3 (x3)
```

Figure 157. User ID and Password Flowing Across the Net

Observe where the cursor is pointing - the text

```
G9weWU6c3cxNTA0cg==
```

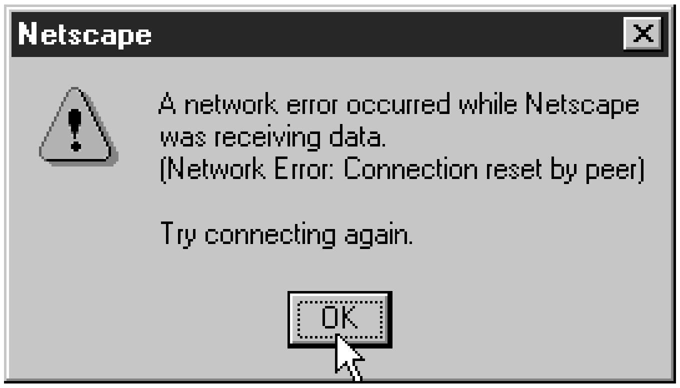
This is the base64 encoded form of

```
userID:password
```

as we verified as usual with our base64-to-binary converter.

We also tried getting rather mischievous and creating a user name with a colon in the middle and a password that had three colons. The system took it when we entered it, but then later it started showing up on the error log files, and then we had a tough time deleting it. So we proscribe the use of colons in both user IDs and passwords.

So what happens in Digest Authentication? We tried setting the same EchoServlet to Digest Authentication, and accessing the page. The client displayed the following error message:


Netscape

A network error occurred while Netscape was receiving data.
(Network Error: Connection reset by peer)

Try connecting again.

Figure 158. Netscape Error

From various documentation we learned that in Digest Authentication, the user ID and a digest containing an encrypted form of the password are sent to the server. The server computes a similar digest and grants access to the protected resources if the two digests are equal. Notice that if the Digest Authentication is enabled, what is sent over the net is not simply an encrypted form of the password, which could be decrypted if one had the correct key, but is a one-hash value of the password, which cannot be decrypted. So Digest Authentication provides a higher level of security than the base-64 encoded password.

Unfortunately, Digest Authentication is not supported by all browsers yet. We were working on Netscape Communicator 4.05, and it didn't seem to support it (at the time of this publication, only Sun's HotJava browser supports this protocol). When we tried to set back the EchoServlet to Basic Authentication, keying in the same user ID and password, it worked just fine. So, as of the release of this book, the only mode of authentication that we were able to use with ServletExpress and Netscape Communicator was the Basic Authentication. For security reasons, it should be used over SSL in an Internet scenario.

4.3 Servlet Sandbox

Servlets can be considered as server-side applets, even if they do not have a Graphical User Interface. Like applets, servlets run inside a sandbox, which is controlled by a SecurityManager. The servlets SecurityManager controls that operations such as network or file access are allowed.

Servlets run on the Web server JVM. Remotely loaded servlet, like applets, also are by default untrusted and must run inside a sandbox, so that such actions as network or file accesses are denied. Only internal servlets (which are servlets built into the Web server) or servlets properly installed in the servlet directory and managed by the Web server administrator are considered trusted and are granted all privileges.

Untrusted servlets, such as servlets that are loaded remotely, cannot accomplish tasks like network or file access. However, sometimes it becomes necessary to trust these remotely loaded servlets and to permit them to access system resources. This can be achieved by signing the JAR file containing the servlet class, and then loading it remotely.

Local and trusted remote servlets have full access to the server's private encryption keys, to the file system and to the network. They could even call the java.lang.System.exit() method, which terminates the currently running JVM.

In the next sections we will consider two ways in which a remote servlet can be run:

1. A servlet can be remotely invoked, meaning that it actually runs on the machine where the class file originally was.
2. A servlet can be remotely loaded, meaning that the class file is transferred from the Web server where it originally was and it runs in a destination Web server. Running remote code has security implications that are not present when you simply invoke a remote servlet. For this reason remotely loaded servlets have two different behaviors depending on whether they are signed or unsigned.

We will continue our discussion using the security features of the ServletExpress servlet engine.

ServletExpress overwrites the default implementation of the SecurityManager, so what you will find is that even unsigned servlets can become trusted and be granted all the permissions, provided that the same permissions are granted to the user named unsigned under the servletMgrRealm. This way, since you cannot distinguish where unsigned servlets originate from, all unsigned servlets will have the same permissions in the ServletExpress security model.

Also for signed servlets we will see that the situation is slightly different from the usual one. For example, in the ServletExpress security model it is not true that signed servlets are by default trusted, so that they have access to all the resources as if they were local servlets. In fact, once you have registered the servlet-signer under the servletMgrRealm, you can control exactly what actions the servlets signed by that user are permitted to do and what resources they are permitted to access.

Permissions to signed and unsigned servlets can be administered through the ServletExpress GUI.

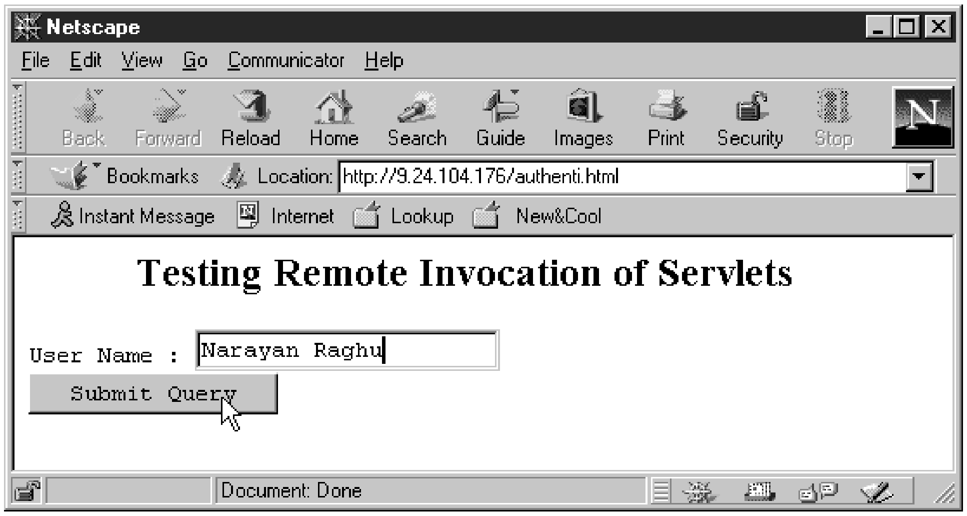
4.3.1 Remote Invocation of Servlets

Invoking a remote servlet is not really a security risk, since in remote invocation, the request is directly sent to a servlet on a new URL. We tried this with the following HTML page on a server A.

```
<HTML>

<BODY>
<CENTER> <H2> Testing Remote Invocation of Servlets </H2> </CENTER>

<FORM Method=GET Action="http://9.24.106.57/servlet/EchoServlet">
<pre>
User Name : <INPUT Type="text" Name="userid">
<INPUT Type="SUBMIT">
</PRE>
</FORM>

</BODY>

</HTML>
```

Figure 159. authenti.html on Server A

This HTML page, which we named autenti.html, when loaded from a client machine looks like this:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Images Print Security Stop
Bookmarks Location: http://9.24.104.176/authenti.html
Instant Message Internet Lookup New&Cool
Document: Done

Testing Remote Invocation of Servlets

User Name : Narayan Raghu

Submit Query

Figure 160. HTML Page on the Client Machine

The servlet is the same EchoServlet we have been using for a long time now - you could refer to it at Figure 91 on page 96. What it is more important to note is that this page invokes the EchoServlet on a server B. The HTTP GET method is used and the EchoServlet is registered as a protected resource on server B, requiring authentication to access. We have the user Popye with permission to GET and you will see that this user ID will have to be given to gain access.

The request, when the **Submit Query** button is pressed, flows directly from the browser to server B, where the servlet resides. This reflects what we have specified as the value for the `Action` attribute in the `<FORM>` tag (see Figure 159). What is really interesting here is that server A only serves the HTML page and it is quickly out of the scene. The authentication password and the final page flow directly between the client and the server B. We verified this using the Network Monitor.

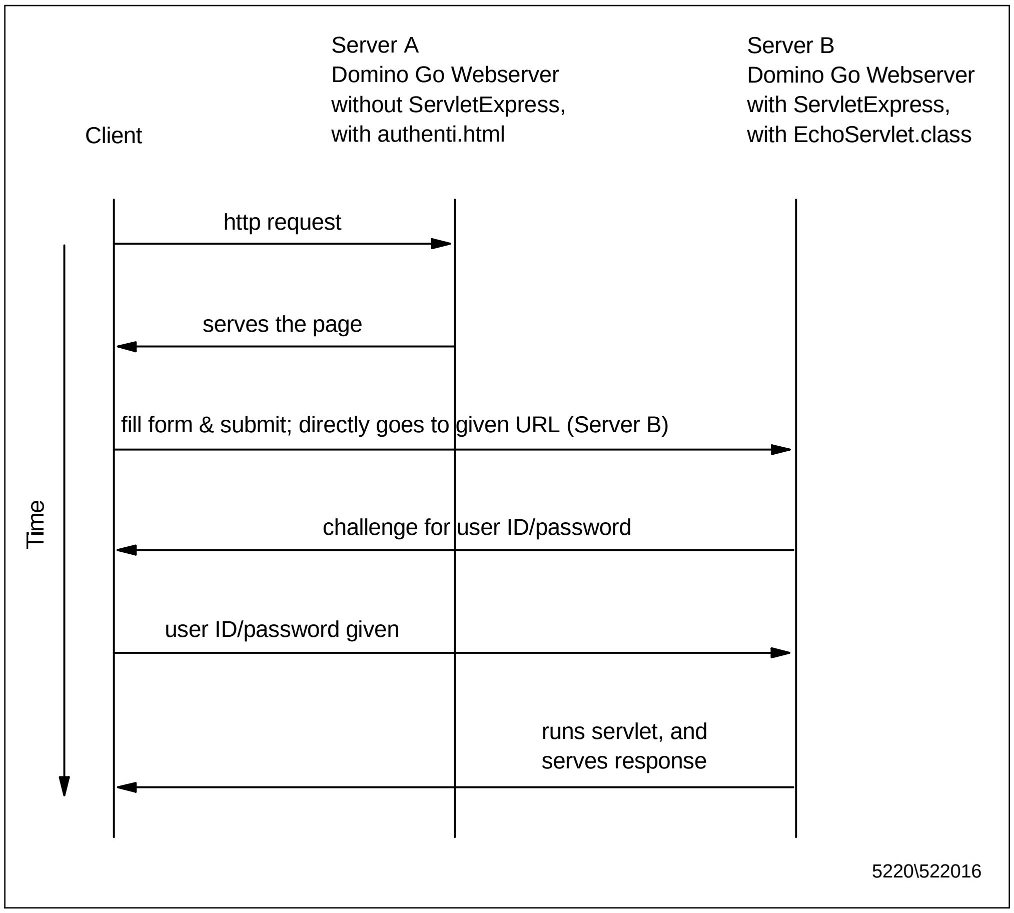


Figure 161. Remotely Invoking a Servlet

4.3.2 Remote Loading of Unsigned and Signed Servlets

Here, we consider the loading of a servlet in a server B from another server A. Unlike the previous case, this time the servlet will run in server A. The servlet should be stored in JAR format in server B and can be either signed or unsigned.

4.3.2.1 Unsigned Servlets

For an unsigned servlet from server A to run in server B, permissions ought to be set to reflect that the ServletExpress user named unsigned in the servletMgrRealm (see Figure 122 on page 131) has permissions to load servlets. A remote servlet is also treated as a resource. An unsigned servlet that is loaded from another server is run in a sandbox and, unless explicitly stated, has several restrictions on its functionality. We can now consider how an unsigned servlet can be loaded remotely and function as a normal servlet.

To experiment with this, we did the following. First, we wrote a simple servlet, called the DumbServlet. We created a JAR file from the servlet class file using the command:

```
jar cvf DumbServlet.jar DumbServlet.class
```

This we placed on server A, which simply had an HTTP daemon running. ServletExpress was not required on this Web server. On another server B, we had ServletExpress too.

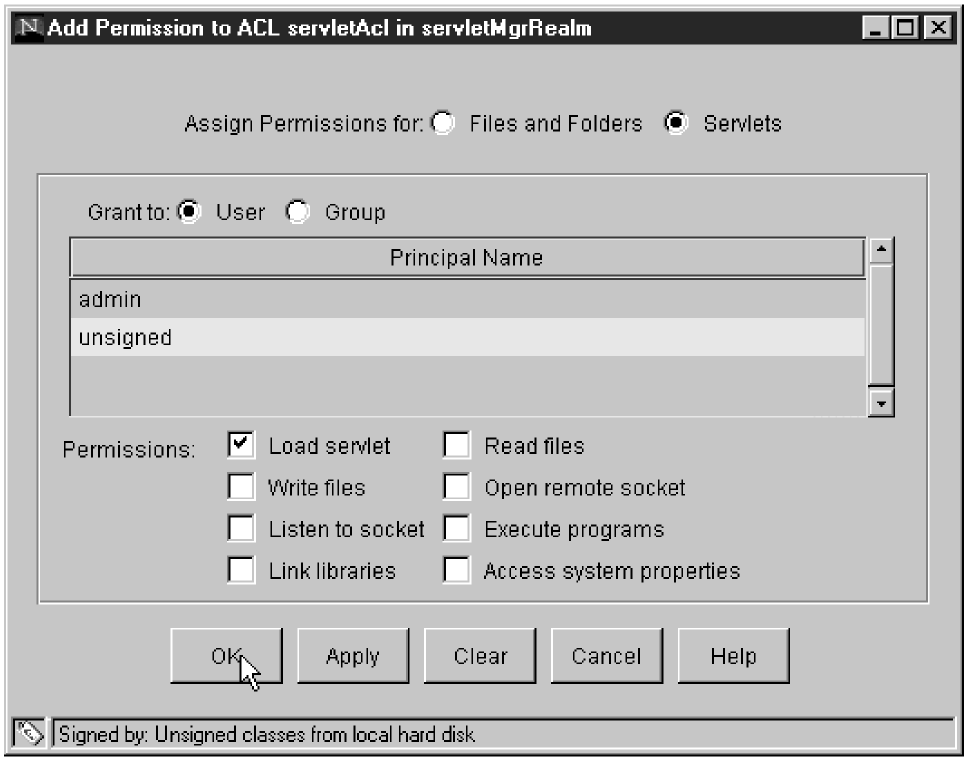
The server names and IP addresses are as follows:

Table 2. The Environment in Our Test with Unsigned Servlets

	Server A - Domino Go Webserver 4.6	**Server B - Domino Go Webserver 4.6.2.2**
Particulars	• Without ServletExpress • Contains the JAR file of the servlet	• With ServletExpress 1.0 • Loads the JAR file of the servlet
IP Address	9.24.104.176	9.24.106.57
Server Name	romeo	wtr05240

We started the ServletExpress Manager on server B logging in as admin and went to the **Security** - **Access Control Lists** screen (see Figure 127 on page 138). We chose the **servletMrgRealm** and clicked on **Add Permission**.

We then chose the permissions for servlets. To do this, we selected the radio box named **Servlets**, and checked **Load servlet**.



Assign Permissions for: Files and Folders [✓ Servlets]

Grant to: [✓ User] Group

admin
unsigned

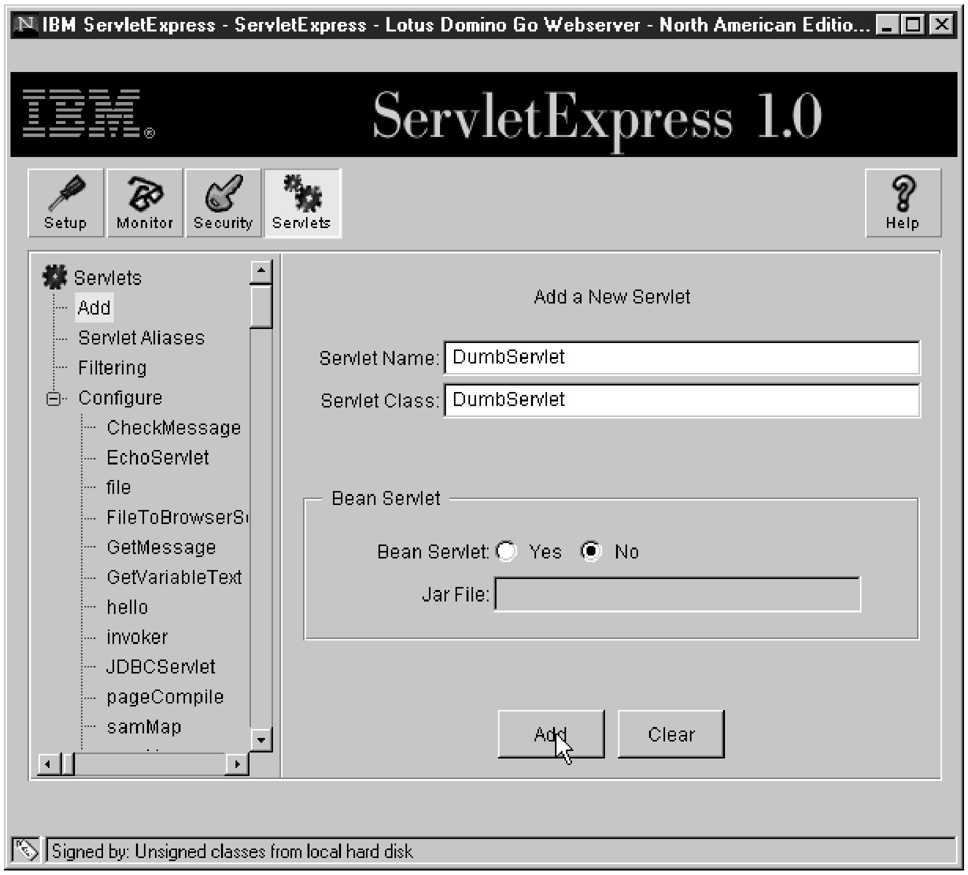
Permissions: [✓ Load servlet] Read files Write files Open remote socket Listen to socket Execute programs Link libraries Access system properties

Signed by: Unsigned classes from local hard disk

Figure 162. Permit Unsigned Servlets to load.

Notice that the selected user was **unsigned**.

Then we clicked on **Servlets** - **Add** and on the menu that appeared, entered the Servlet Name, Servlet Class and clicked **Add**.



Add a New Servlet

Servlet Name: DumbServlet

Servlet Class: DumbServlet

Bean Servlet

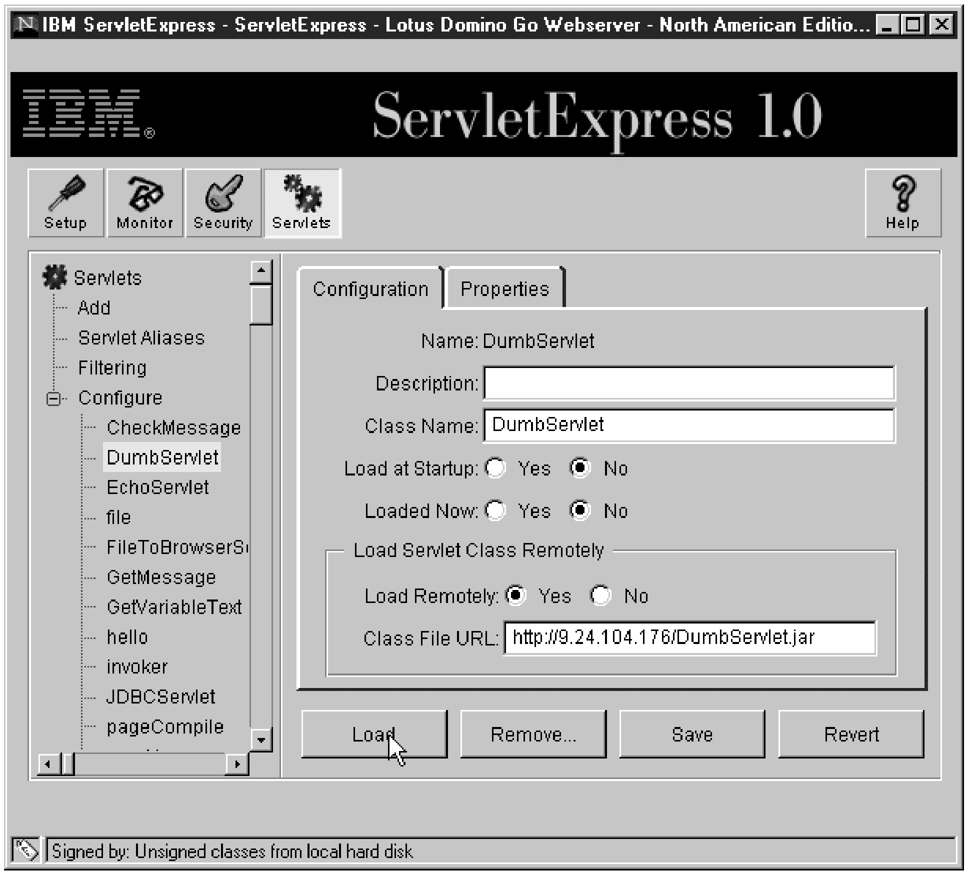
Bean Servlet: Yes [✓ No]

Jar File:

Signed by: Unsigned classes from local hard disk

Figure 163. Registering the New Servlet

On the next screen, we entered the particulars, chose the option **Load Remotely: Yes** and gave the URL where we had put the JAR file containing the DumbServlet. Then we clicked **Save** and **Load**.



Name: DumbServlet

Description:

Class Name: DumbServlet

Load at Startup: Yes [✓ No]

Loaded Now: Yes [✓ No]

Load Servlet Class Remotely

Load Remotely: [✓ Yes] No

Class File URL: http://9.24.104.176/DumbServlet.jar

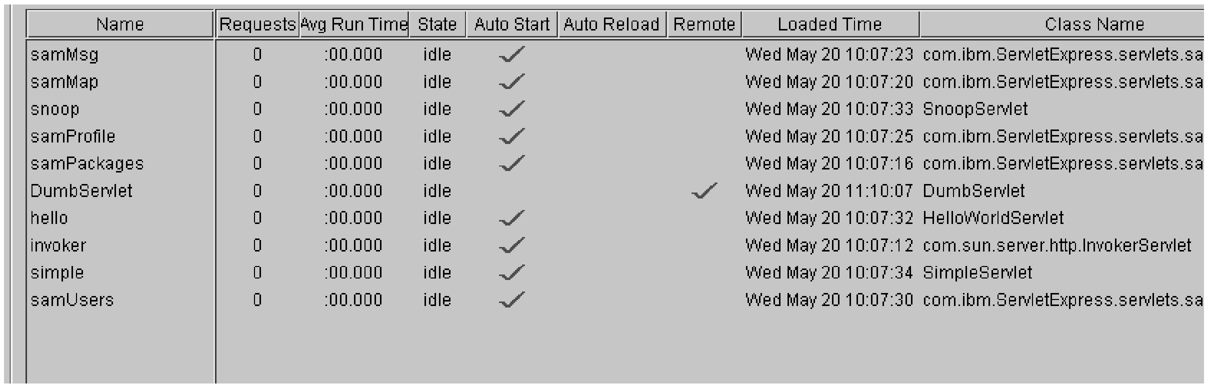
Signed by: Unsigned classes from local hard disk

Figure 164. Specifying the URL for Remote Loading

Save and Load

Sometimes the newly entered values will not take effect if you simply click on **Load**. Hence, it is advisable to click on **Save** before clicking on **Load** every time.

To verify that the servlet was indeed up and running in server B, we went to **Monitor** and clicked on **Loaded Servlets**. We got the particulars of all the loaded servlets.



Name	Requests	Avg Run Time	State	Auto Start	Auto Reload	Remote	Loaded Time	Class Name
samMsg	0	:00.000	idle	✓			Wed May 20 10:07:23	com.ibm.ServletExpress.servlets.sa
samMap	0	:00.000	idle	✓			Wed May 20 10:07:20	com.ibm.ServletExpress.servlets.sa
snoop	0	:00.000	idle	✓			Wed May 20 10:07:33	SnoopServlet
samProfile	0	:00.000	idle	✓			Wed May 20 10:07:25	com.ibm.ServletExpress.servlets.sa
samPackages	0	:00.000	idle	✓			Wed May 20 10:07:16	com.ibm.ServletExpress.servlets.sa
DumbServlet	0	:00.000	idle			✓	Wed May 20 11:10:07	DumbServlet
hello	0	:00.000	idle	✓			Wed May 20 10:07:32	HelloWorldServlet
invoker	0	:00.000	idle	✓			Wed May 20 10:07:12	com.sun.server.http.InvokerServlet
simple	0	:00.000	idle	✓			Wed May 20 10:07:34	SimpleServlet
samUsers	0	:00.000	idle	✓			Wed May 20 10:07:30	com.ibm.ServletExpress.servlets.sa

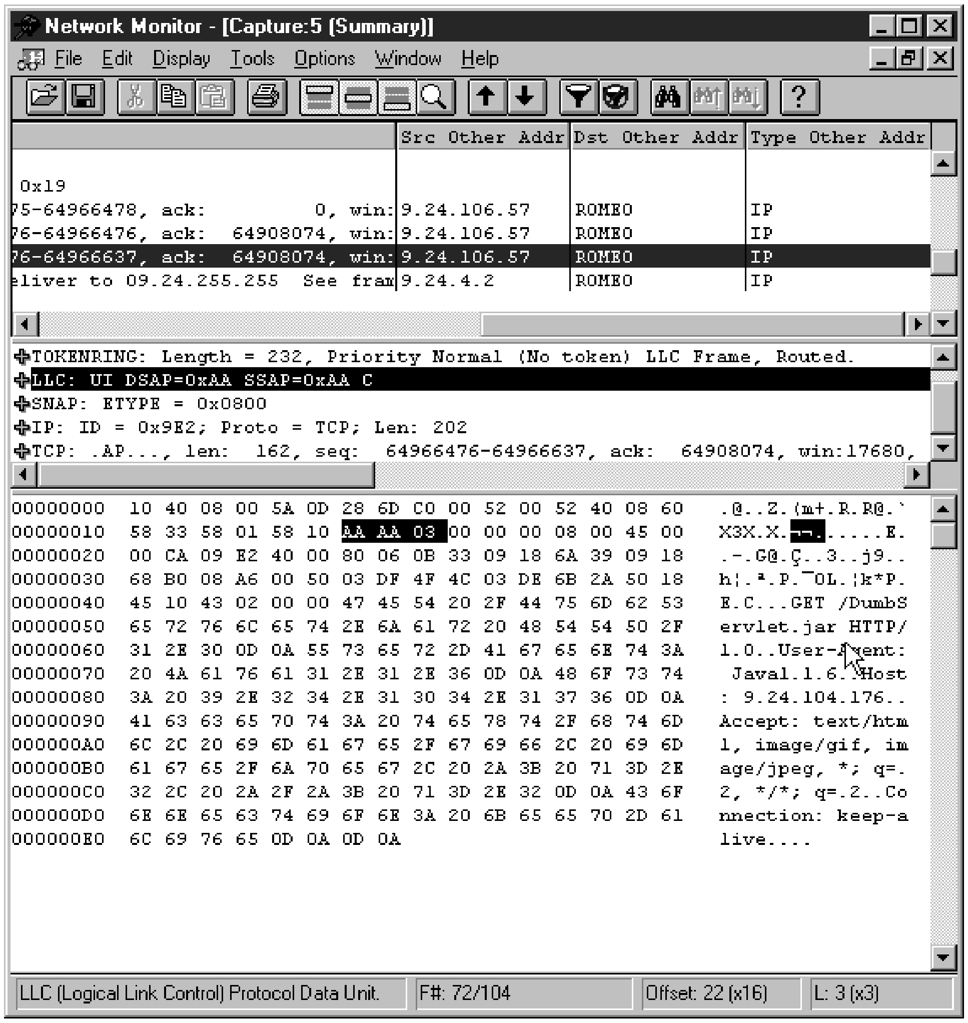
Figure 165. Verify That the Servlet Is Indeed Running in Server B

And as you can see, the DumbServlet was running and it appeared to be the only remotely loaded servlet between all the servlets registered in ServletExpress.

Later we verified that it really worked fine, by accessing the servlet with a browser on a client machine pointing to the server B. Also, we checked the network traffic using the Network Monitor.

The results were as expected. Once server B loads a servlet from server A, it no longer needs to interact with server A with regards to this servlet. Any client requests are handled by server B, and server A does not come into the picture. We discovered other noteworthy results, which we are going to tell you now.

We tried to capture the flow through the network when the servlet was loaded using the **Load** button shown in Figure 164 on page 167. The following two screens show respectively the incoming packets to the server A (romeo) and the incoming packets to the server B (wtr05240).


Network Monitor - [Capture:5 (Summary)]
File Edit Display Tools Options Window Help

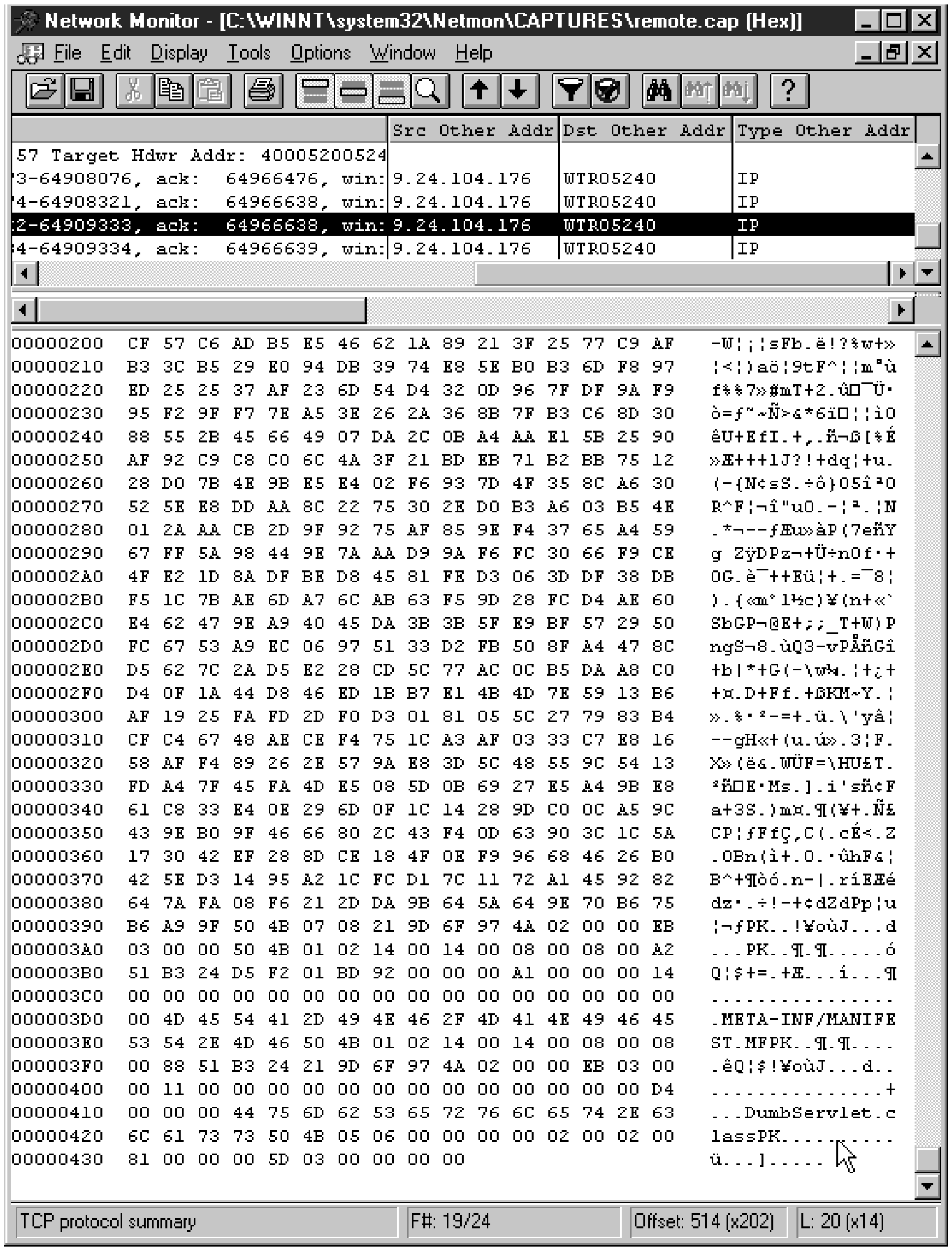
```
                                          Src Other Addr Dst Other Addr Type Other Addr

 0x19
75-64966478, ack:         0, win: 9.24.106.57    ROMEO          IP
76-64966476, ack:  64908074, win: 9.24.106.57    ROMEO          IP
76-64966637, ack:  64908074, win: 9.24.106.57    ROMEO          IP
eliver to 09.24.255.255  See fram 9.24.4.2       ROMEO          IP
```

```
TOKENRING: Length = 232, Priority Normal (No token) LLC Frame, Routed.
LLC: UI DSAP=0xAA SSAP=0xAA C
SNAP: ETYPE = 0x0800
IP: ID = 0x9E2; Proto = TCP; Len: 202
TCP: .AP..., len:  162, seq:  64966476-64966637, ack:  64908074, win:17680,
```

```
00000000  10 40 08 00 5A 0D 28 6D C0 00 52 00 52 40 08 60   .@..Z.(m+.R.R@.`
00000010  58 33 58 01 58 10 AA AA 03 00 00 00 08 00 45 00   X3X.X.¬¬......E.
00000020  00 CA 09 E2 40 00 80 06 0B 33 09 18 6A 39 09 18   .-.G@.Ç..3..j9..
00000030  68 B0 08 A6 00 50 03 DF 4F 4C 03 DE 6B 2A 50 18   h¦.ª.P.¯OL.¦k*P.
00000040  45 10 43 02 00 00 47 45 54 20 2F 44 75 6D 62 53   E.C...GET /DumbS
00000050  65 72 76 6C 65 74 2E 6A 61 72 20 48 54 54 50 2F   ervlet.jar HTTP/
00000060  31 2E 30 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A   1.0..User-Agent:
00000070  20 4A 61 76 61 31 2E 31 2E 36 0D 0A 48 6F 73 74    Java1.1.6..Host
00000080  3A 20 39 2E 32 34 2E 31 30 34 2E 31 37 36 0D 0A   : 9.24.104.176..
00000090  41 63 63 65 70 74 3A 20 74 65 78 74 2F 68 74 6D   Accept: text/htm
000000A0  6C 2C 20 69 6D 61 67 65 2F 67 69 66 2C 20 69 6D   l, image/gif, im
000000B0  61 67 65 2F 6A 70 65 67 2C 20 2A 3B 20 71 3D 2E   age/jpeg, *; q=.
000000C0  32 2C 20 2A 2F 2A 3B 20 71 3D 2E 32 0D 0A 43 6F   2, */*; q=.2..Co
000000D0  6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61   nnection: keep-a
000000E0  6C 69 76 65 0D 0A 0D 0A                           live....
```

```
LLC (Logical Link Control) Protocol Data Unit.   F#: 72/104   Offset: 22 (x16)   L: 3 (x3)
```

Figure 166. Incoming Packet Captured at Server A (romeo)



```
Network Monitor - [C:\WINNT\system32\Netmon\CAPTURES\remote.cap (Hex)]

                                         Src Other Addr  Dst Other Addr  Type Other Addr
57 Target Hdwr Addr: 40005200524
3-64908076, ack:  64966476, win: 9.24.104.176    WTR05240        IP
4-64908321, ack:  64966638, win: 9.24.104.176    WTR05240        IP
2-64909333, ack:  64966638, win: 9.24.104.176    WTR05240        IP
4-64909334, ack:  64966639, win: 9.24.104.176    WTR05240        IP

00000200  CF 57 C6 AD B5 E5 46 62 1A 89 21 3F 25 77 C9 AF   -W¦;¦sFb.ë!?%w+»
00000210  B3 3C B5 29 E0 94 DB 39 74 E8 5E B0 B3 6D F8 97   ¦<¦)aö¦9tF^¦¦m°ù
00000220  ED 25 25 37 AF 23 6D 54 D4 32 0D 96 7F DF 9A F9   f%%7»#mT+2.û□¯Ü·
00000230  95 F2 9F F7 7E A5 3E 26 2A 36 8B 7F B3 C6 8D 30   ò=ƒ˜~Ñ>&*6ï□¦¦ì0
00000240  88 55 2B 45 66 49 07 DA 2C 0B A4 AA E1 5B 25 90   êU+EfI.+,.ñ¬ß[%É
00000250  AF 92 C9 C8 C0 6C 4A 3F 21 BD EB 71 B2 BB 75 12   »Æ+++lJ?!+dq¦+u.
00000260  28 D0 7B 4E 9B E5 E4 02 F6 93 7D 4F 35 8C A6 30   (-{N¢sS.÷ô}O5îª0
00000270  52 5E E8 DD AA 8C 22 75 30 2E D0 B3 A6 03 B5 4E   R^F¦¬î"u0.-¦ª.¦N
00000280  01 2A AA CB 2D 9F 92 75 AF 85 9E F4 37 65 A4 59   .*¬--ƒÆu»àP(7eñY
00000290  67 FF 5A 98 44 9E 7A AA D9 9A F6 FC 30 66 F9 CE   g ZÿDPz¬+Ü÷n0f·+
000002A0  4F E2 1D 8A DF BE D8 45 81 FE D3 06 3D DF 38 DB   OG.è¯++Eü¦+.=¯8¦
000002B0  F5 1C 7B AE 6D A7 6C AB 63 F5 9D 28 FC D4 AE 60   ).{«mº l¼c)¥(n+«`
000002C0  E4 62 47 9E A9 40 45 DA 3B 3B 5F E9 BF 57 29 50   SbGP¬@E+;;_T+W)P
000002D0  FC 67 53 A9 EC 06 97 51 33 D2 FB 50 8F A4 47 8C   ngS¬8.ùQ3-vPÅñGî
000002E0  D5 62 7C 2A D5 E2 28 CD 5C 77 AC 0C B5 DA A8 C0   +b|*+G(-\w¼.¦+¿+
000002F0  D4 0F 1A 44 D8 46 ED 1B B7 E1 4B 4D 7E 59 13 B6   +¤.D+Ff.+ßKM~Y.¦
00000300  AF 19 25 FA FD 2D F0 D3 01 81 05 5C 27 79 83 B4   ».%·²-=+.ü.\'yâ¦
00000310  CF C4 67 48 AE CE F4 75 1C A3 AF 03 33 C7 E8 16   --gH«+(u.ú».3¦F.
00000320  58 AF F4 89 26 2E 57 9A E8 3D 5C 48 55 9C 54 13   X»(ë&.WÜF=\HU£T.
00000330  FD A4 7F 45 FA 4D E5 08 5D 0B 69 27 E5 A4 9B E8   ²ñ□E·Ms.].i'sñ¢F
00000340  61 C8 33 E4 0E 29 6D 0F 1C 14 28 9D C0 0C A5 9C   a+3S.)m¤.¶(¥+.Ñ£
00000350  43 9E B0 9F 46 66 80 2C 43 F4 0D 63 90 3C 1C 5A   CP¦ƒFfÇ,C(.cÉ<.Z
00000360  17 30 42 EF 28 8D CE 18 4F 0E F9 96 68 46 26 B0   .0Bn(ì+.O.·ûhF&¦
00000370  42 5E D3 14 95 A2 1C FC D1 7C 11 72 A1 45 92 82   B^+¶òó.n-|.riEÆé
00000380  64 7A FA 08 F6 21 2D DA 9B 64 5A 64 9E 70 B6 75   dz·.÷!-+¢dZdPp¦u
00000390  B6 A9 9F 50 4B 07 08 21 9D 6F 97 4A 02 00 00 EB   ¦¬ƒPK..!¥oùJ...d
000003A0  03 00 00 50 4B 01 02 14 00 14 00 08 00 08 00 A2   ...PK..¶.¶.....ó
000003B0  51 B3 24 D5 F2 01 BD 92 00 00 00 A1 00 00 00 14   Q¦$+=.+Æ...í...¶
000003C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000003D0  00 4D 45 54 41 2D 49 4E 46 2F 4D 41 4E 49 46 45   .META-INF/MANIFE
000003E0  53 54 2E 4D 46 50 4B 01 02 14 00 14 00 08 00 08   ST.MFPK..¶.¶....
000003F0  00 88 51 B3 24 21 9D 6F 97 4A 02 00 00 EB 03 00   .êQ¦$!¥oùJ...d..
00000400  00 11 00 00 00 00 00 00 00 00 00 00 00 00 00 D4   ...............+
00000410  00 00 00 44 75 6D 62 53 65 72 76 6C 65 74 2E 63   ...DumbServlet.c
00000420  6C 61 73 73 50 4B 05 06 00 00 00 00 02 00 02 00   lassPK..........
00000430  81 00 00 00 5D 03 00 00 00 00                     ü...].....

TCP protocol summary   F#: 19/24   Offset: 514 (x202)   L: 20 (x14)
```

Figure 167. Incoming Packet Captured at Server B (wtr05240)

You will notice how the servlet is requested from server A by server B in Figure 166 on page 168, and how it flows from server A to server B in Figure 167. Both these screens were captured immediately after the **Load** button was clicked. Also, note that the servlet, once loaded onto the server B, remains in its memory till the server is shut down. This is irrespective of whether it is subsequently unloaded.

For experimenting, we checked on **Servlets**, chose **DumbServlet** and explicitly unloaded it, and then reloaded it again (see Figure 164 on page 167). This activity was not reflected at the Network Monitor, indicating that the servlet is loaded on from a local source, and that it is loaded from the remote source *only* for the first time.

Of course, if you indicate in the options that the servlet is to be loaded automatically on startup, it will be done, and this will obviously be indicated at the Network Monitor every time the Web server B is started.

Unload and Reload

One serious shortcoming we found here is that if the servlet is loaded remotely, and *after* loading it we change the permissions such that it forbids the loading of unsigned remote servlets, it is not reflected in the system till the Web server is restarted. In other words, if we are to uncheck the **Load** permission in Figure 162 on page 165, then go to the **Servlets** and to the screen shown in Figure 164 on page 167, unload the servlet and then reload it, it would reload, although we had just revoked the permission to load a remote unsigned servlet. We also tried unloading it first, and then revoking the permission, and then tried loading it - it still worked. So we came to the conclusion that the permissions regarding the remote loading of servlets do not affect the already loaded servlets, and since the servlets are loaded either at startup of the server, or after it, the new permissions can affect the already loaded servlet only on restarting the Web server.

The feature that a remote servlet is loaded only once could have a negative side to it, in that if the servlet on the source server changes, the local server (server B) will never have the latest version unless it is restarted. This feature might be changed in the subsequent release of ServletExpress, by making the servlet to be downloaded from the remote server every time it is *unloaded* and then *loaded* again, even without bringing down the server.

4.3.2.2 Signed Servlets

As a final example of ServletExpress security, we will demonstrate to you how we got a remotely loaded signed servlet to access a file on the server's disk.

The environment configuration of this scenario also includes a server A and a server B, with the server A containing the signed JAR file of the servlet and the server B remotely loading and then running the servlet. The following table shows IP addresses, host names and particulars of our environment, and you can refer to this table when you see the pictures of this section.

Table 3. The Environment in Our Test with Signed Servlets

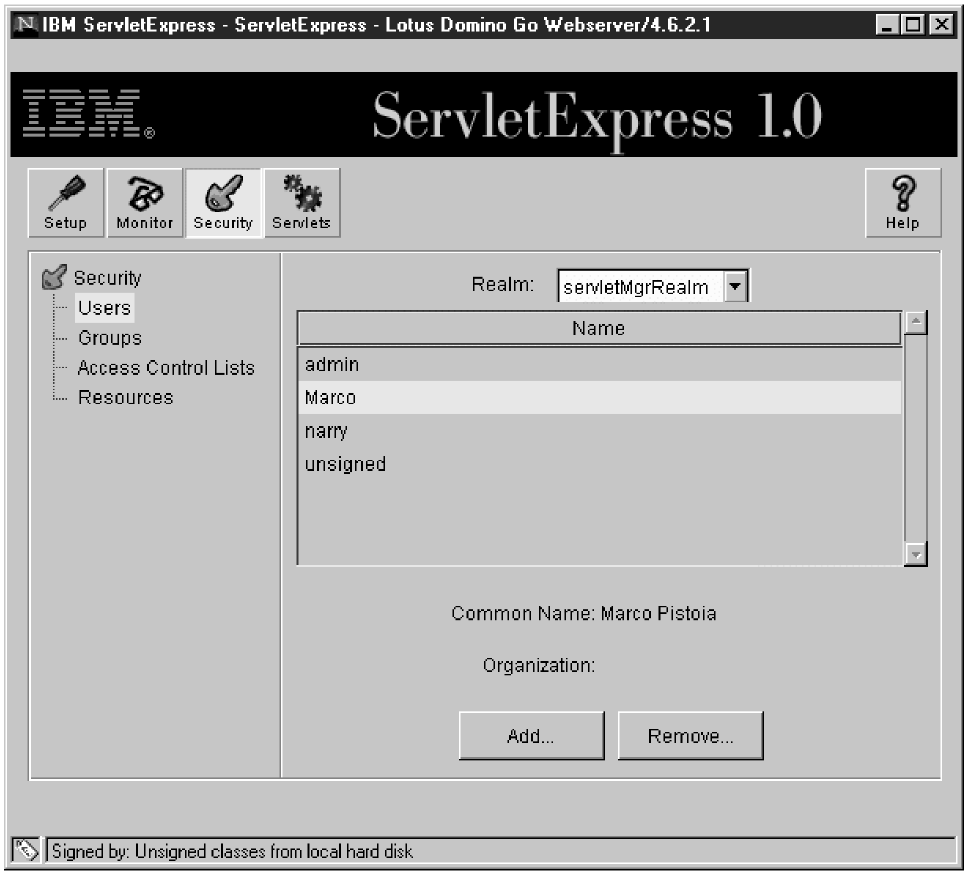
	Server A - Domino Go Webserver 4.6	**Server B - Domino Go Webserver 4.6.2.2**
Particulars	• Without ServletExpress • Contains the signed JAR file of the servlet	• With ServletExpress 1.0 • Loads the signed JAR file of the servlet
IP Address	9.24.104.176	9.24.104.51
Host Name	romeo	wtr05366

Notice, as discussed in 4.3.2.1, “Unsigned Servlets” on page 164, that the server A runs Lotus Domino Go Webserver without ServletExpress. In fact server A only has to serve the signed JAR file of the servlet, so the ServletExpress servlet engine was not considered necessary. In other words, server A only needs an HTTP daemon running.

Following the same steps described in 4.1.2, “Users” on page 128, we created a user under the servletMgrRealm in server B and we named this user Marco. Remember that Marco is the name used by ServletExpress to identify the servlet-signer, and the name registered in the certificate can be different. ServletExpress picks up the real name of the servlet-signer from the certificate and displays it in the window

```
Common Name: Marco Pistoia
```

as shown in the following screen:



Realm: servletMgrRealm

Name
admin
Marco
narry
unsigned

Common Name: Marco Pistoia

Organization:

Signed by: Unsigned classes from local hard disk

Figure 168. New Servlet-Signer Marco Added Under the servletMgrRealm

Then we wrote the Java code of the ReadFile servlet, shown in the following figure:

```
import javax.servlet.*;
import java.io.*;

public class ReadFile extends GenericServlet
{
  String fileParam;
  String lineIn;
  BufferedReader fileIn;

  public void service(ServletRequest req, ServletResponse res) throws IOException
  {
    res.setContentType("text/plain");
    ServletOutputStream os = res.getOutputStream();

    try
    {
      fileParam = req.getParameter("file");
      fileIn = new BufferedReader(new FileReader(fileParam));
      os.println("Writing from file " + fileParam);
      os.println();
      while ((lineIn = fileIn.readLine()) != null)
        if (lineIn.length() > 0)
          os.println(lineIn);
        else
          os.print(lineIn);
    }

        catch(Throwable e)
    {
        os.println(e.toString());
        PrintWriter pout = new PrintWriter(os);
        e.printStackTrace(pout);
        pout.flush();
        pout.close();
    }
  }     //end of service()
}
```

Figure 169. ReadFile.java

You will notice that the heart of this servlet is all in the service() method. The ReadFile servlet simply reads a file in the server where it runs (in our case, the server B) and displays the file name and the contents of the file on the client's browser, or it throws and catches an exception if something didn't work out fine.

We decided that the ReadFile servlet would read a text file named narry.txt, that we saved in the directory D:\NCF of our server B. In our experiment, it was not necessary to write a complicate file, so our simple file contained only the sentence

```
this should be shown
```

and nothing else.

Of course, the servlet needs to know what the file name is. The file name is actually known by the servlet through the value of the parameter named file. In the service() method, the servlet tries to get the value of the parameter file and to assign that value to the fileParam String variable:

```
fileParam = req.getParameter("file");
```

In order to let the servlet know the name of the file that you want it to read, you have several possibilities. For example, you can establish that the servlet read that parameter when it is loaded, and for this purpose ServletExpress provides you with a Properties panel for each servlet you add (we discuss this possibility in the comments regarding Figure 312 on page 331). Another chance that you have, is to pass that parameter through the URL, appending a question mark and then typing the string `file=D:\NCF\narry.txt`. Actually, even if a similar way to invoke the servlet is permitted, it didn't work out well with us, because it requires that particular characters of the URL, such as for example back slashes and colons, are encoded. So we thought that for a user on a client machine, a third possibility would be the best one: it would be easier to pass the file name as an input string on an HTML form, rather than encoding the file name and passing it appended to the URL.

This is the HTML file that we wrote and saved in the directory D:\WWW\HTML of our Web server B. We simply named it page.html:

```
<HTML>
<BODY>

<FORM Action=/servlet/ReadFile Method=GET>

<PRE>
FILE NAME <INPUT Type="text" Name="file">
<INPUT Type="SUBMIT">
</PRE>

</FORM>

</BODY>
</HTML>
```

Figure 170. page.html Invoking the ReadFile Servlet

Notice that it invokes the ReadFile servlet using the GET method, so the name of the file will be encoded anyway and then appended to the URL, but this process is done automatically without requiring any effort by the user on the client machine. Another particular that you should notice is that the servlet is invoked as if it were physically stored in the servlet directory D:\ServletExpress\servlets, but in reality the servlet will be remotely loaded from the server A and then it will only stay in the RAM memory of the server B.

Again in 4.1.2, “Users” on page 128 you can see the simple steps that are necessary to jar and sign a file. We compiled the ReadFile servlet entering the command:

```
javac ReadFile.java
```

in a directory D:\WWW\HTML\pistoia that we had created in the server A. Then we jared the class file with:

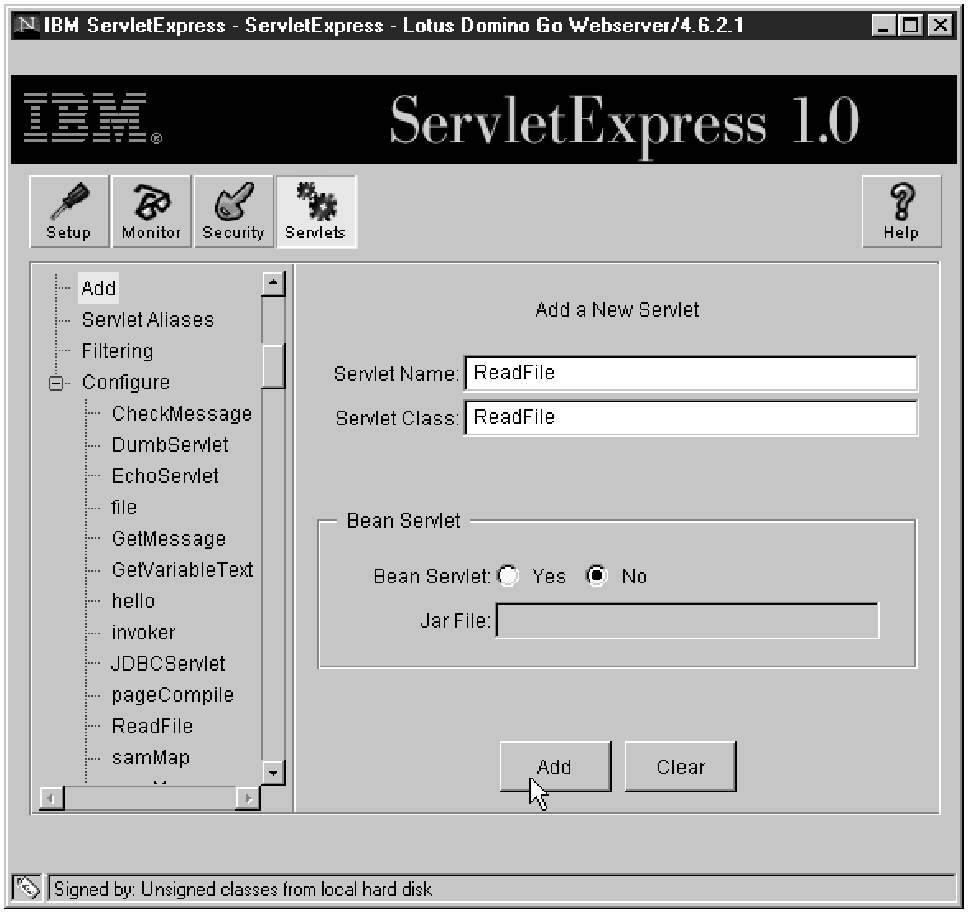
```
jar cvf ReadFile.jar ReadFile.class
```

and we signed the JAR file issuing:

```
javakey -gs sign.direc ReadFile.jar
```

sign.direc being the signature directive file, as discussed in 4.1.2, “Users” on page 128. The signed JAR file automatically carried the double extension .jar.sig, so that its name was ReadFile.jar.sig.

On the server B, which had to remotely load the servlet, it was necessary to add the servlet in the ServletExpress configuration panel. We clicked as usual on **Servlets - Add**. In the window that appeared we typed the Servlet Name and the Servlet Class and then we clicked **Add**, as shown here.



Add a New Servlet

Servlet Name: ReadFile

Servlet Class: ReadFile

Bean Servlet

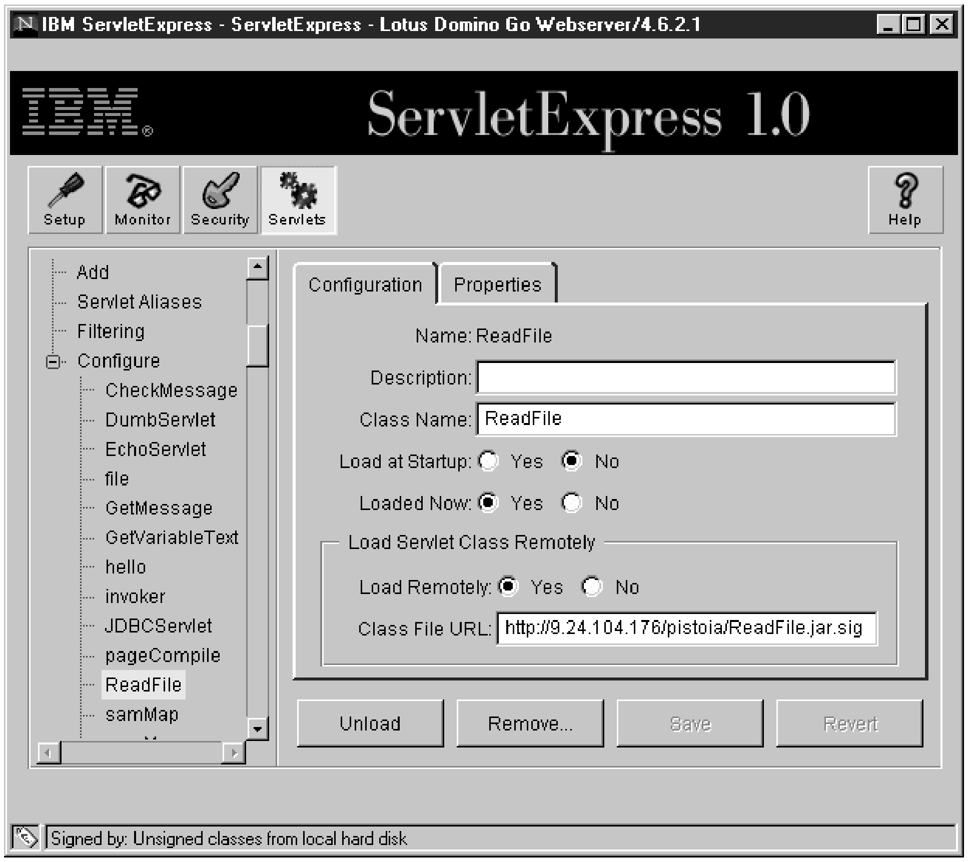
Bean Servlet: Yes [✓ No]

Jar File:

Signed by: Unsigned classes from local hard disk

Figure 171. Registering the ReadFile Servlet

The next screen that appeared allowed us to enter the particulars of the new servlet. It was in this panel that we configured the servlet to be remotely loaded, and to do this we had to check **Load Remotely: Yes** and specify the signed JAR file URL `http://9.24.104.176/pistoia/ReadFile.jar.sig`.



Name: ReadFile

Description:

Class Name: ReadFile

Load at Startup: Yes [✓ No]

Loaded Now: [✓ Yes] No

Load Servlet Class Remotely

Load Remotely: [✓ Yes] No

Class File URL: http://9.24.104.176/pistoia/ReadFile.jar.sig

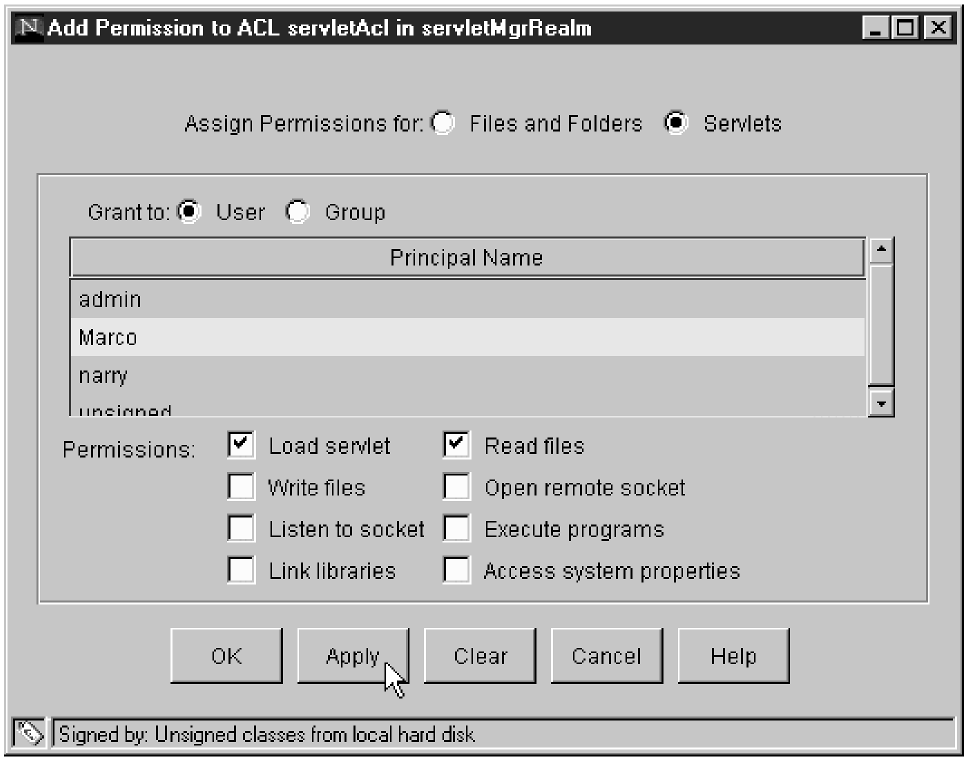
Signed by: Unsigned classes from local hard disk

Figure 172. Configuring the Signed Servlet To Be Loaded Remotely

We noticed that remote loading of a signed servlet also works fine if the servlet doesn't carry the double extension .jar.sig, but only the .jar extension, provided it has been signed and renamed. As you can see, it is not necessary that a signed JAR file be located in the servlet directory D:\WWW\servlets\public of the server A. In fact in this case, the Web server A simply serves the servlet to the server B, but it is the server B that runs it.

Before invoking the servlet, it was necessary to give to the servlet-signer Marco the adequate permissions. Remember, in fact, that in the ServletExpress security model, a servlet, even if signed, is not necessarily granted all the permissions as if it were a local servlet. It is up to the ServletExpress administrator to select what permissions are granted, through the ServletExpress GUI.

In this case, the servlet needs to be loaded and it also needs to read a file in the local hard disk. For this reason, the servlet-signer named Marco must be at least granted to load servlets and read files. To do this, it is necessary to click on **Access Control Lists - Add Permission...**. Then, in the window that appears, you should mark the corresponding **Load servlet** and **Read files** check boxes, as shown in the following screen:


Add Permission to ACL servletAcl in servletMgrRealm

Assign Permissions for: Files and Folders [✓ Servlets]

Grant to: [✓ User] Group

Principal Name
admin
Marco
narry
unsigned

Permissions:

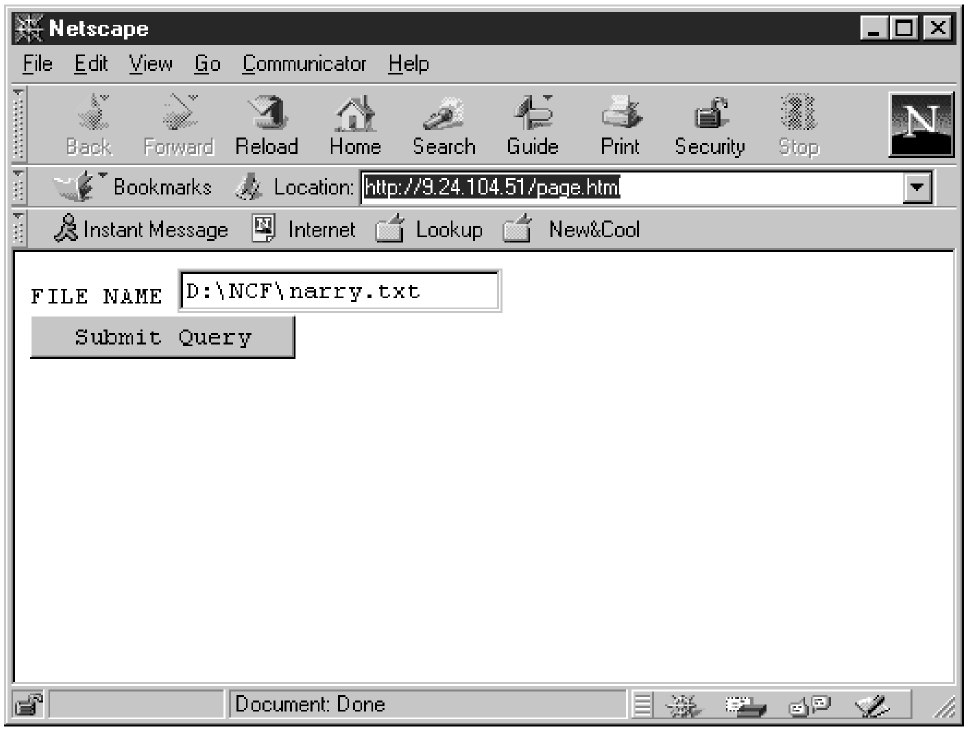
✓ Load servlet	✓ Read files
Write files	Open remote socket
Listen to socket	Execute programs
Link libraries	Access system properties

Signed by: Unsigned classes from local hard disk

Figure 173. Allowing Marco to Load Servlets and Read Files

Notice that, as we already observed in 4.3.2.1, “Unsigned Servlets” on page 164, if the ReadFile servlet has already been loaded, it remains in the memory of the Web server even if you try to download it, so that every time you change the permissions to the servlet-signer Marco or you upgrade the servlet in the original Web server A, you need to restart the Web server B if you want ServletExpress to pick up the changes.

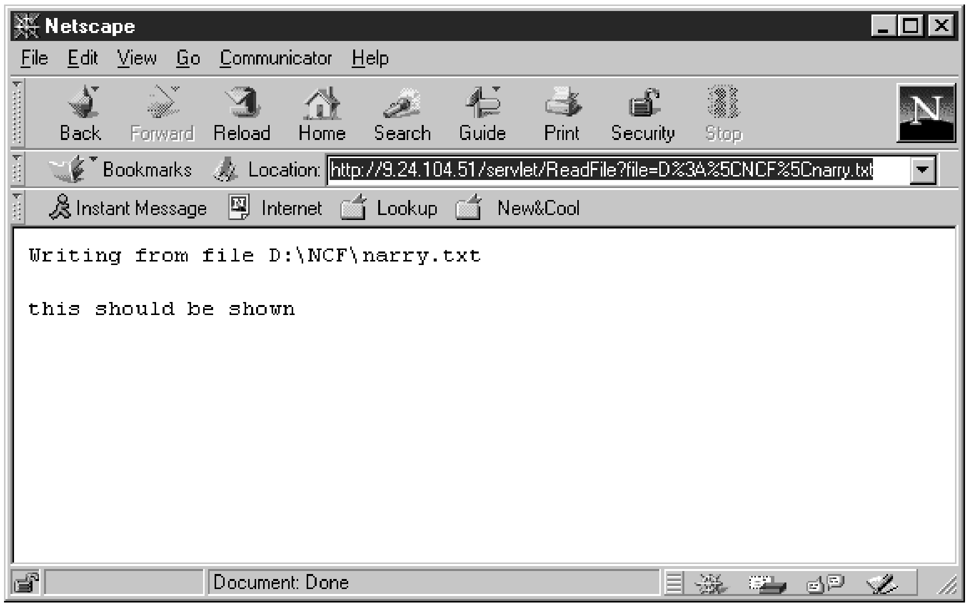
Then we were able to invoke the HTML page from the client machine, by pointing the browser to http://9.24.104.51/page.html. The HTML form was displayed and we filled it out with the name of the file.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://9.24.104.51/page.html
Instant Message Internet Lookup New&Cool
Document: Done

```
FILE NAME D:\NCF\narry.txt
Submit Query
```

Figure 174. Invoking and Filling the HTML Form

When we clicked the **Submit Query** button, the output was displayed without any problems:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://9.24.104.51/servlet/ReadFile?file=D%3A%5CNCF%5Cnarry.txt
Instant Message Internet Lookup New&Cool
Document: Done

```
Writing from file D:\NCF\narry.txt

this should be shown
```

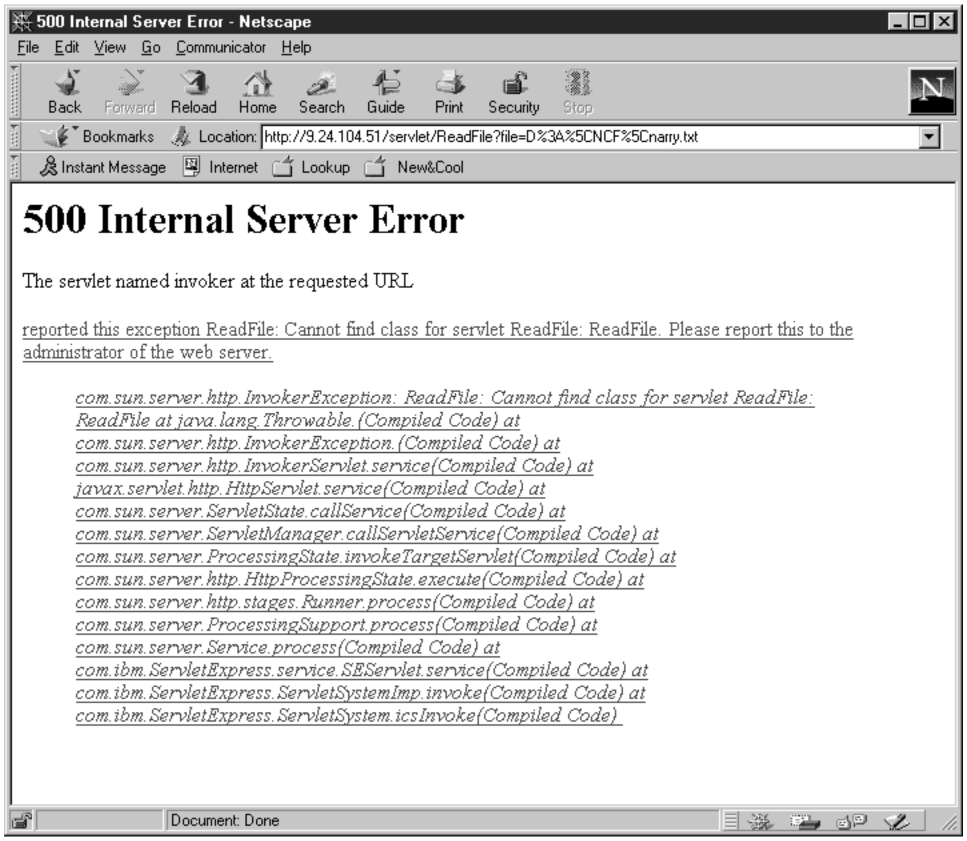
Figure 175. The HTML Page Invoked the Remotely Loaded Signed Servlet

Reading the string that appears in the Location field, you can see that what the GET method appends after the question mark is encoded, as we said.

What happens if the servlet-signer Marco is not granted permission to load servlets or to read files? In this case, all the servlets signed by Marco, even if signed, do not have the adequate permissions. In other words, we verified that it was really necessary to grant those permissions:

1. The permission to load servlets allows the servlet-signer to load a remote signed servlet.
2. The permission to read files allows the servlet signed by the servletMgrRealm user to read files on the local disk, as if it were a local servlet.

The Java source code of the ReadFile servlet had been written just to print on the client's browser all the exceptions coming from an incorrect configuration. Denying the servlet-signer Marco permission to load remote servlets, the browser displays the following error page:


500 Internal Server Error - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://9.24.104.51/servlet/ReadFile?file=D%3A%5CNCF%5Cnarry.txt
Instant Message Internet Lookup New&Cool
Document: Done

500 Internal Server Error

The servlet named invoker at the requested URL

reported this exception ReadFile: Cannot find class for servlet ReadFile: ReadFile. Please report this to the administrator of the web server.

```
com.sun.server.http.InvokerException: ReadFile: Cannot find class for servlet ReadFile:
ReadFile at java.lang.Throwable.(Compiled Code) at
com.sun.server.http.InvokerException.(Compiled Code) at
com.sun.server.http.InvokerServlet.service(Compiled Code) at
javax.servlet.http.HttpServlet.service(Compiled Code) at
com.sun.server.ServletState.callService(Compiled Code) at
com.sun.server.ServletManager.callServletService(Compiled Code) at
com.sun.server.ProcessingState.invokeTargetServlet(Compiled Code) at
com.sun.server.http.HttpProcessingState.execute(Compiled Code) at
com.sun.server.http.stages.Runner.process(Compiled Code) at
com.sun.server.ProcessingSupport.process(Compiled Code) at
com.sun.server.Service.process(Compiled Code) at
com.ibm.ServletExpress.service.SEServlet.service(Compiled Code) at
com.ibm.ServletExpress.ServletSystemImp.invoke(Compiled Code) at
com.ibm.ServletExpress.ServletSystem.icsInvoke(Compiled Code)
```

Figure 176. Error Page Displayed if the Permission to Load Servlet Is Denied

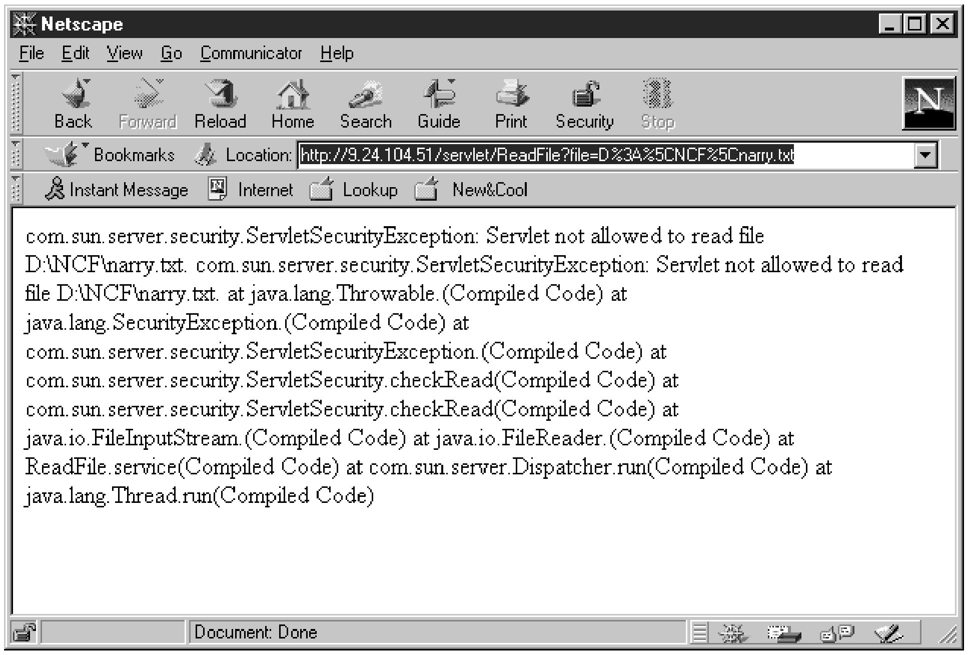
You can read the error message displayed by the servlet invoker for ServletExpress:

```
Cannot find class for servlet ReadFile
```

If the servlet-signer Marco has permission to load servlets but lacks permission to read files on the local disk, a ServletSecurityException is thrown:

```
com.sun.server.security.ServletSecurityException:
Servlet not allowed to read file D:\NCF\narry.txt.
```

The following figure shows the complete output:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://9.24.104.51/servlet/ReadFile?file=D%3A%5CNCF%5Cnarry.txt
Instant Message Internet Lookup New&Cool
Document: Done

```
com.sun.server.security.ServletSecurityException: Servlet not allowed to read file
D:\NCF\narry.txt. com.sun.server.security.ServletSecurityException: Servlet not allowed to read
file D:\NCF\narry.txt. at java.lang.Throwable.(Compiled Code) at
java.lang.SecurityException.(Compiled Code) at
com.sun.server.security.ServletSecurityException.(Compiled Code) at
com.sun.server.security.ServletSecurity.checkRead(Compiled Code) at
com.sun.server.security.ServletSecurity.checkRead(Compiled Code) at
java.io.FileInputStream.(Compiled Code) at java.io.FileReader.(Compiled Code) at
ReadFile.service(Compiled Code) at com.sun.server.Dispatcher.run(Compiled Code) at
java.lang.Thread.run(Compiled Code)
```

Figure 177. ServletSecurityException if the Permission To Read Files Is Not Granted

Chapter 5. Client-Side Security Technical References

This chapter describes NCF security from the client point of view. Other aspects of client security have already been covered in other parts of this redbook. For example we discuss Java 1.2 security in Chapter 2, "The New Java 1.2 Security Model" on page 9 and in particular, the security implications of signed and unsigned Java 1.2 applets are described in 2.7, "Applets in the New JDK 1.2 Security Model" on page 52. SSL client/server communication is extensively described in Chapter 3, "Web Server Security" on page 69 and in particular you can refer to 3.1.4, "SSL Client Authentication" on page 84 to know all the details for the client authentication.

We describe in this chapter those aspects of client security that we have not covered in other chapters.

We start with an overview of the security features offered by the most common Java-enabled Web browsers: Netscape Navigator, Microsoft Internet Explorer and Sun HotJava. In the rest of the chapter, we make extensive use of Netscape Communicator 4.05 and JDK 1.1.6. The reason why we did not use JDK 1.2 is that this release was still in beta at the time of this project and no browser supported JDK 1.2 yet. In this chapter we describe also security implications related to cookies and the Java applet programming model for NCF developers.

5.1 Specific Browsers and Security - General Description

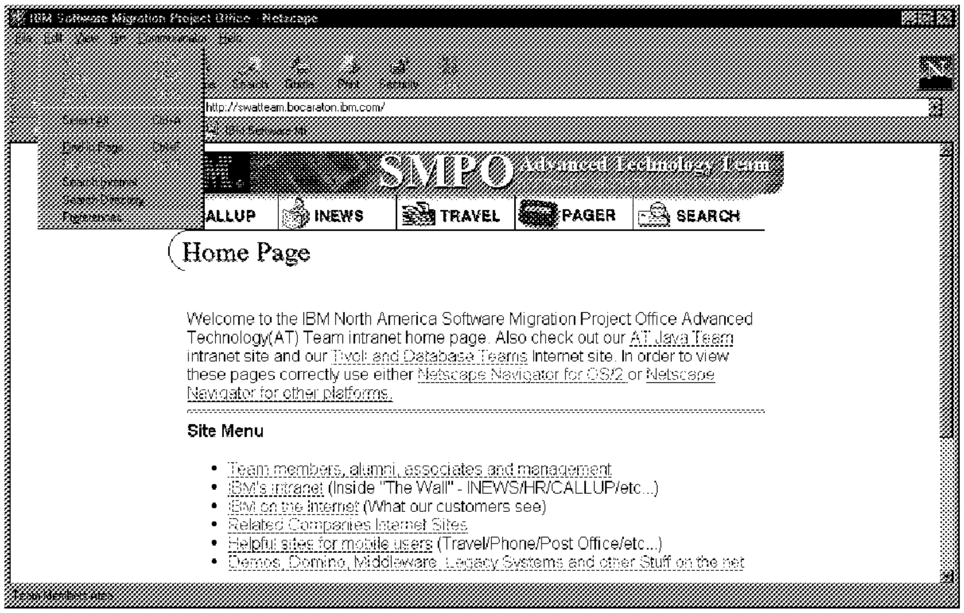
This first section provides an overview of the security features of the Netscape, Microsoft and Sun browsers. Enabling or disabling any of the security features in these browsers has trade-offs associated with the decision to do so. You should evaluate your own risk tolerance and your policies, and set up your security accordingly.

We also find it interesting in general that each browser concentrates security settings on the technology they created. Microsoft pays great attention to ActiveX controls, while Netscape and Sun pay great attention to Java.

5.1.1 Netscape Navigator

Netscape Navigator is the browser part of Netscape Communicator. We have used the Netscape browser in other sections of this chapter and in other chapters, and so some screen shots will be referred to, rather than repeated. What you will read in this section is based on our experience with Netscape Navigator 4.05.

Security-related configuration information can be found in two places in Navigator. Some items are found in the Preferences settings and some in the Security Information panels. The Preferences are accessed from the Edit pull-down on the menu bar and the Security Information panels can be accessed by clicking on the lock icon on the Navigation Toolbar.


IBM Software Migration Project Office - Netscape

http://swatteam.bocaraton.ibm.com/

SMPO Advanced Technology Team

ALLUP INEWS TRAVEL PAGER SEARCH

Home Page

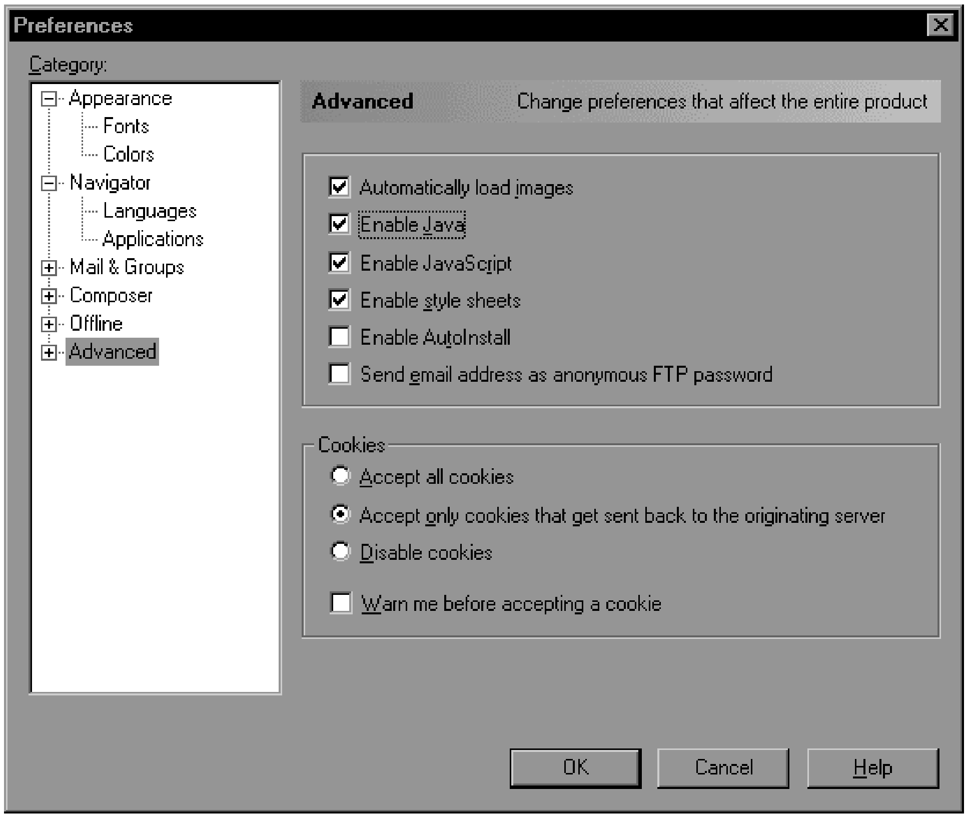
Welcome to the IBM North America Software Migration Project Office Advanced Technology(AT) Team intranet home page. Also check out our AT Java Team intranet site and our Tivoli and Database Teams Internet site. In order to view these pages correctly use either Netscape Navigator for OS/2 or Netscape Navigator for other platforms.

Site Menu

- Team members, alumni, associates and management
- IBM's intranet (Inside "The Wall" - INEWS/HR/CALLUP/etc...)
- IBM on the Internet (What our customers see)
- Related Companies Internet Sites
- Helpful sites for mobile users (Travel/Phone/Post Office/etc...)
- Demos, Domino, Middleware, Legacy Systems and other Stuff on the net

Figure 178. Netscape Navigator Browser

The Advanced section of the Preferences settings allows you to select whether to enable executable files to run in the browser and whether to allow servers to set cookies on your machine. Style sheets and Autoinstall are Netscape specific items. Autoinstall allows Smartupdate to run from the Netscape site to install patches (see Figure 220 on page 227). Style sheets allow uniform styles to be applied to Web pages.


Preferences

Category:

- Appearance
  - Fonts
  - Colors
- Navigator
  - Languages
  - Applications
- Mail & Groups
- Composer
- Offline
- Advanced

Advanced — Change preferences that affect the entire product

- [✓] Automatically load images
- [✓] Enable Java
- [✓] Enable JavaScript
- [✓] Enable style sheets
- [ ] Enable AutoInstall
- [ ] Send email address as anonymous FTP password

Cookies

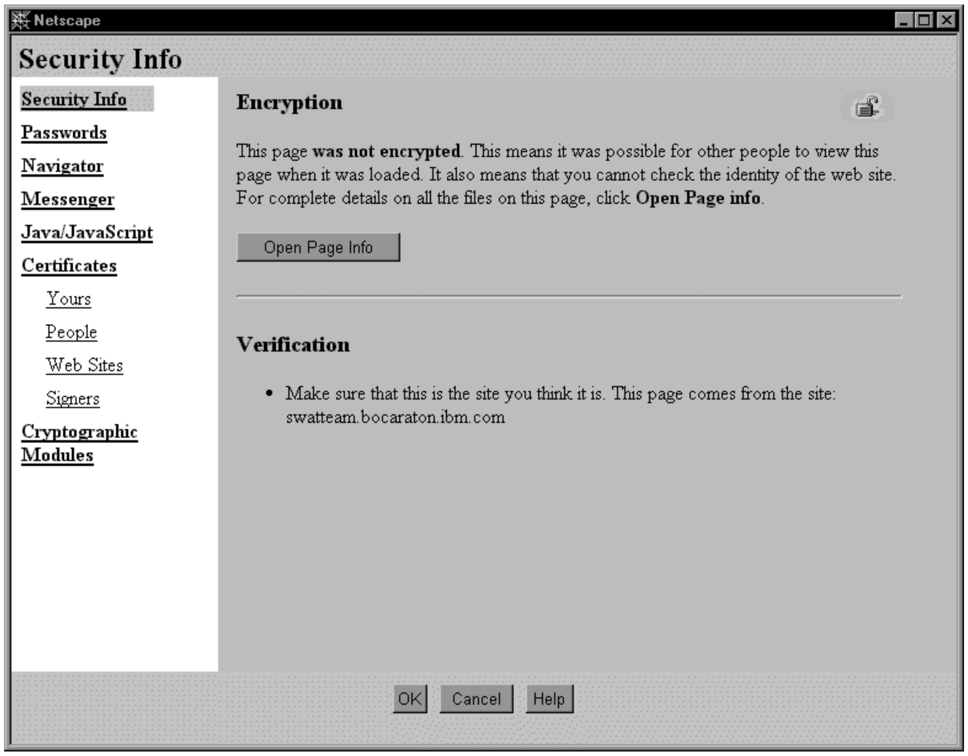
- ○ Accept all cookies
- ● Accept only cookies that get sent back to the originating server
- ○ Disable cookies
- [ ] Warn me before accepting a cookie

OK Cancel Help

Figure 179. Netscape Navigator Advanced Preferences

The section related to cookies allows you to set whether to allow information to be stored on your PC that Web sites can retrieve. A cookie is a text file that the browser maintains in the computer hard disk. Cookies are used to set preferences for Web sites and allow you to customize how a Web site knows you. Some Web sites track your usage of their site so that they can show you options or products in your interest area. A shopping Web site uses cookies for the shopping cart function. IBM WebSphere Application Server uses cookies for session control information. Microsoft uses them so you can have a personal Web page when you visit their site. Cookie use is controversial from the viewpoint of allowing information to be saved on your computer and then read by Web sites. Some consider turning off cookies for issues of privacy. Netscape gives you here the ability to disable cookies, allow them, or allow them only if they get sent back to the server that originated them. There is also a check box that allows you to see a warning if a server tries to set a cookie. See 5.5, "Cookie Security Implications" on page 250 for further details.

Other security-related information can be configured in the Security Information panels. Security Info is accessed by clicking on the lock icon on the Navigation Toolbar.



Encryption

This page **was not encrypted**. This means it was possible for other people to view this page when it was loaded. It also means that you cannot check the identity of the web site. For complete details on all the files on this page, click **Open Page info**.

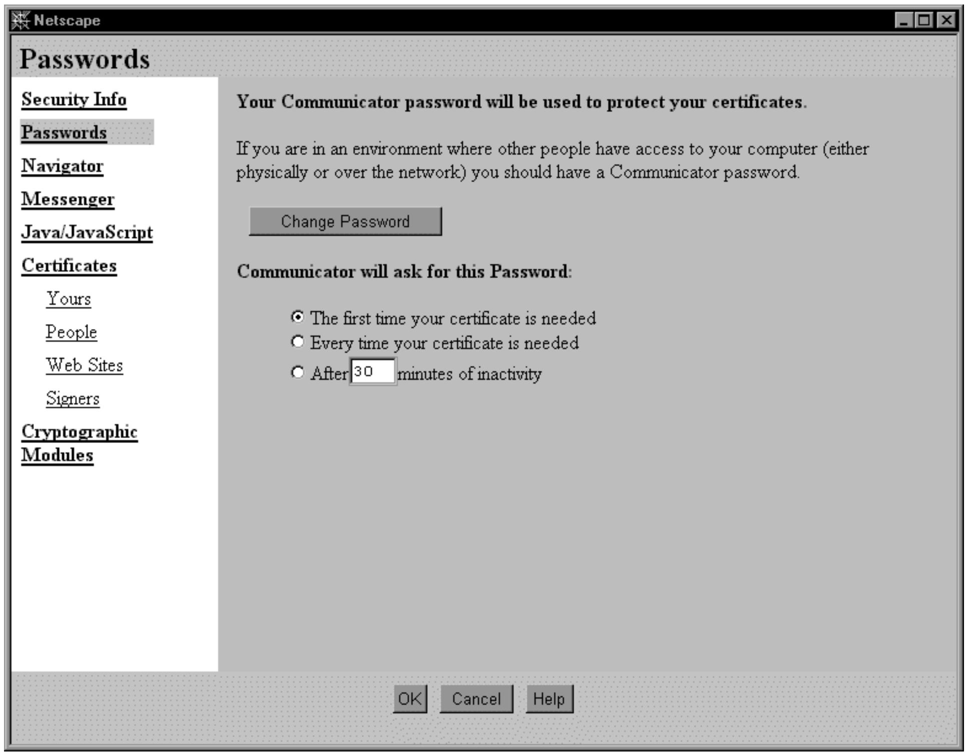
Verification

- Make sure that this is the site you think it is. This page comes from the site: swatteam.bocaraton.ibm.com

Figure 180. Netscape Security Information Panel

Notice that this window is shared by all the Communicator components. It provides access to the security features of Netscape and Netcaster as well as in e-mail and discussion groups. This panel will tell you security information about the page you are viewing, or the security operation that is taking place.

The following window shows the section of the Security page that you can access by clicking on **Passwords**:


Netscape

Passwords

Security Info
Passwords
Navigator
Messenger
Java/JavaScript
Certificates
Yours
People
Web Sites
Signers
Cryptographic Modules

Your Communicator password will be used to protect your certificates.

If you are in an environment where other people have access to your computer (either physically or over the network) you should have a Communicator password.

Change Password

Communicator will ask for this Password:

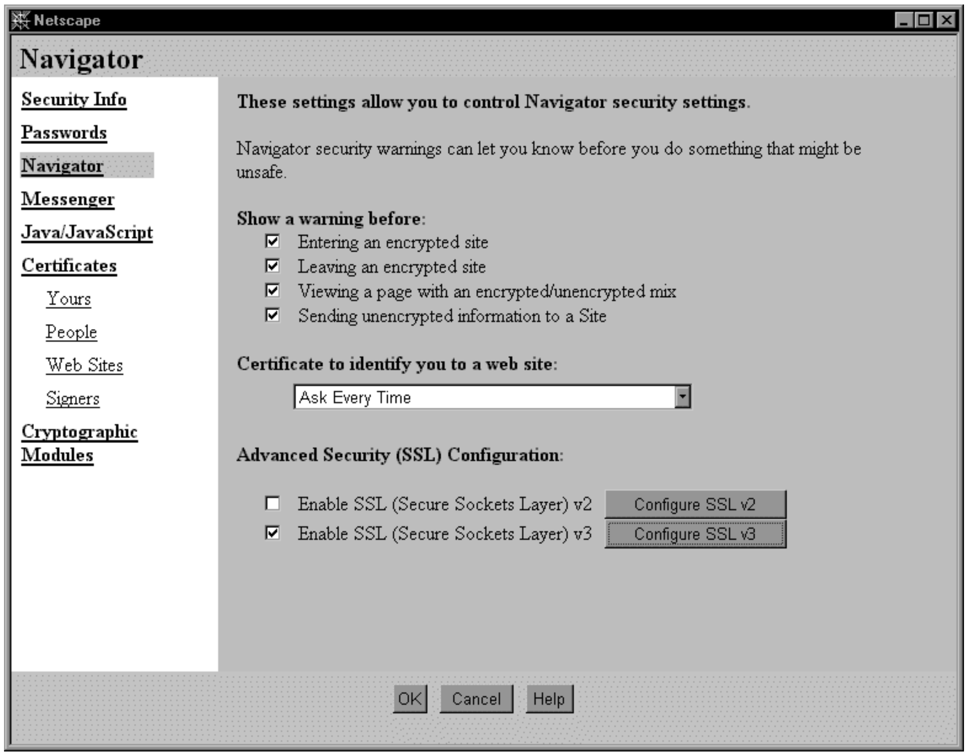
[✓ The first time your certificate is needed]
Every time your certificate is needed
After 30 minutes of inactivity

OK Cancel Help

Figure 181. Netscape Security Info - Passwords

The Password panel allows you to control the passwords on your Netscape Key Ring Organizer (KRO), the location on your computer where your digital certificates and also public/private key pairs are stored. This is a very important password because it protects access to digital certificates, used in cryptographic operations. If the PC you are using is used by other people, or is stolen, this password will protect the file that holds the certificates. It is important to consider the selection of how often you are challenged for the certificate access password. If you are vigilant about the use of lockup passwords when you leave your system, then you might choose to be challenged once a session. If it is possible for someone to sit at your computer while you are away, you may want to accept entering the password every time a certificate is used. As the legal status of digital signatures becomes more binding, protecting your signature should become more important.

The following window appears after selecting **Navigator**:


Netscape

Navigator

Security Info
Passwords
Navigator
Messenger
Java/JavaScript
Certificates
Yours
People
Web Sites
Signers
Cryptographic Modules

These settings allow you to control Navigator security settings.

Navigator security warnings can let you know before you do something that might be unsafe.

Show a warning before:

- ☑ Entering an encrypted site
- ☑ Leaving an encrypted site
- ☑ Viewing a page with an encrypted/unencrypted mix
- ☑ Sending unencrypted information to a Site

Certificate to identify you to a web site:

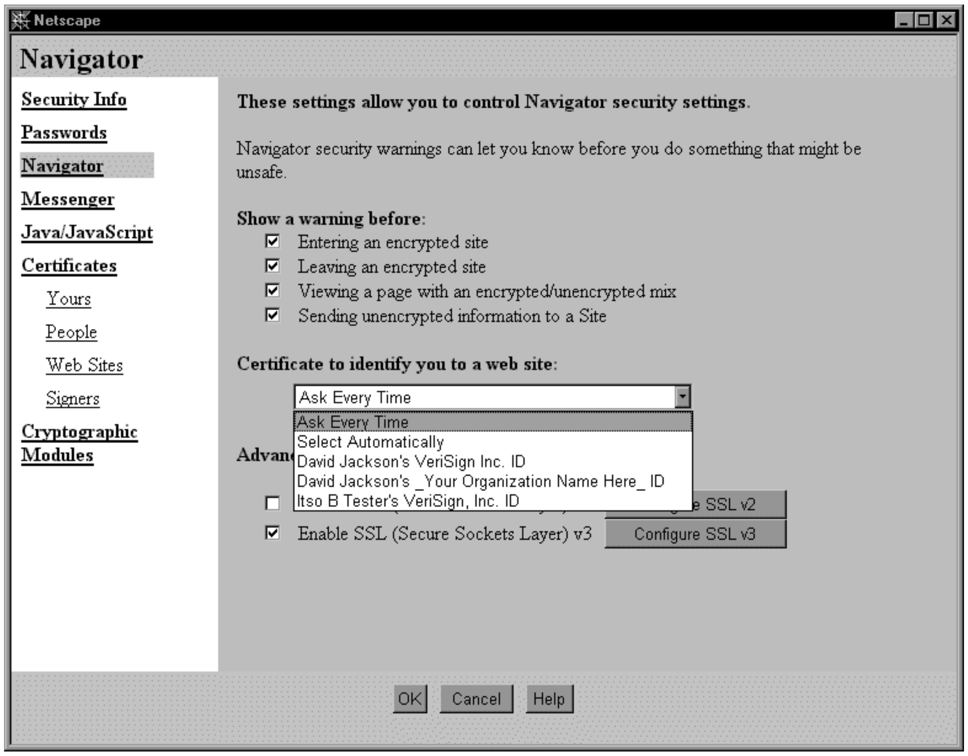
Ask Every Time

Advanced Security (SSL) Configuration:

- ☐ Enable SSL (Secure Sockets Layer) v2 — Configure SSL v2
- ☑ Enable SSL (Secure Sockets Layer) v3 — Configure SSL v3

Figure 182. Netscape Security Info - Navigator

The Navigator panel allows you to set warnings on entering or leaving a site that uses SSL for security, and for using encrypted information with Web servers. The section named Certificate to identify you to a Web site gives you the possibility to select whether Netscape will ask you which certificate you want to use in responding to a server if you have more than one certificate. You can also select a certificate to respond with by default. Some Web servers might only accept certificates by specific signers or certificates exceeding a particular grade. If you frequently transact with multiple sites that have conflicting requirements, you should select **Ask Every Time**.



These settings allow you to control Navigator security settings.

Navigator security warnings can let you know before you do something that might be unsafe.

Show a warning before:

- ☑ Entering an encrypted site
- ☑ Leaving an encrypted site
- ☑ Viewing a page with an encrypted/unencrypted mix
- ☑ Sending unencrypted information to a Site

Certificate to identify you to a web site:

Ask Every Time

- Ask Every Time
- Select Automatically
- David Jackson's VeriSign Inc. ID
- David Jackson's _Your Organization Name Here_ ID
- Itso B Tester's VeriSign, Inc. ID

- ☐ ... SSL v2
- ☑ Enable SSL (Secure Sockets Layer) v3 — Configure SSL v3

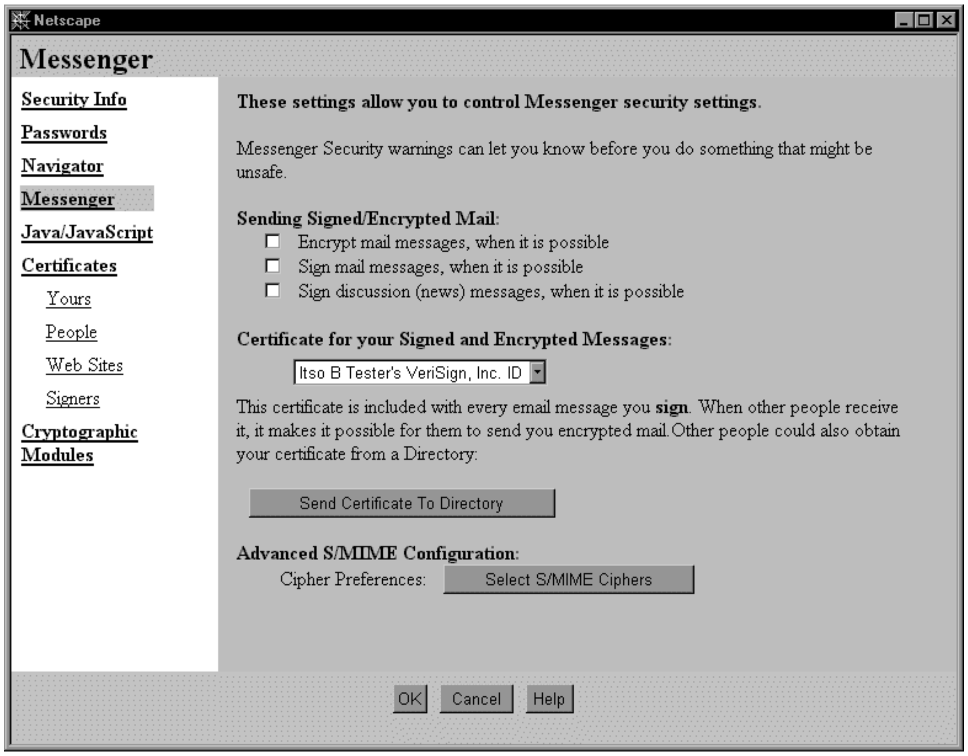
Figure 183. Certificate Information List Box

Finally, the above window gives you the ability to enable SSL Version 2 and SSL Version 3. SSL Version 3 is the latest and most secure version of SSL. Version 2 is included for compatibility with older servers.

SSL Version 3.0 vs. Version 2.0

SSL Version 3.0 enables client authentication. Actually client authentication was possible also in Version 2.0 but it presented some bugs. Those bugs were subsequently fixed, but not all the browsers picked up the changes, as we discuss in 3.1.1, “SSL Overview” on page 69. Those fixes have been completely integrated in the new version of SSL, Version 3.0, which offers other advantages in terms of security that were not present in Version 2.0. It uses a more secure implementation of MD5 message authentication to detect attempts to modify data in transit. It also implements SHA (Secure Hash Algorithm) message authentication. SHA is a government-standardized algorithm that is used to construct a message authentication code that detects attempts to modify data while in transit. SHA is slower than MD5 but it is stronger.

The Messenger panel, which you access by clicking on **Messenger**, allows you to set certificate options for sending secure e-mail and news group items.



These settings allow you to control Messenger security settings.

Messenger Security warnings can let you know before you do something that might be unsafe.

Sending Signed/Encrypted Mail:
- ☐ Encrypt mail messages, when it is possible
- ☐ Sign mail messages, when it is possible
- ☐ Sign discussion (news) messages, when it is possible

Certificate for your Signed and Encrypted Messages:

Itso B Tester's VeriSign, Inc. ID

This certificate is included with every email message you **sign**. When other people receive it, it makes it possible for them to send you encrypted mail. Other people could also obtain your certificate from a Directory:

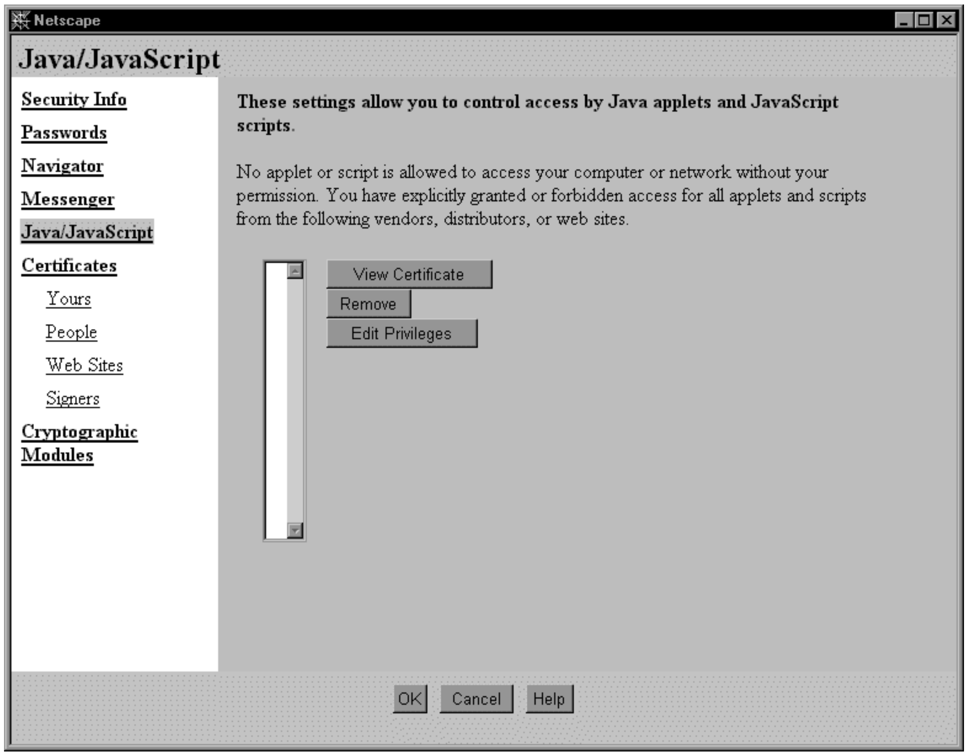
Advanced S/MIME Configuration:
Cipher Preferences:

Figure 184. Netscape Security Info - Messenger

From this window, you can select a certificate to use for your Messenger operations. The certificate that appears highlighted in the section named Certificate for your Signed and Encrypted Messages will be included with every e-mail that you sign. This way, when your correspondent receives it, it is possible for them to send you encrypted e-mail by using your public key, which is part of your digital certificate. Clicking the **Send Certificate to Directory** button sends your certificate to an LDAP directory. This makes it easily accessible for others to be able to send you encrypted e-mail without having first received your certificate in another way.

Click the **Select S/MIME Ciphers** to indicate types of ciphers you prefer a correspondent uses when encrypting and sending a message to you. Your enabled ciphers are included in signed, outgoing messages. However, this does not prevent a correspondent from encrypting a message to you with a type of cipher you have disabled.

The following screen shows the section you obtain after clicking on **Java/JavaScript**:



These settings allow you to control access by Java applets and JavaScript scripts.

No applet or script is allowed to access your computer or network without your permission. You have explicitly granted or forbidden access for all applets and scripts from the following vendors, distributors, or web sites.

Figure 185. Netscape Security Info - Java/Javascript

The Java/Javascript panel shows a list of certificates, which in the above figure is empty. A digital certificate will be listed in the Java/Javascript panel when you receive on your browser a Java applet or a JavaScript file signed by the owner of the certificate and you will ask Communicator to remember your choice, as we will show in 5.3, “Interacting with Signed Java Applets” on page 224.

Each certificate represents a trusted entity, and using this panel you can review the access permissions granted to Java applets and JavaScript files signed by those entities. You have the ability to see each certificate, to remove it from that list and, by clicking on **Edit Privileges** when a certificate in the list is highlighted, you can choose specific privileges you want to grant for the current session only or for any future sessions. You may also be able to deny specific privileges.

Signed applets in the JDK 1.2 security model are examined in 2.7.2, “Signed Applets” on page 56. See also 5.3, “Interacting with Signed Java Applets” on page 224 for more details on the JDK 1.1 security model, on which the Java Virtual Machine (JVM) implemented on Netscape Communicator 4.05 is based.

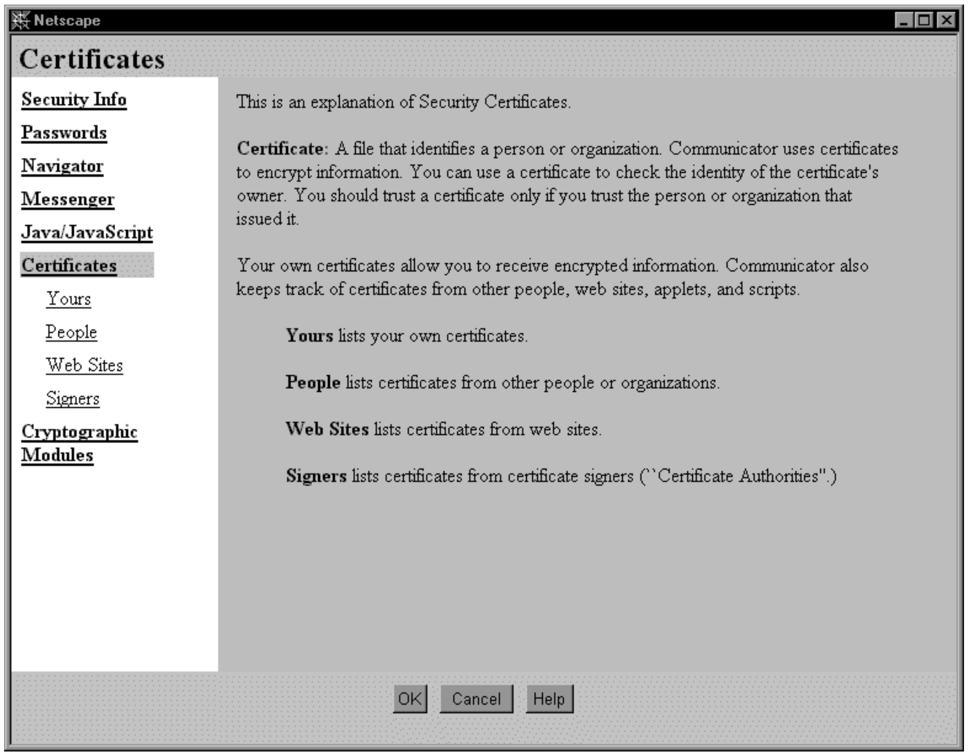
Netscape Communicator JDK 1.1 Level

Note that the level of the JDK in Netscape Communicator 4.05 is still based on a 1.1 beta version from JavaSoft. Even if the current version of the JDK is 1.1.6, Communicator is not implementing a full JDK 1.1 support.

Also note that not all base parts of the JDK are shipped with the Netscape browser. There is a missing part that is java.security so any applications that make use of this will not run in a Netscape JVM. This limitation causes an overhead penalty in terms of Java portability. Since Netscape Communicator 4.05 does not ship a java.security set of classes, applets cannot depend on these being on the platform.

However, Netscape's longer-term strategy is to move towards an open Java API, as you can learn from the Netscape's Web site http://developer.netscape.com/tech/java/index.html. Other third-party JVMs will be allowed to run inside of Communicator by replacing the default Netscape's JVM implementation. This will give more flexibility to application developers, allowing them to use whatever JVM makes the most sense for their application.

Clicking on **Certificates** will display the Certificates section:



This is an explanation of Security Certificates.

Certificate: A file that identifies a person or organization. Communicator uses certificates to encrypt information. You can use a certificate to check the identity of the certificate's owner. You should trust a certificate only if you trust the person or organization that issued it.

Your own certificates allow you to receive encrypted information. Communicator also keeps track of certificates from other people, web sites, applets, and scripts.

Yours lists your own certificates.

People lists certificates from other people or organizations.

Web Sites lists certificates from web sites.

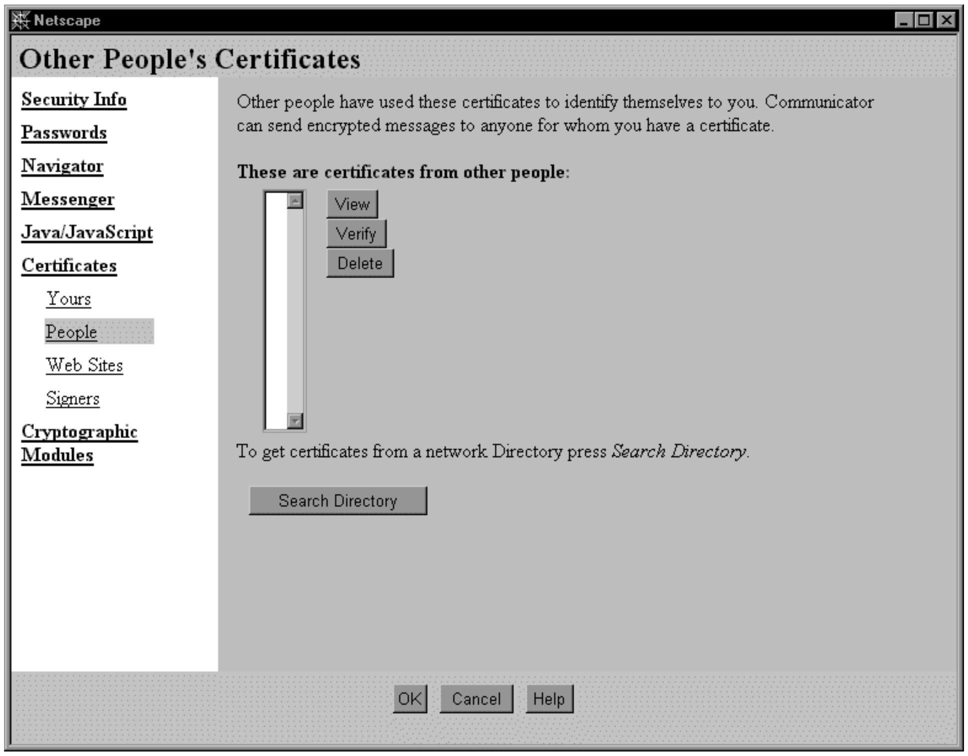
Signers lists certificates from certificate signers (``Certificate Authorities''.)

Figure 186. Netscape Security Info - Certificates

The Certificates section is used to perform functions related to receiving and using digital certificates. Information about most of the functions in the Certificates section follows in 5.4, "How to Manage Client Certificates" on page 232. See in particular 5.4.1, "Obtaining a VeriSign Evaluation Certificate" on page 232 for information on getting a certificate with the section labeled Yours. Figure 227 on page 233 shows the feature used to export a digital certificate to a file by selecting a certificate and clicking on the **Export** button. In 5.4.4, "How to Export and Import

Certificates” on page 239 we show the export steps with Netscape Communicator and then the import steps with Internet Explorer.

The next certificate section is related to the certificates other people have sent to you or ones that you have retrieved from a directory service. You can access this section by clicking on **People**. When another person sends you a signed e-mail, their certificate is sent along with it to allow Messenger to verify the signature. You have the option to keep their certificate so that you can send them encrypted e-mail in the future. Other People's Certificates are reviewed with the next panel.



Other people have used these certificates to identify themselves to you. Communicator can send encrypted messages to anyone for whom you have a certificate.

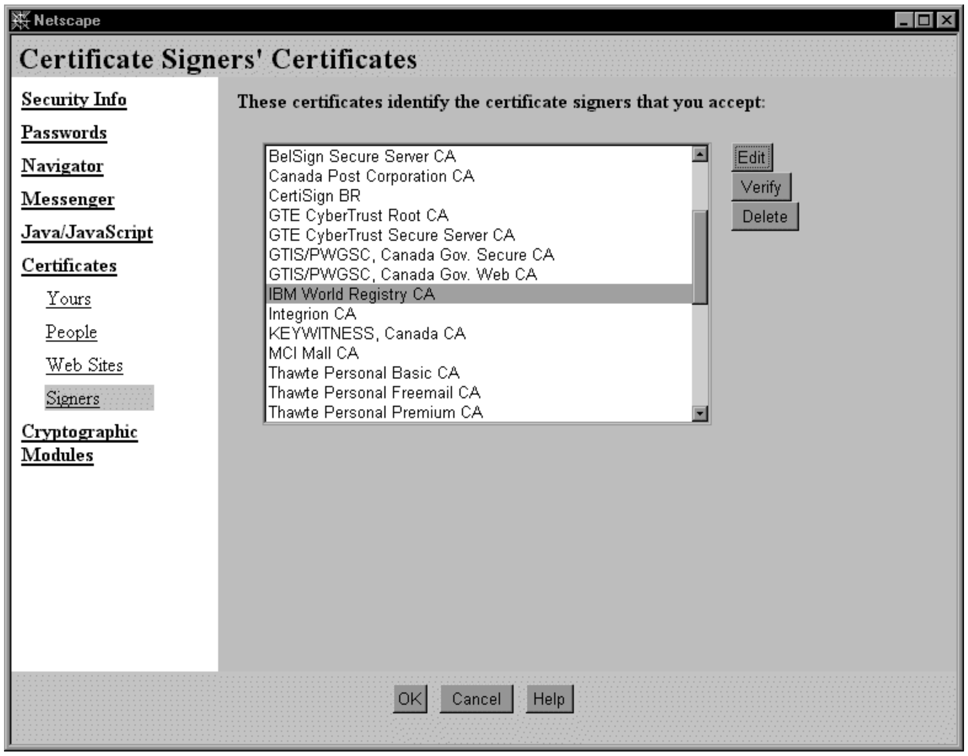
These are certificates from other people:

To get certificates from a network Directory press *Search Directory*.

Figure 187. Netscape Security Info - Other People's Certificates

All the certificates listed in the certificate list were sent to you in e-mail messages. Use this section to verify, view or delete certificates in the list. Notice in particular that verifying a certificate is an operation you might want to do to ensure that a given certificate was issued by the signing authority the certificate claims.

As we discuss in 3.1.1, “SSL Overview” on page 69 and 3.1.3, “Lotus Domino Go Webserver SSL Server Authentication” on page 79, the most common use of SSL is called server authentication. Here the server sends its digital certificate to the client browser to begin the handshake. The client generates a random number and encrypts it in the public key received as part of the server's certificate. This is called a challenge. If the server can successfully decrypt the random number, it proves that the private key corresponding to the public key presented in the server certificate is held by the server. As long as the certificate is signed by a trusted CA, the SSL establishment will continue. What determines whether this server certificate is trusted or not? The digital certificates of several CAs are shipped with the browser, and referred to when the server certificate is presented. This screen is from the Netscape browser, showing the CAs delivered with the browser. You can access this page by clicking on **Signers**.



These certificates identify the certificate signers that you accept:

BelSign Secure Server CA
Canada Post Corporation CA
CertiSign BR
GTE CyberTrust Root CA
GTE CyberTrust Secure Server CA
GTIS/PWGSC, Canada Gov. Secure CA
GTIS/PWGSC, Canada Gov. Web CA
IBM World Registry CA
Integrion CA
KEYWITNESS, Canada CA
MCI Mall CA
Thawte Personal Basic CA
Thawte Personal Freemail CA
Thawte Personal Premium CA

Figure 188. Netscape Certificate Signers' Certificates

By default, after installing Netscape Communicator 4.05, the following signers' certificates are considered trusted:

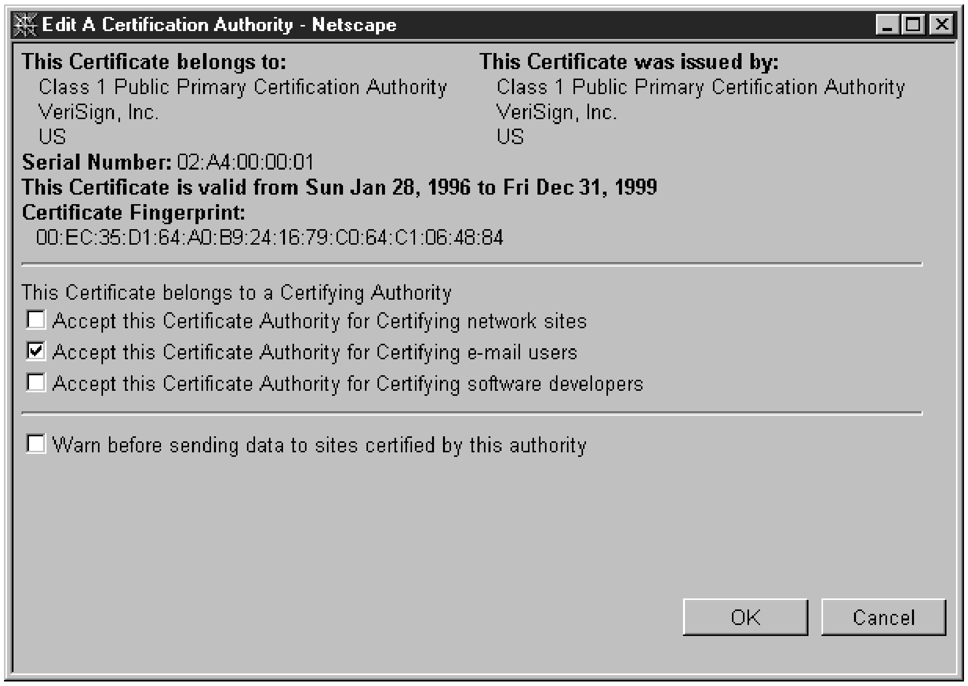
1. AT&T Certificate Services
2. AT&T Directory Services
3. BBN Certificate Services CA Root 1
4. BelSign Class 1 CA
5. BelSign Class 2 CA
6. BelSign Class 3 CA
7. BelSign Object Publishing CA
8. BelSign Secure Server CA
9. Canada Post Corporation CA
10. CertiSign BR
11. GTE Cyber Trust Root CA
12. GTE Cyber Trust Secure Server CA
13. GTIS/PWGSC, Canada Gov. Secure CA
14. GTIS/PWGSC, Canada Gov. Web CA
15. IBM World Registry CA
16. Integrion CA
17. KEYWITNESS, Canada CA
18. MCI Mall CA

19. Thawte Personal Basic CA
20. Thawte Personal Freemail CA
21. Thawte Personal Premium CA
22. Thawte Server CA
23. Uptime Group Plc. Class 1 CA
24. Uptime Group Plc. Class 2 CA
25. Uptime Group Plc. Class 3 CA
26. Uptime Group Plc. Class 4 CA
27. VeriSign Class 1 Primary CA
28. VeriSign Class 2 Primary CA
29. VeriSign Class 3 Primary CA
30. VeriSign Class 4 Primary CA
31. VeriSign/RSA Commercial CA
32. VeriSign/RSA Secure Server CA

You can compare this list with the one that you have on your Netscape Communicator to trust that these certificates have not been compromised. You can highlight each certificate in the list and then press the **Edit** button to see more details about the specific certificate and also to select one of more of the following three options:

1. **Accept this Certificate Authority for Certifying network sites**
2. **Accept this Certificate Authority for Certifying e-mail users**
3. **Accept this Certificate Authority for Certifying software developers**

You can also mark the check box **Warn me before sending data to sites certified by this authority**. The following figure shows what we got when we highlighted **VeriSign Class 1 Primary CA** and then press the **Edit** button:



Edit A Certification Authority - Netscape

This Certificate belongs to:
Class 1 Public Primary Certification Authority
VeriSign, Inc.
US

This Certificate was issued by:
Class 1 Public Primary Certification Authority
VeriSign, Inc.
US

Serial Number: 02:A4:00:00:01
This Certificate is valid from Sun Jan 28, 1996 to Fri Dec 31, 1999
Certificate Fingerprint:
00:EC:35:D1:64:A0:B9:24:16:79:C0:64:C1:06:48:84

This Certificate belongs to a Certifying Authority
- ☐ Accept this Certificate Authority for Certifying network sites
- ☑ Accept this Certificate Authority for Certifying e-mail users
- ☐ Accept this Certificate Authority for Certifying software developers

- ☐ Warn before sending data to sites certified by this authority

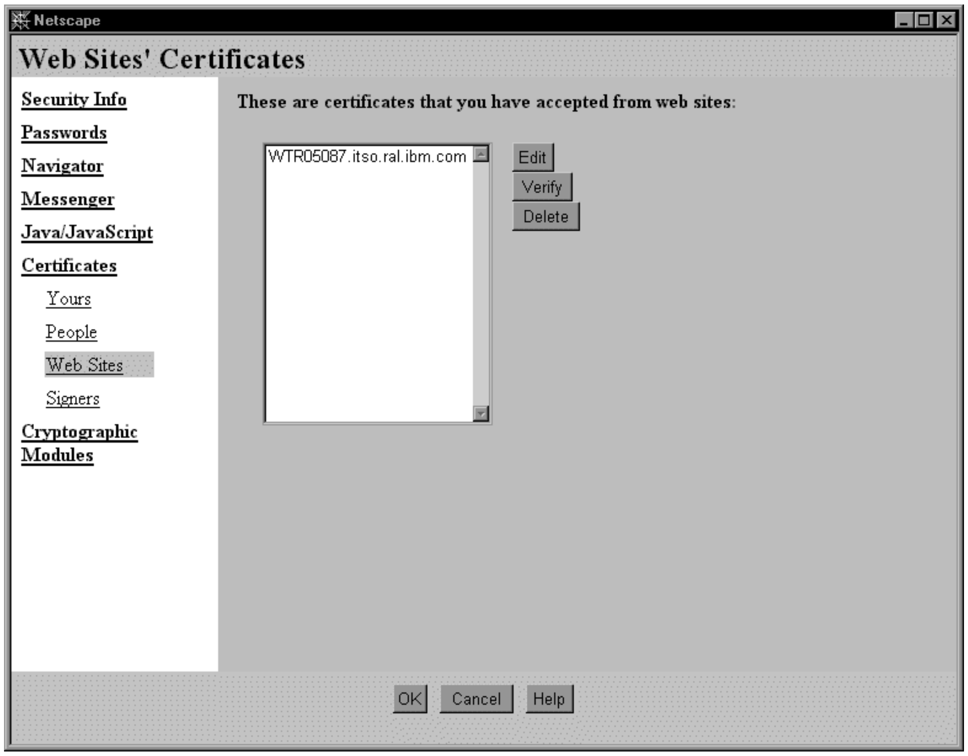
Figure 189. Editing Properties for VeriSign Class 1 Primary CA

In the above CA list you have noticed that there are several different classes of certificates that VeriSign can sign, corresponding to different grades of security. When an individual applies for a certificate being signed by VeriSign, in general no VeriSign employee will meet that individual in order to authenticate their identity. More likely, that individual just has to fill out a form from a Web page (see for example 5.4.1, “Obtaining a VeriSign Evaluation Certificate” on page 232). Such a form asks the requester the name, organization, country and e-mail address, plus some more information. In general the key (or directions on how to fetch the key) is mailed to that e-mail address. Thus you may reasonably trust that the e-mail address is genuine, but in effect the requester could have filled in any name and organization. With a Class 1 ID from VeriSign, that information is not verified, and this is the reason why options 1 on page 192 and 3 on page 192 by default are not selected for VeriSign Class 1 Primary CA, as shown in Figure 189. However there are more stringent classes of IDs. With higher classes of IDs, VeriSign will require the requester to appear before a notary public, will check the financial rating of the requester, etc., and all the options shown in Figure 189 are selected for the VeriSign CAs corresponding to classes higher than Class 1.

Other CAs may have different procedures to authenticate users. However, when you receive an authenticated message, it is important that you know what, in effect, has been authenticated.

Notice that if the certificate is signed by someone unknown, the user will be prompted with a series of screens, allowing the user to decide whether or not to trust the server connection, because the signer of the server's certificate is unknown. We describe this entire process in 3.1.3, “Lotus Domino Go Webserver SSL Server Authentication” on page 79.

The next time you connect with SSL to the site for whom you have accepted the site certificate, Netscape will connect to the site using the certificate just accepted to verify the connection. It will remember the settings chosen in the panels, and warn you if you selected the option for warning before sending information to the site. Certificates for servers you have accepted in this manner can be reviewed in the Netscape Security Information panel, by clicking on **Web Sites**.



These are certificates that you have accepted from web sites:

WTR05087.itso.ral.ibm.com

Figure 190. Web Sites' Certificates

The Cryptographic Modules section can be accessed by clicking on **Cryptographic Modules**, in the Security Info panel. Cryptographic modules are loadable pieces of software that provide a function of cryptographic services, such as smart card support (including PCMCIA smart cards and disk-based smart cards), hardware-accelerated cryptography and new ciphers.

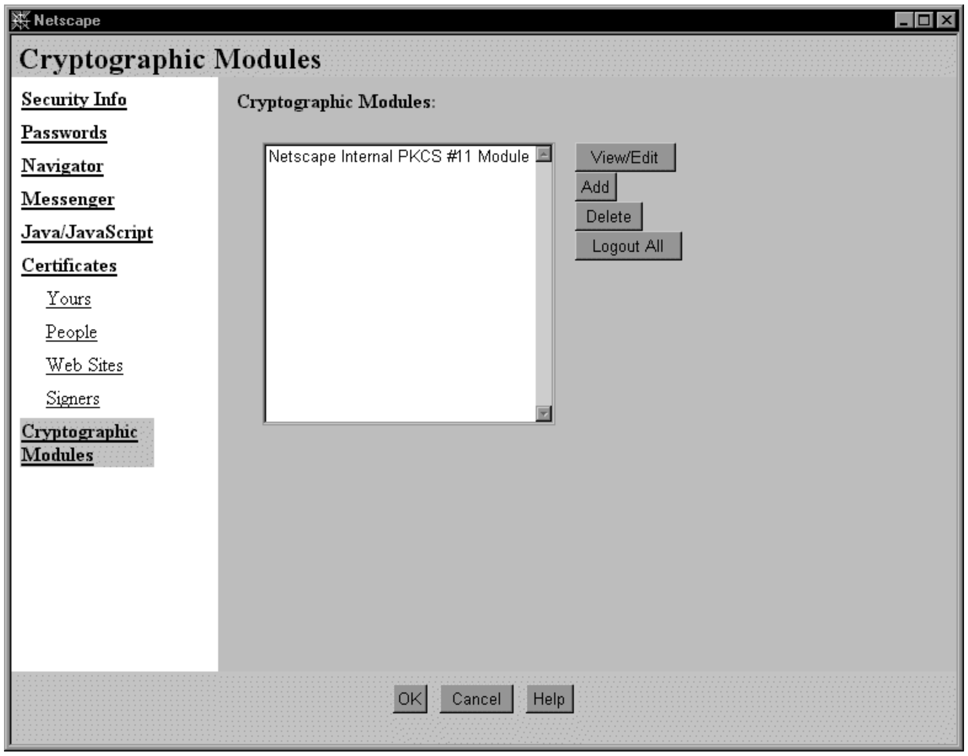


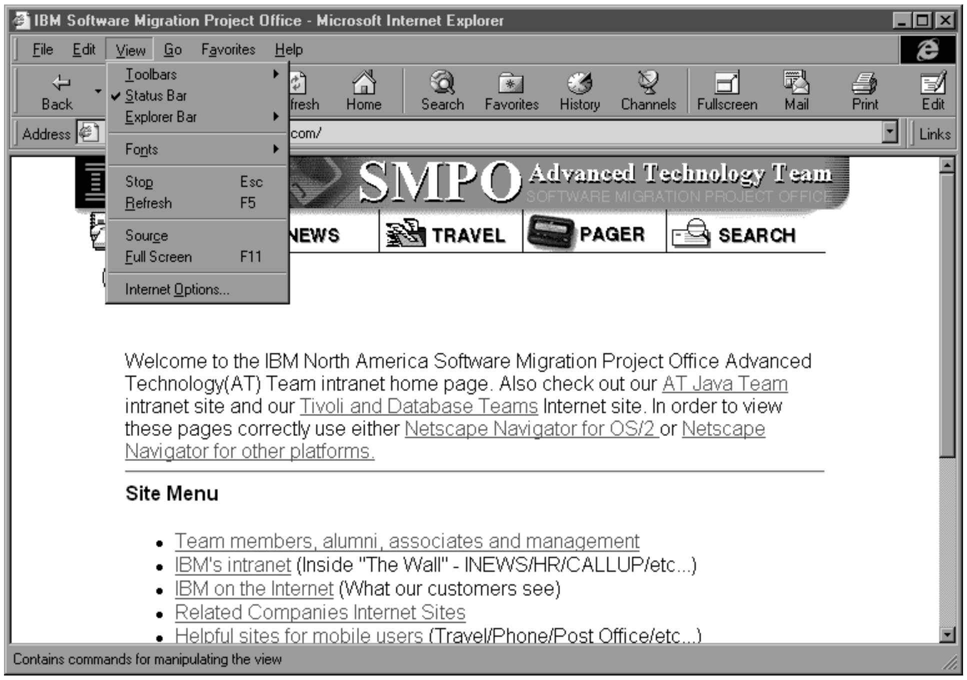
Figure 191. Netscape Cryptographic Modules Panel

Netscape Internal PKCS #11 Module is the cryptographic module that is used and shipped by Netscape Communicator. It is used for smart card support.

5.1.2 Microsoft Internet Explorer

We had the opportunity to work with Release 4.01 for Microsoft Internet Explorer (MSIE).

The security information for MSIE is found in the Internet Options, accessed from the View menu as shown next.


IBM Software Migration Project Office - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit
Toolbars
✓ Status Bar
Explorer Bar
Fonts
Stop Esc
Refresh F5
Source
Full Screen F11
Internet Options...
Address .com/ Links
SMPO Advanced Technology Team
SOFTWARE MIGRATION PROJECT OFFICE
NEWS TRAVEL PAGER SEARCH

Welcome to the IBM North America Software Migration Project Office Advanced Technology(AT) Team intranet home page. Also check out our AT Java Team intranet site and our Tivoli and Database Teams Internet site. In order to view these pages correctly use either Netscape Navigator for OS/2 or Netscape Navigator for other platforms.

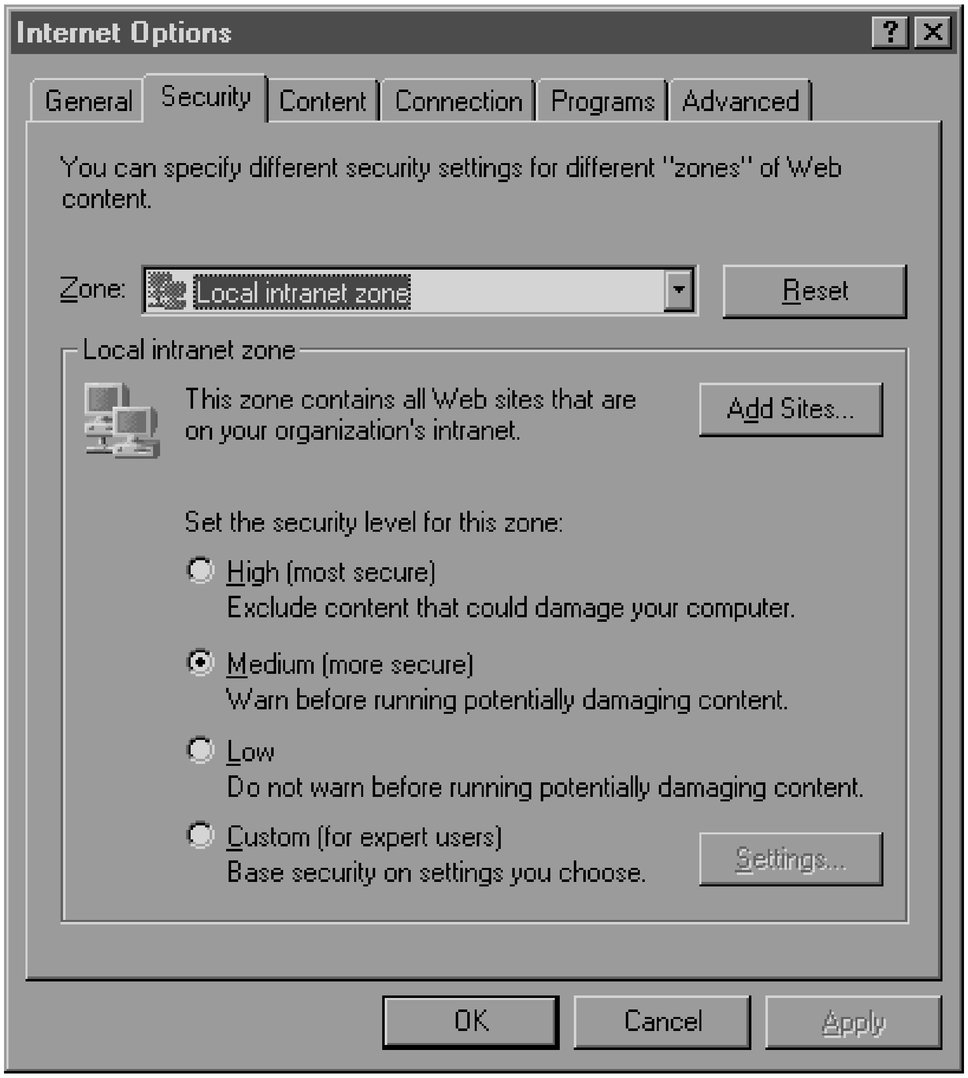
Site Menu

- Team members, alumni, associates and management
- IBM's intranet (Inside "The Wall" - INEWS/HR/CALLUP/etc...)
- IBM on the Internet (What our customers see)
- Related Companies Internet Sites
- Helpful sites for mobile users (Travel/Phone/Post Office/etc...)

Contains commands for manipulating the view

Figure 192. Microsoft Internet Explorer

Microsoft's security design for Internet Explorer is based upon the concept of *Security Zones*, as you first notice on the Security page for the Internet Options:


Internet Options

You can specify different security settings for different "zones" of Web content.

Zone: Local intranet zone

Local intranet zone

This zone contains all Web sites that are on your organization's intranet.

Set the security level for this zone:

- High (most secure)
  Exclude content that could damage your computer.
- [✓ Medium (more secure)]
  Warn before running potentially damaging content.
- Low
  Do not warn before running potentially damaging content.
- Custom (for expert users)
  Base security on settings you choose.

Figure 193. Microsoft Internet Explorer - Internet Options

With the Security Zones approach, you can decide how much access to allow to visitors to your computer. Web sites that you trust - such as those on you company's intranet or from established companies in whom you have confidence - you can assign as trusted, permitting them even to run executable content on your computer. On the other hand, you can strictly limit Web sites that you do not want to trust to access your computer.

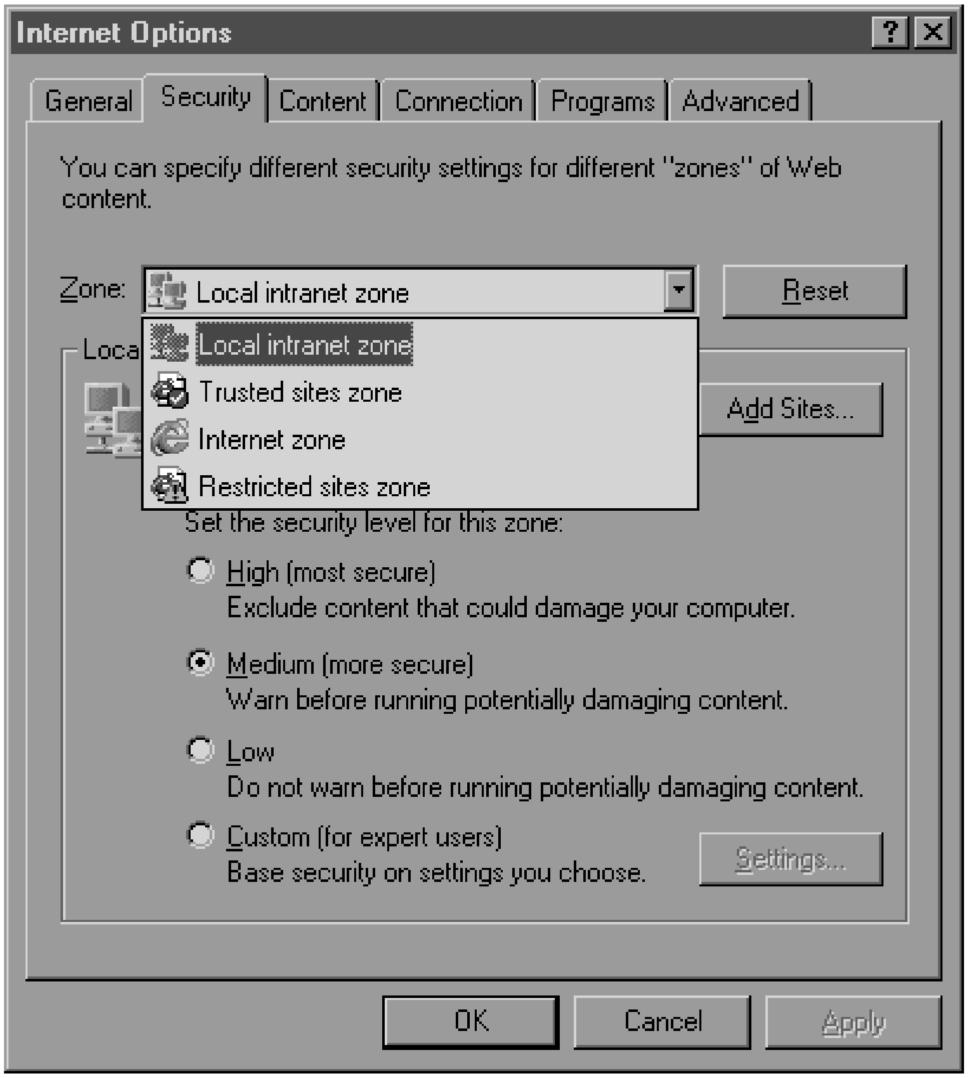
Security Zones offer the advantage of providing advanced protection for your computer and your privacy without interrupting you with repeated warnings while you are visiting Web sites that you have already decided to trust. Moreover it is also possible for companies to set automatic boundaries so their user do not have to make security decisions on a case-by-case basis.

Microsoft has set four pre-defined Security Zones:

1. Local intranet zone
2. Trusted sites zone
3. Internet zone

4. Restricted sites zone

These zones are visible by opening the Zone list box, as shown in the following screen:



You can specify different security settings for different "zones" of Web content.

Zone: Local intranet zone

Local intranet zone
Trusted sites zone
Internet zone
Restricted sites zone

Set the security level for this zone:

( ) High (most secure)
Exclude content that could damage your computer.

(•) Medium (more secure)
Warn before running potentially damaging content.

( ) Low
Do not warn before running potentially damaging content.

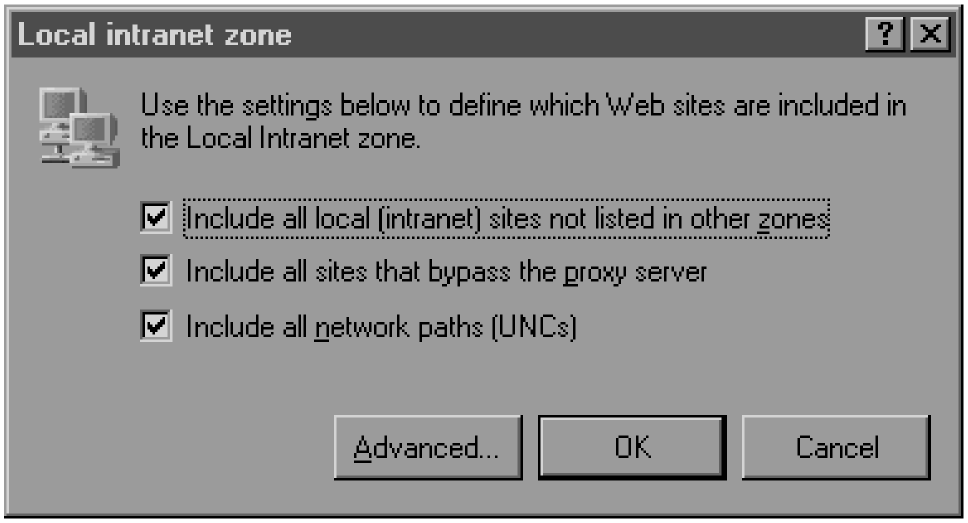
( ) Custom (for expert users)
Base security on settings you choose.

Figure 194. Internet Options - Zones List Box

The browser notifies you at the bottom of the screen which zone is in effect.

For each of these zones, security settings of **High**, **Medium** and **Low** can be selected, but there is also an option to select your own settings for each zone with **Custom**. Using this dialog box, you can set the security options you want for each zone and then add or remove sites from the zones depending on your level of trust in the site. In corporate environments, administrators can tailor these four zones for users and even add or remove the authentication certificates of software publishers that they do or do not trust in advance. This way users do not have to make security decisions while using the Internet.

For the Local intranet zone, the default security setting is Medium. By clicking on the **Add Sites...** button you can select areas to include in the Local zone.


Local intranet zone

Use the settings below to define which Web sites are included in the Local Intranet zone.

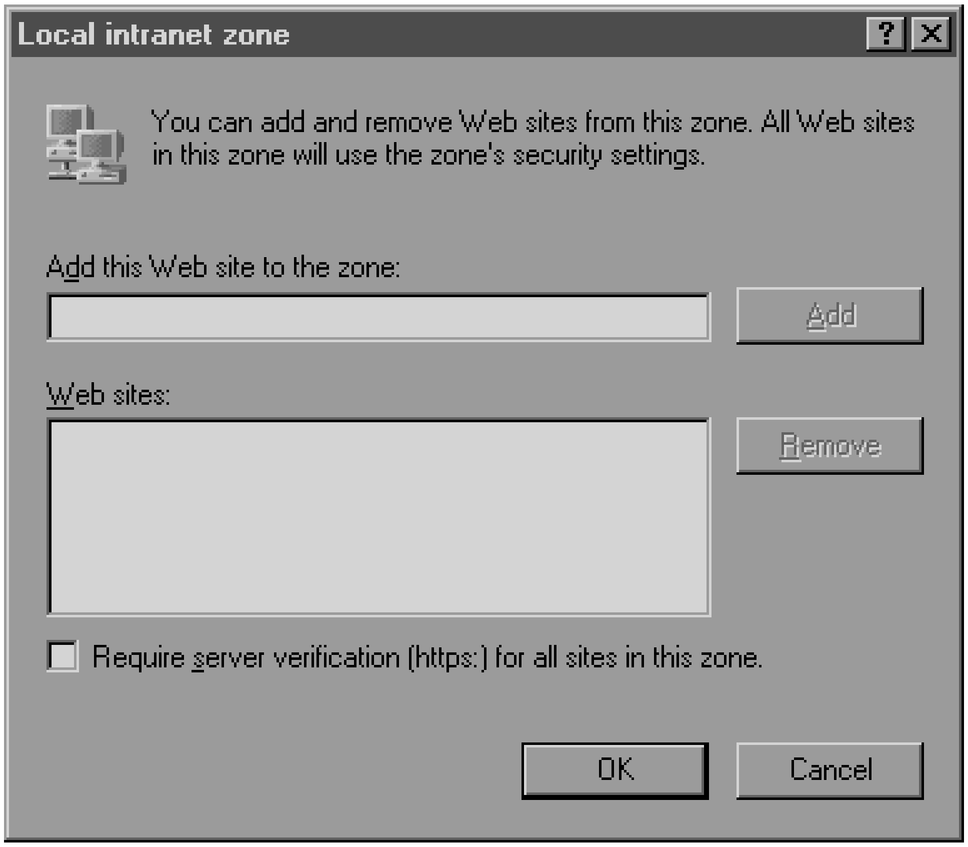
☑ Include all local (intranet) sites not listed in other zones

☑ Include all sites that bypass the proxy server

☑ Include all network paths (UNCs)

Figure 195. Internet Options - Include Sites in Intranet Zone

You are able to specify sites by name and add them to the zone by clicking on the **Advanced** button.


Local intranet zone

You can add and remove Web sites from this zone. All Web sites in this zone will use the zone's security settings.

Add this Web site to the zone:

Web sites:

☐ Require server verification (https:) for all sites in this zone.

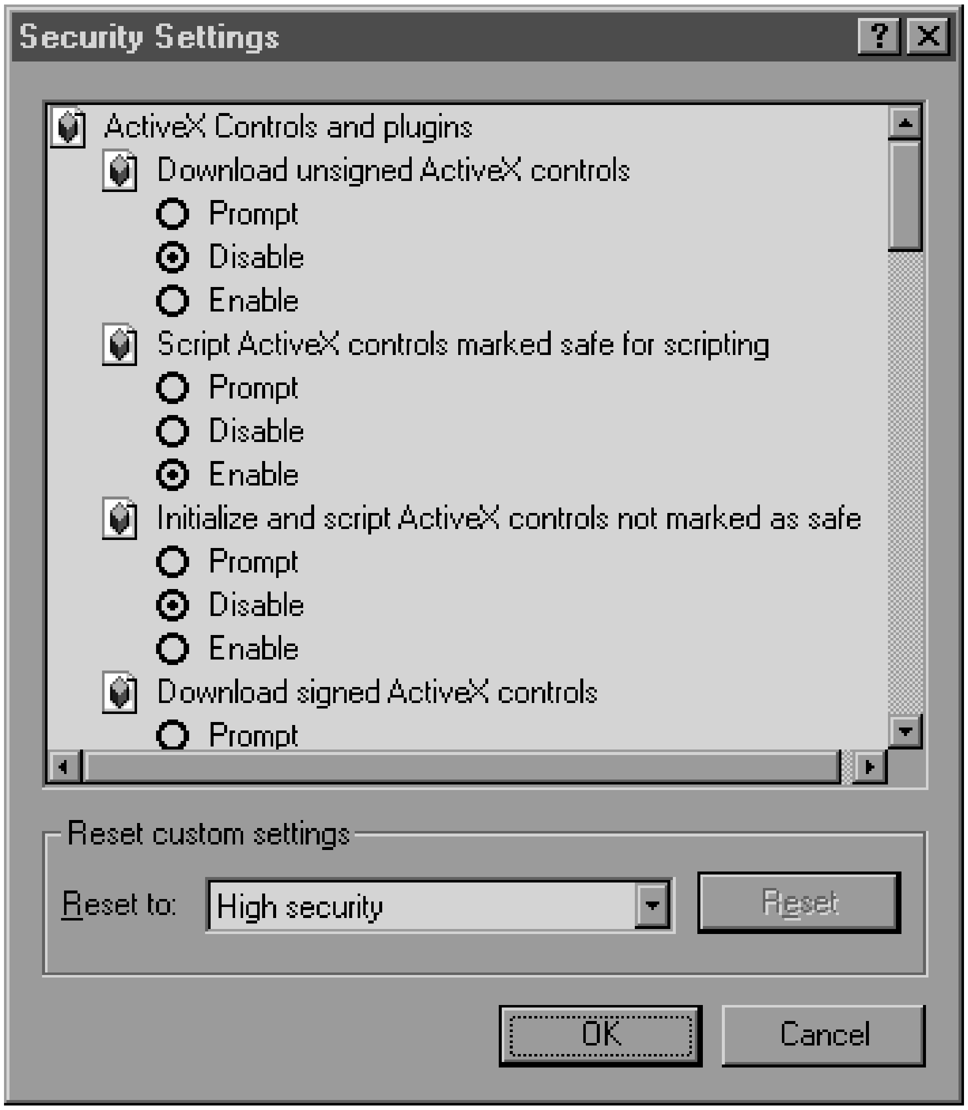
Figure 196. Internet Options - Add Sites to Intranet - Advanced

The Trusted sites, Internet and Restricted sites zones default respectively to Low, Medium and High security settings. Notice in particular that the Restricted sites zone is considered the high risk zone, and so the default settings is for the most security.

You can add sites to these three zones by clicking on the **Add Sites...** button. The dialog to add sites, similar to Figure 196, is available directly in these case, without the Include dialog that appeared for the Local intranet zone section (see Figure 195).

Next we will look at the Custom Settings that can be made for each zone. To get to the Custom settings, select the **Custom** radio button and click on the **Settings...** button (see Figure 193 on page 197). The Security Settings dialog is brought up, as shown in Figure 197. By using the **Reset** custom settings button at the bottom of the Security Settings dialog, you can also see what choices are made for the High, Medium and Low security settings.

The first set of choices in the Security Settings deals with ActiveX controls.



ActiveX Controls and plugins
- Download unsigned ActiveX controls
  - Prompt [✓ Disable] Enable
- Script ActiveX controls marked safe for scripting
  - Prompt Disable [✓ Enable]
- Initialize and script ActiveX controls not marked as safe
  - Prompt [✓ Disable] Enable
- Download signed ActiveX controls
  - Prompt

Reset custom settings

Reset to: High security

Figure 197. Security Settings - ActiveX Controls

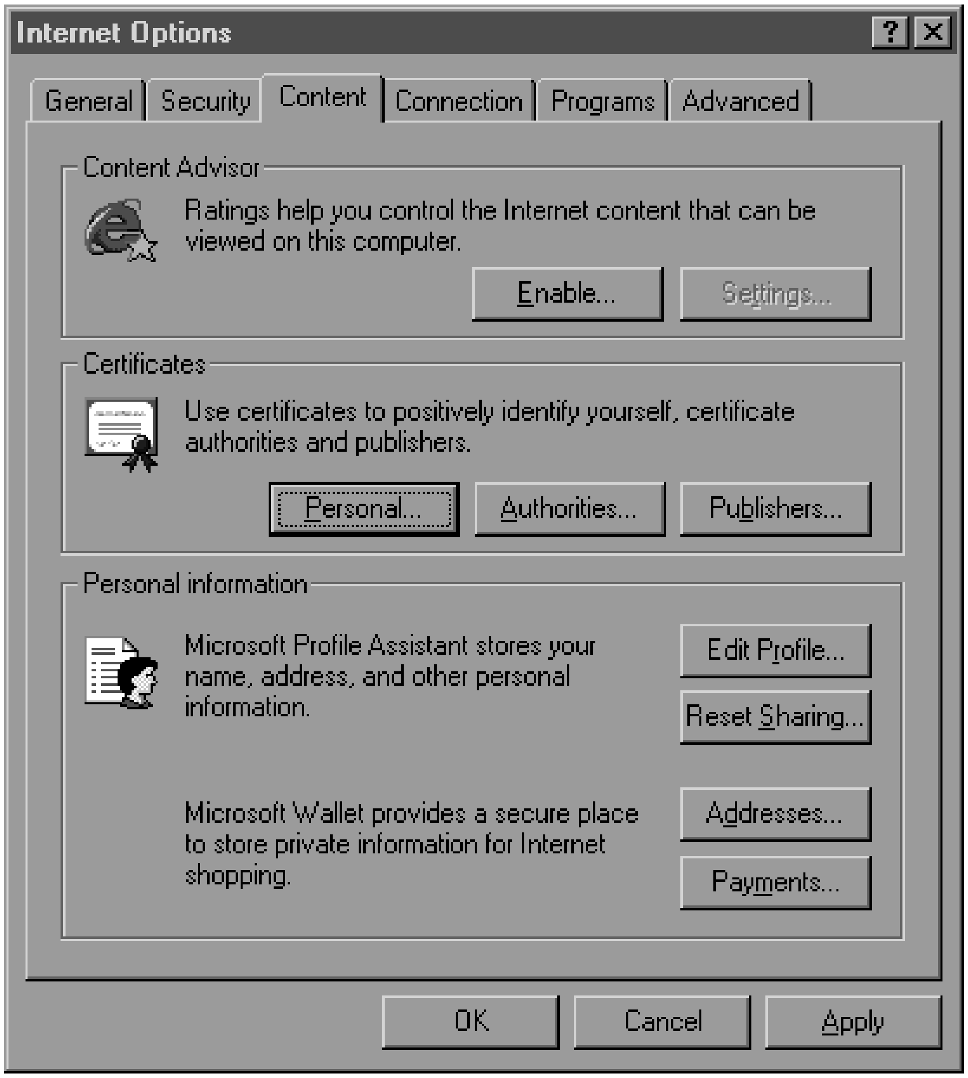
ActiveX controls are compiled system code and do not run in a sandbox like Java applets. They can do anything to your systems that you can do. There is no granularity of permissions in signed ActiveX controls; you can only deny or grant them permission to run. Many sources consider ActiveX controls to be dangerous when used from the internet because of the lack of a sandbox mechanism to contain their activity. Consider carefully your choices for the Security Settings for ActiveX controls.

The next portion of the Security Settings gives advanced users and administrators more control over all other security options:

- Font and file downloads
- Password protection
- The level of capabilities given to Java applets

In particular, with the password protection Security Settings, depending on which Zone a server is in, Internet Explorer can send password information automatically, prompt the user for user ID and password information or simply deny any login request.

The Content page of the Internet Options has settings for the Content Advisor, Certificates, and Personal Information. You can access this page by clicking on the **Content** tab for the Internet Options window.


Internet Options

Content Advisor: Ratings help you control the Internet content that can be viewed on this computer.

Certificates: Use certificates to positively identify yourself, certificate authorities and publishers.

Personal information: Microsoft Profile Assistant stores your name, address, and other personal information.

Microsoft Wallet provides a secure place to store private information for Internet shopping.

Figure 198. Internet Options - Content Page

The Content Advisor section allows you to set controls based on content ratings for Internet pages. These are sometimes referred to as parental controls, and are meant to stop access to Web pages of a sexual or violent nature.

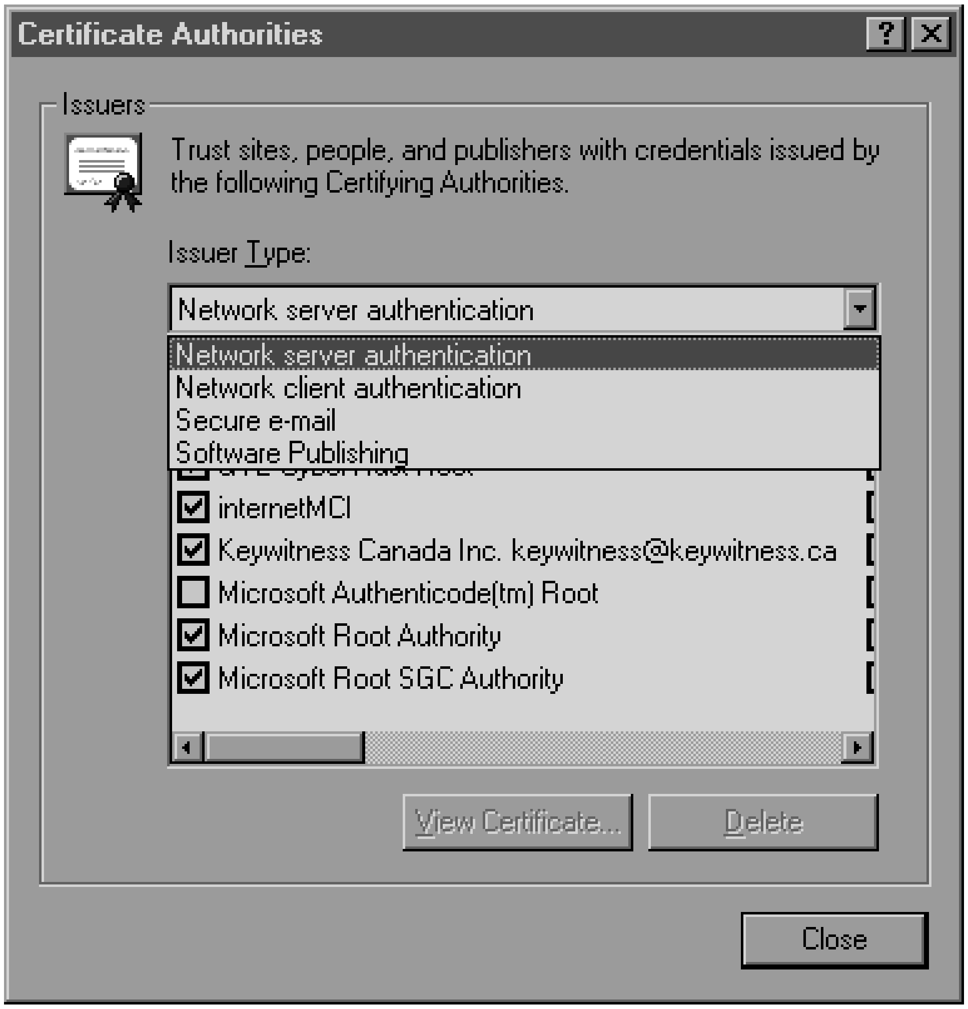
The Personal Information section allows you to fill out business card type information which can be sent along to places you choose.

To use the Certificates section, click on the **Personal** button and you see a dialog that shows all your personal digital certificates (see Figure 236 on page 241 and Figure 238 on page 242). We will show more details on how to use this section in 5.4.4, "How to Export and Import Certificates" on page 239, where we will export a certificate with Netscape Communicator and import it with Microsoft Internet Explorer.

Like Netscape Communicator (see Figure 188 on page 191), MSIE provides a listing of certificates for CAs who might sign server or user certificates sent to you. You can access the Certificate Authorities listing by clicking on the **Authorities** button on the Content page, shown in Figure 198 on page 201. The CAs that MSIE lists here are more or less the same ones that Netscape Communicator also lists (see Figure 188 on page 191). The only difference is that Microsoft has arranged them in groupings related to their most common usage:

1. Network server authentication
2. Network client authentication
3. Secure e-mail
4. Software Publishing

as shown in the following figure:


Certificate Authorities

Issuers

Trust sites, people, and publishers with credentials issued by the following Certifying Authorities.

Issuer Type: Network server authentication

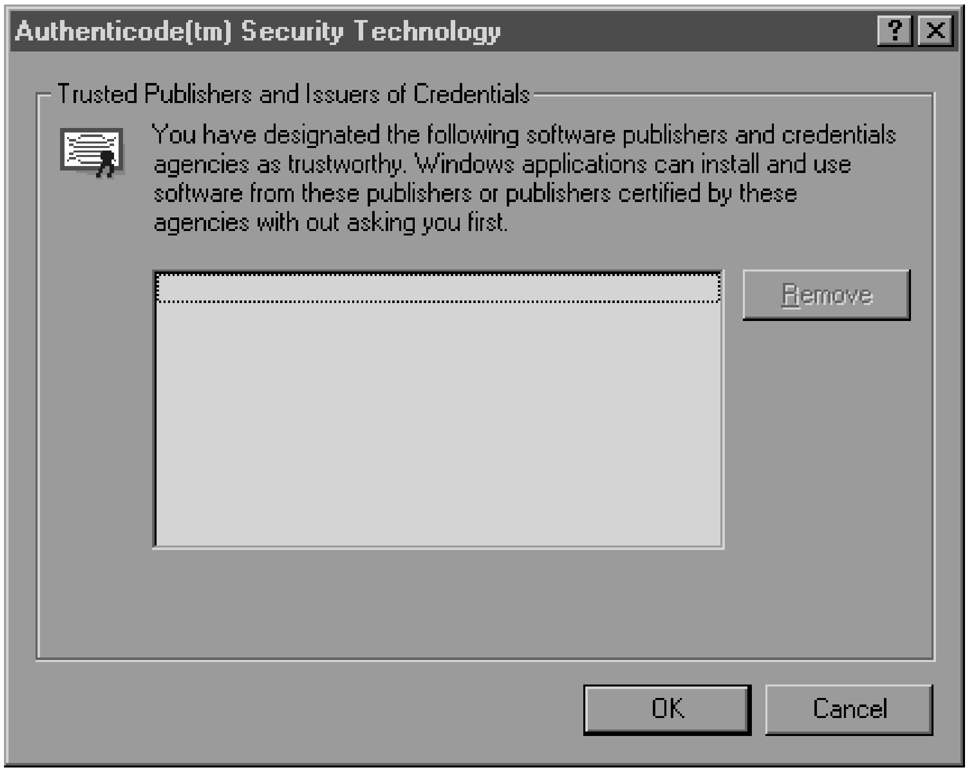
- Network server authentication
- Network client authentication
- Secure e-mail
- Software Publishing

- [✓] internetMCI
- [✓] Keywitness Canada Inc. keywitness@keywitness.ca
- [ ] Microsoft Authenticode(tm) Root
- [✓] Microsoft Root Authority
- [✓] Microsoft Root SGC Authority

Figure 199. Certificate Authorities - List Box

Taking again VeriSign Class 1 Primary CA as an example, we see that Microsoft sets all the individuals with credentials issued by this CA as untrusted for Network server authentication and Software Publishing, as trusted for Network client authentication and Secure e-mail.

Clicking on the **Publishers** button on the Content page shown in Figure 198 on page 201 will show you the listing of the software publishers you have accepted for remote installation using the Microsoft Authenticode identification process. The selections listed here allow the software sent by the publisher to be installed without your notification. As you can see, that list was empty in our configuration.



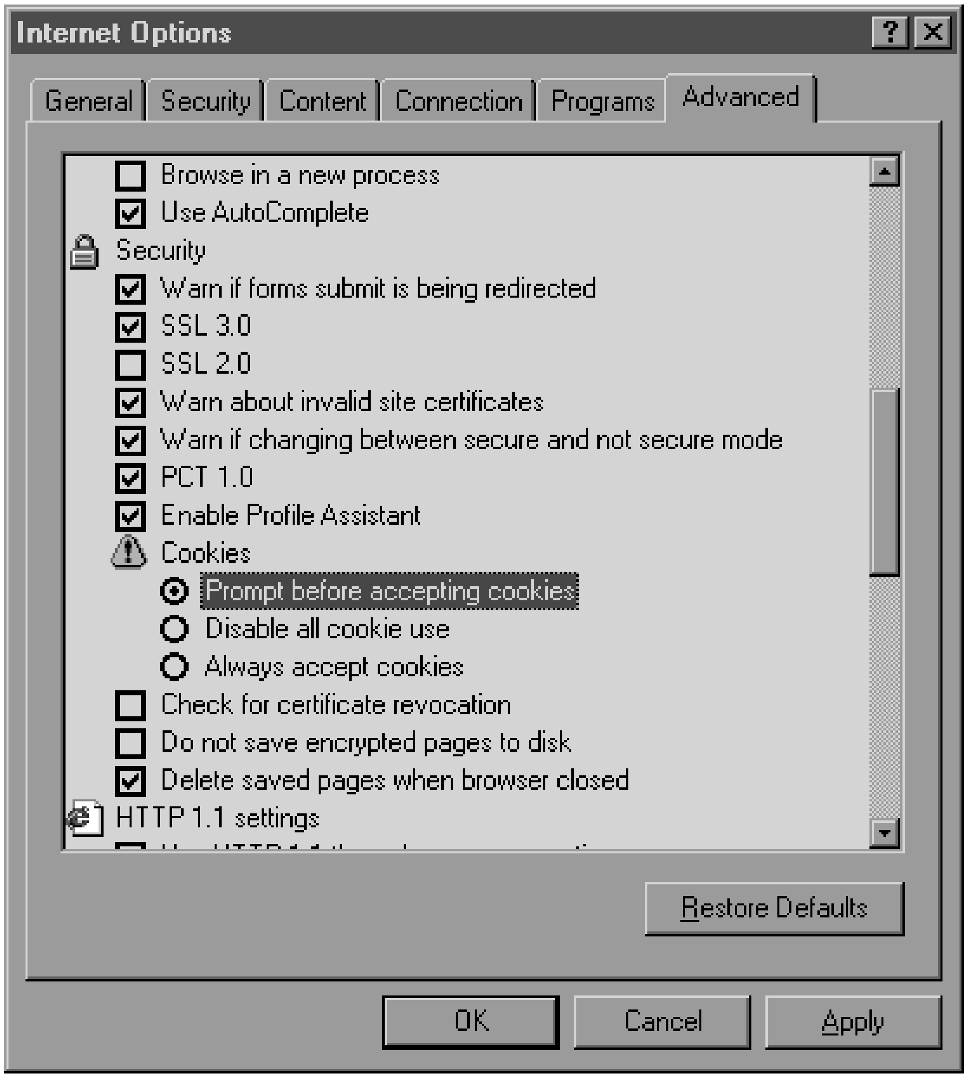
Authenticode(tm) Security Technology

Trusted Publishers and Issuers of Credentials

You have designated the following software publishers and credentials agencies as trustworthy. Windows applications can install and use software from these publishers or publishers certified by these agencies with out asking you first.

Figure 200. Certificates - Trusted Publishers for Authenticode

On the Advanced page of the Internet Options window are the settings for SSL versions, cookies, and other security Internet options. You can reach the Advanced page by pressing the **Advanced** tab on the top of the Internet Options window.



- [ ] Browse in a new process
- [x] Use AutoComplete

Security
- [x] Warn if forms submit is being redirected
- [x] SSL 3.0
- [ ] SSL 2.0
- [x] Warn about invalid site certificates
- [x] Warn if changing between secure and not secure mode
- [x] PCT 1.0
- [x] Enable Profile Assistant
- Cookies
  - ● Prompt before accepting cookies
  - ○ Disable all cookie use
  - ○ Always accept cookies
- [ ] Check for certificate revocation
- [ ] Do not save encrypted pages to disk
- [x] Delete saved pages when browser closed

HTTP 1.1 settings

Figure 201. Advanced Internet Options - Security Section

The security Internet options are explained in the following list:

- **Warn if forms submit is being redirected**

  If the form you are submitting is going to a different server from the one where the form was received, checking this box will set a warning for you.

- **SSL 3.0**

  This is the most recent version of SSL, with security enhancements over SSL 2.0 (see 3.1.1, “SSL Overview” on page 69).

- **SSL 2.0**

  This is the older version of SSL. It is here only for backward compatibility purposes.

- **Warn about invalid site certificates**

  If the URL listed in the site certificate is different from the URL the browser is attached to, checking this box will set a warning for you.

- **Warn if changing between secure and non-secure mode**

If you mark this check box, this warning will be given when you access or leave an SSL session.

- **PCT 1.0**

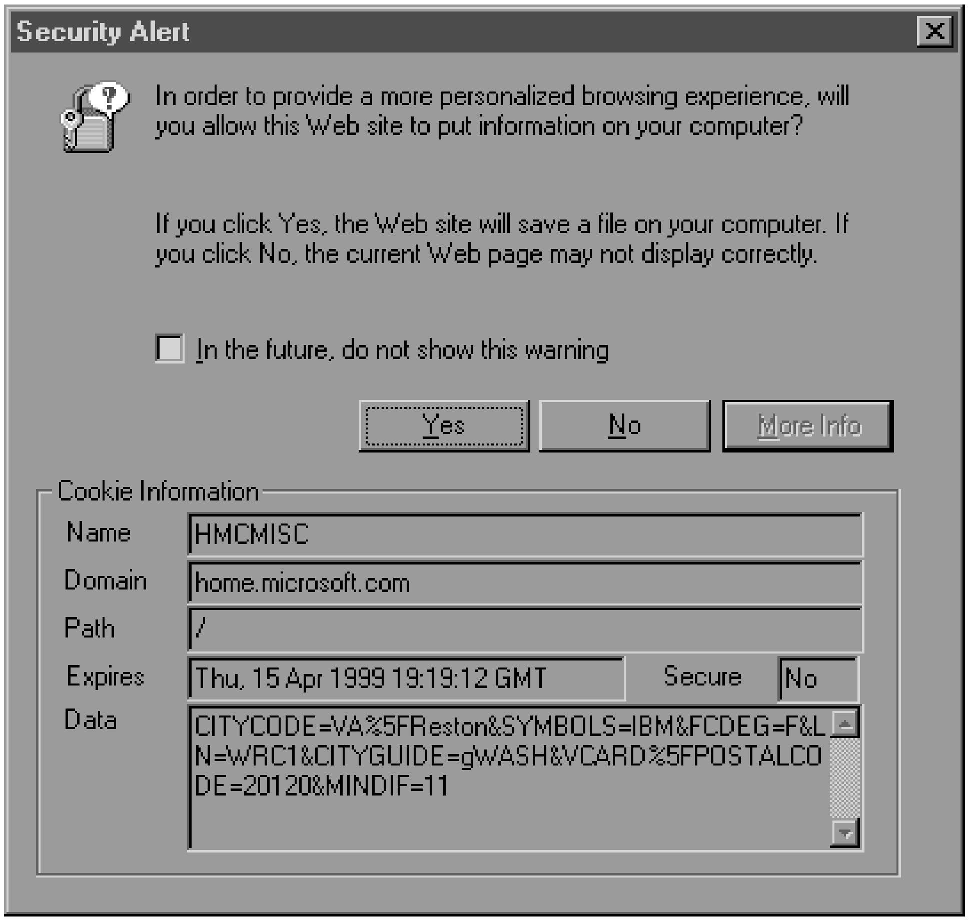
  Private Communications Technology (PCT) is a Microsoft proprietary security protocol similar in function to SSL. It was not widely accepted and is included here only for backward compatibility purposes.

- **Enable Profile Assistant**

  Profile Assistant will send your personal information that you entered by clicking on the **Edit Profile** button on the Content page, shown in Figure 198 on page 201. This information is sent in response to a request from a Web server, rather than requesting that you fill out a form. According to the Help file, the information will not be sent without your knowledge.

- **Cookies**

  We discussed cookies earlier in the Netscape section (see 5.1.1, "Netscape Navigator" on page 181) and we will talk again about them in 5.5, "Cookie Security Implications" on page 250. Figure 201 on page 205 lists the choices you get in MSIE. If you select to be prompted before accepting cookies and a server attempts to set a cookie on your system, you will see something similar to the following window:


Security Alert

In order to provide a more personalized browsing experience, will you allow this Web site to put information on your computer?

If you click Yes, the Web site will save a file on your computer. If you click No, the current Web page may not display correctly.

☐ In the future, do not show this warning

Cookie Information

Name	HMCMISC
Domain	home.microsoft.com
Path	/
Expires	Thu, 15 Apr 1999 19:19:12 GMT
Secure	No
Data	CITYCODE=VA%5FReston&SYMBOLS=IBM&FCDEG=F&LN=WRC1&CITYGUIDE=gWASH&VCARD%5FPOSTALCODE=20120&MINDIF=11

Figure 202. Cookie Set Warning

- **Check for certificate revocation**

If you check this item MSIE will check the Certificate Revocation List, managed by the CA of a certificate, before accepting the certificate as valid.

- **Do not save encrypted pages to disk**

  If you use MSIE from a server, or you share your PC, checking this item will prevent secured forms (now decrypted) from being saved to the hard disk where they could be seen by others.

- **Delete saved pages when browser closed**

  Checking this item will clear the page cache when the browser is closed. This is important, again, if you use MSIE from a server or if you share your PC. Checking this box will prevent confidential information that you could have gathered using your browser from being seen by other people.

5.1.3 Sun HotJava

Sun's HotJava browser is a Java application. The version we tested was HotJava 1.1.2. It can be run from any platform that supports the JDK 1.1. Sun provides specific launchers for the Windows platform so that users do not have to know the specifics of launching it from the JDK, and Sun also provides a Java Runtime Environment (JRE) with the browser for the Windows platforms so that the browser can be run without the JDK. Notice that since HotJava runs on JDK 1.1, it has a limited security model, that of JDK 1.1 (see 2.2, “The Evolution of the Java Security Model” on page 10).

The security features are configured in HotJava by clicking **Edit** on the menu bar, and then selecting **Preferences**, as seen in the picture below.

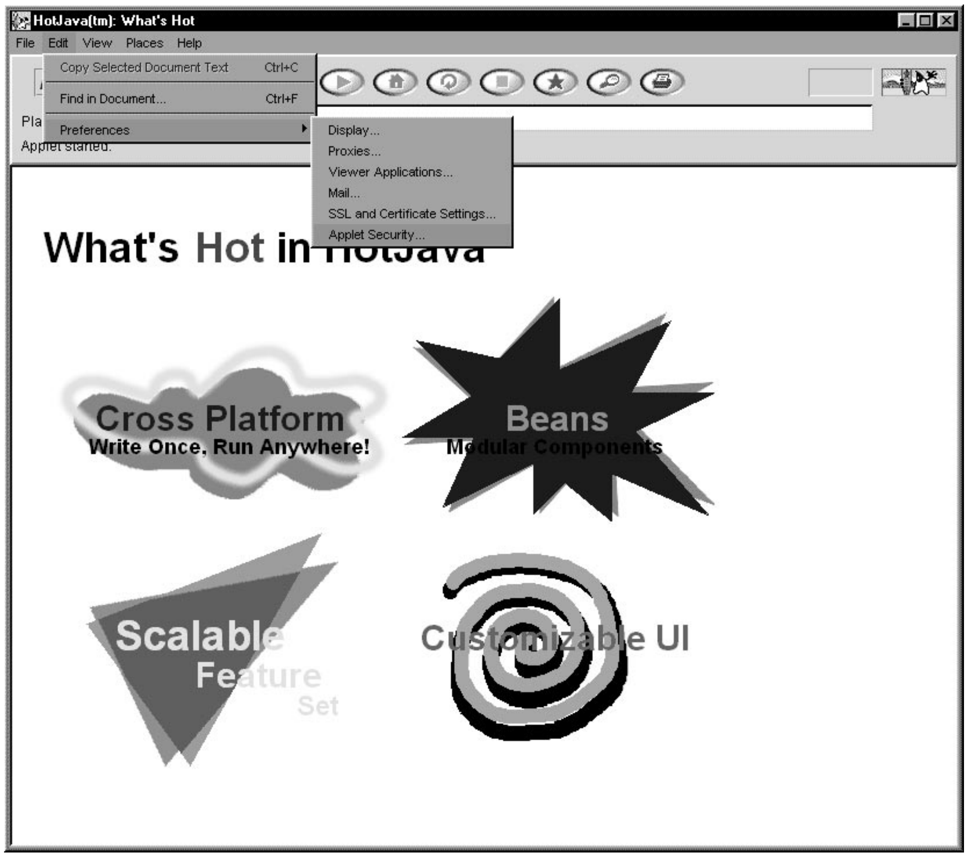
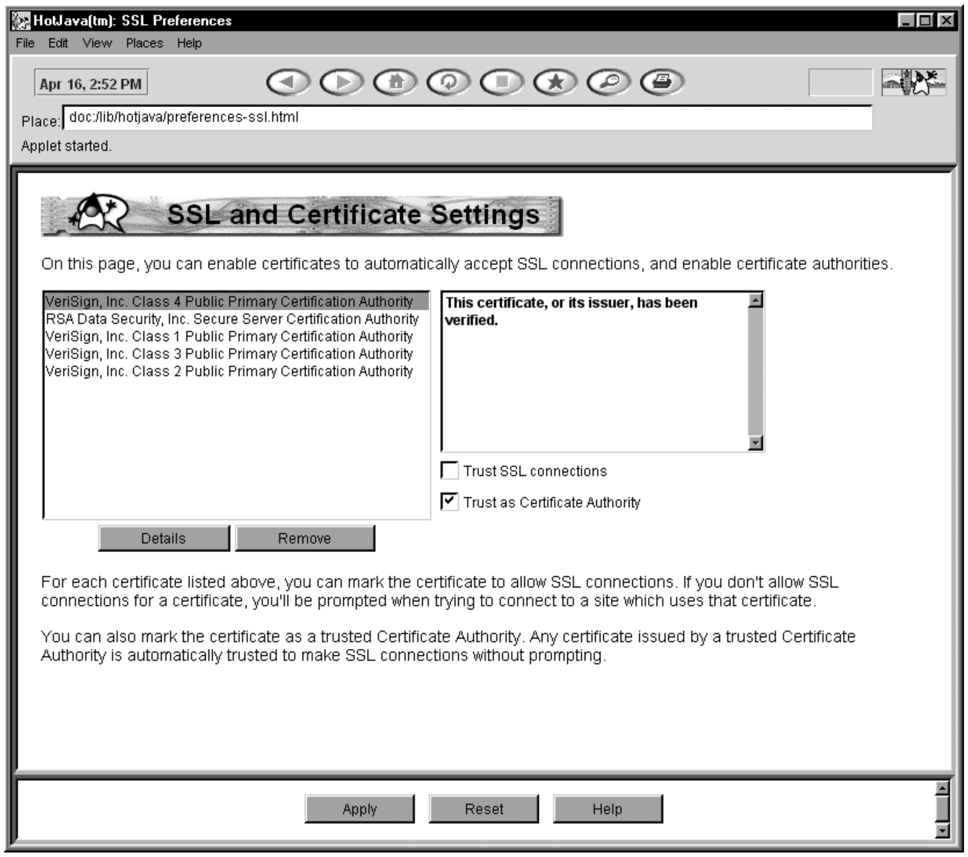


Figure 203. HotJava Browser - Edit - Preferences

The first security item on the Preferences menu is a page dealing with CA listings for server certificates as shown below. This page is named SSL Preferences.


HotJava(tm): SSL Preferences
File Edit View Places Help
Apr 16, 2:52 PM
Place: doc:/lib/hotjava/preferences-ssl.html
Applet started.
SSL and Certificate Settings

On this page, you can enable certificates to automatically accept SSL connections, and enable certificate authorities.

VeriSign, Inc. Class 4 Public Primary Certification Authority
RSA Data Security, Inc. Secure Server Certification Authority
VeriSign, Inc. Class 1 Public Primary Certification Authority
VeriSign, Inc. Class 3 Public Primary Certification Authority
VeriSign, Inc. Class 2 Public Primary Certification Authority

This certificate, or its issuer, has been verified.

☐ Trust SSL connections

☑ Trust as Certificate Authority

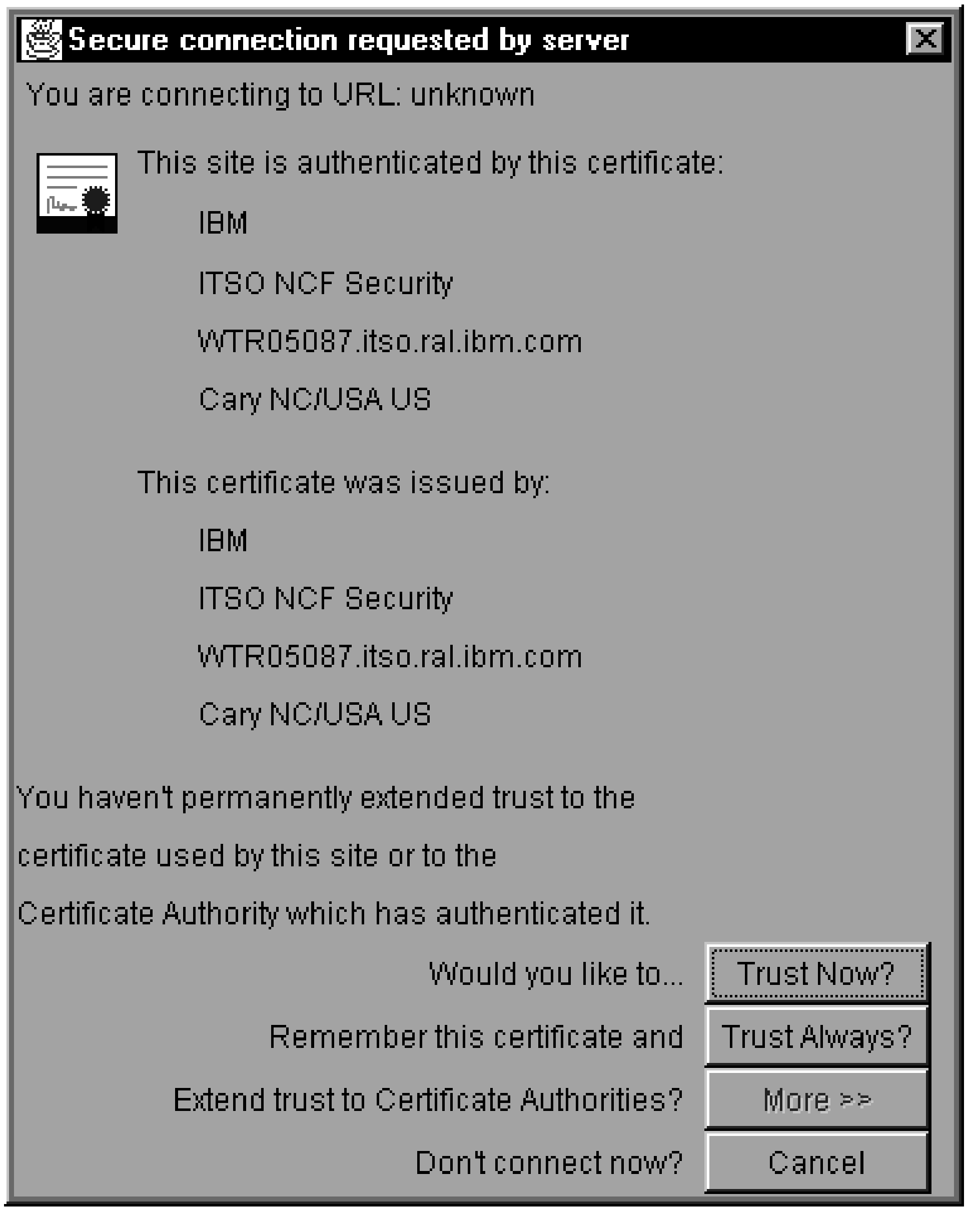
For each certificate listed above, you can mark the certificate to allow SSL connections. If you don't allow SSL connections for a certificate, you'll be prompted when trying to connect to a site which uses that certificate.

You can also mark the certificate as a trusted Certificate Authority. Any certificate issued by a trusted Certificate Authority is automatically trusted to make SSL connections without prompting.

Figure 204. HotJava Browser - SSL and Certificate Settings

With the **Trust as Certificate Authority** check box marked, incoming certificates for SSL connections signed by the CA specified are verified and the SSL session is connected. SSL connections from certificates signed by these CAs are trusted even though the **Trust SSL connections** check box is not marked.

Next, we show what happened when we tried to connect to our test Web server with an SSL connection, by typing `https://wtr05087` on the URL line of the browser. We received the warning shown in the next picture.



Secure connection requested by server

You are connecting to URL: unknown

This site is authenticated by this certificate:

IBM

ITSO NCF Security

WTR05087.itso.ral.ibm.com

Cary NC/USA US

This certificate was issued by:

IBM

ITSO NCF Security

WTR05087.itso.ral.ibm.com

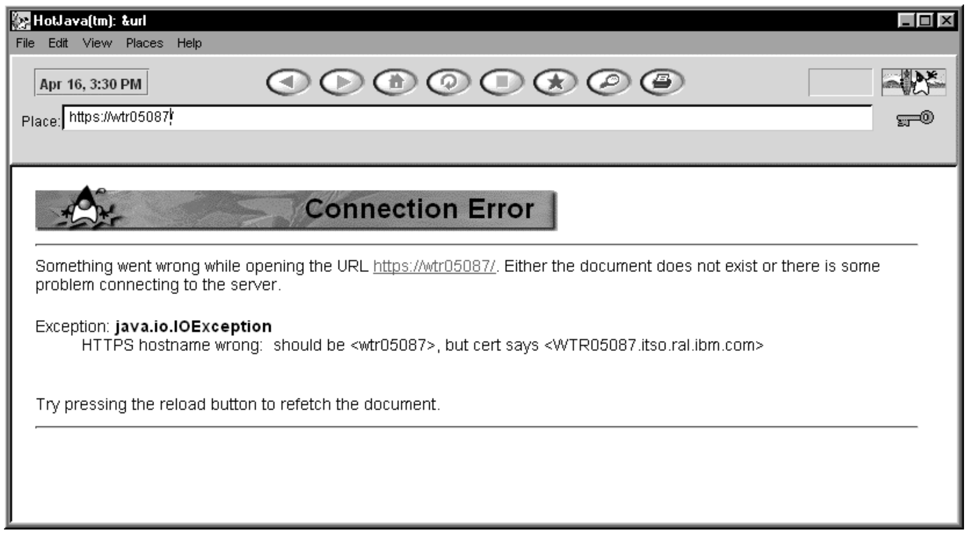
Cary NC/USA US

You haven't permanently extended trust to the certificate used by this site or to the Certificate Authority which has authenticated it.

Would you like to... Trust Now?
Remember this certificate and Trust Always?
Extend trust to Certificate Authorities? More >>
Don't connect now? Cancel

Figure 205. HotJava Browser - SSL Connection Request

As can be seen, we are given the opportunity to trust this self-signed certificate now or always, by pressing **Trust Now?** or **Trust Always?** respectively. This dialog appeared because the secure Web server we had invoked or the entity that signed its certificate were not listed in Figure 204 on page 209. Notice that both the entity who holds the certificate and the signer information are shown. You can also erase the connection by clicking on **Cancel**. To test what would happen, we clicked on **Trust Now?** the first time. We received a connection error shown below.



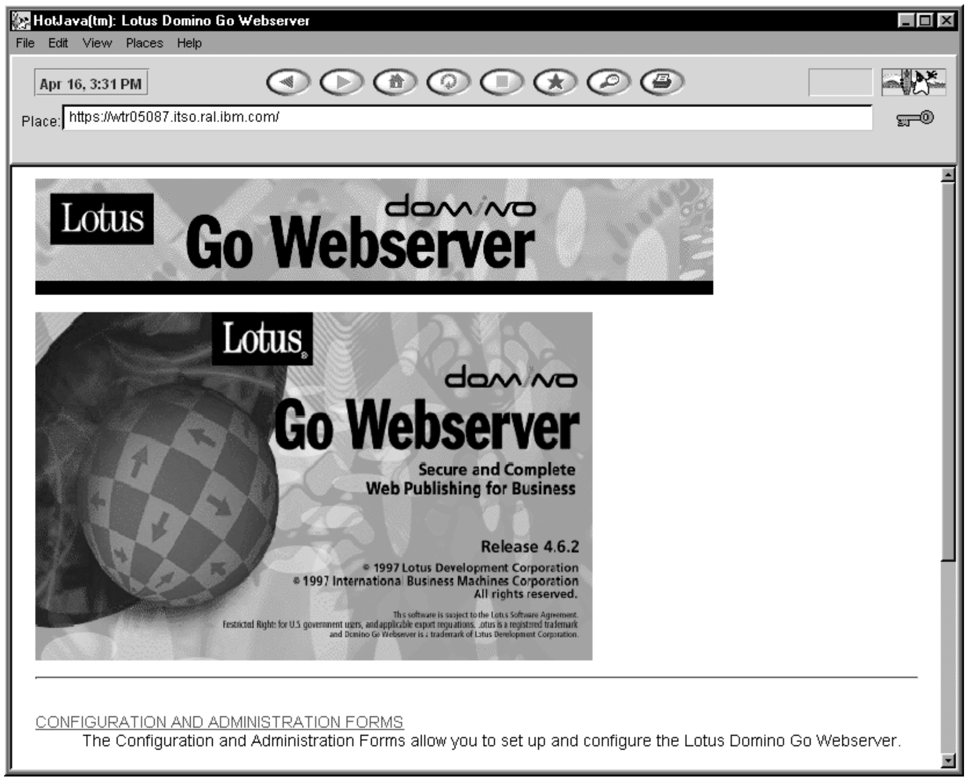
Something went wrong while opening the URL https://wtr05087/. Either the document does not exist or there is some problem connecting to the server.

Exception: **java.io.IOException**
HTTPS hostname wrong: should be <wtr05087>, but cert says <WTR05087.itso.ral.ibm.com>

Try pressing the reload button to refetch the document.

Figure 206. HotJava Browser - Connection Error

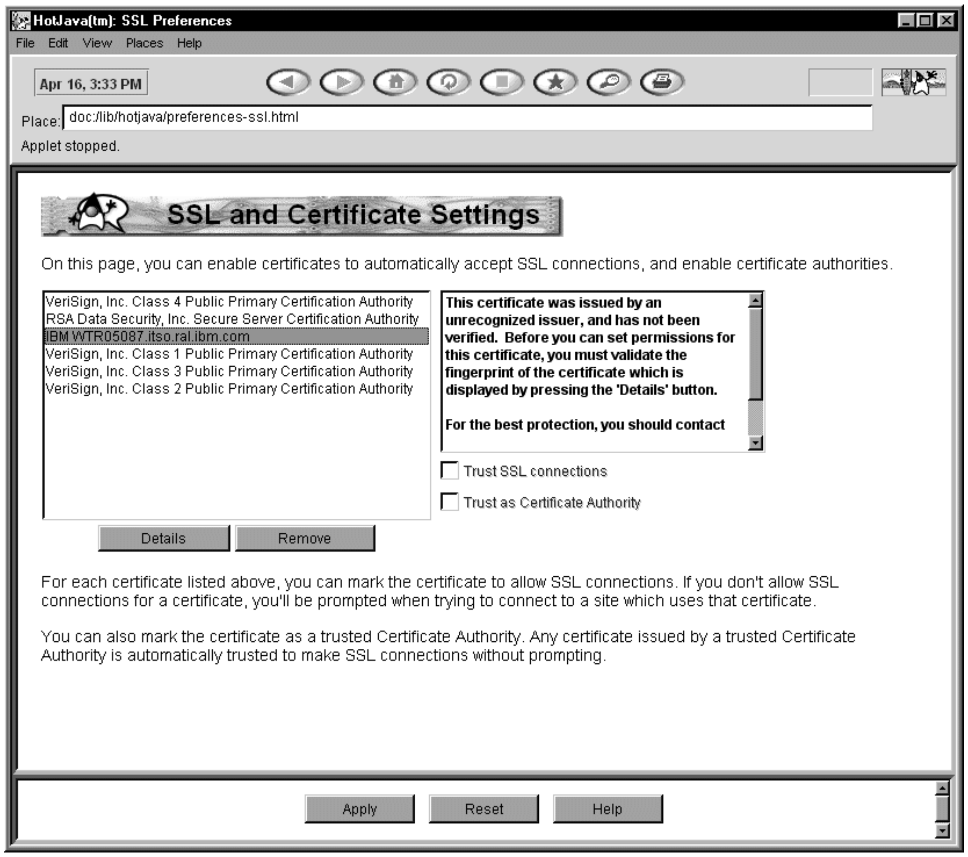
This connection error was generated because the host name used in the URL, `wtr05087`, did not match the fully qualified host name `WTR05087.itso.ral.ibm.com` supplied in the certificate. We changed the host name request in the URL as you can see in the next picture, and reloaded the page. Again we received the warning message shown in Figure 205 on page 210. This time we clicked on the **Trust Always?** button, and the connection was made as shown in the following picture.


HotJava(tm): Lotus Domino Go Webserver
File Edit View Places Help
Apr 16, 3:31 PM
Place: https://wtr05087.itso.ral.ibm.com/
Lotus
domino
Go Webserver
Lotus
domino
Go Webserver
Secure and Complete
Web Publishing for Business
Release 4.6.2
© 1997 Lotus Development Corporation
© 1997 International Business Machines Corporation
All rights reserved.
CONFIGURATION AND ADMINISTRATION FORMS

The Configuration and Administration Forms allow you to set up and configure the Lotus Domino Go Webserver.

Figure 207. HotJava Browser - SSL Connection Successful

Next, we returned to the SSL Preferences page to see that the certificate from our test server had been added to the list. Neither the **Trust SSL connections**, nor the **Trust as Certificate Authority** boxes could be checked.


HotJava(tm): SSL Preferences
File Edit View Places Help
Apr 16, 3:33 PM
Place: doc:/lib/hotjava/preferences-ssl.html
Applet stopped.
SSL and Certificate Settings
Details
Remove
Apply
Reset
Help

On this page, you can enable certificates to automatically accept SSL connections, and enable certificate authorities.

VeriSign, Inc. Class 4 Public Primary Certification Authority
RSA Data Security, Inc. Secure Server Certification Authority
IBM WTR05087.itso.ral.ibm.com
VeriSign, Inc. Class 1 Public Primary Certification Authority
VeriSign, Inc. Class 3 Public Primary Certification Authority
VeriSign, Inc. Class 2 Public Primary Certification Authority

This certificate was issued by an unrecognized issuer, and has not been verified. Before you can set permissions for this certificate, you must validate the fingerprint of the certificate which is displayed by pressing the 'Details' button.

For the best protection, you should contact

☐ Trust SSL connections

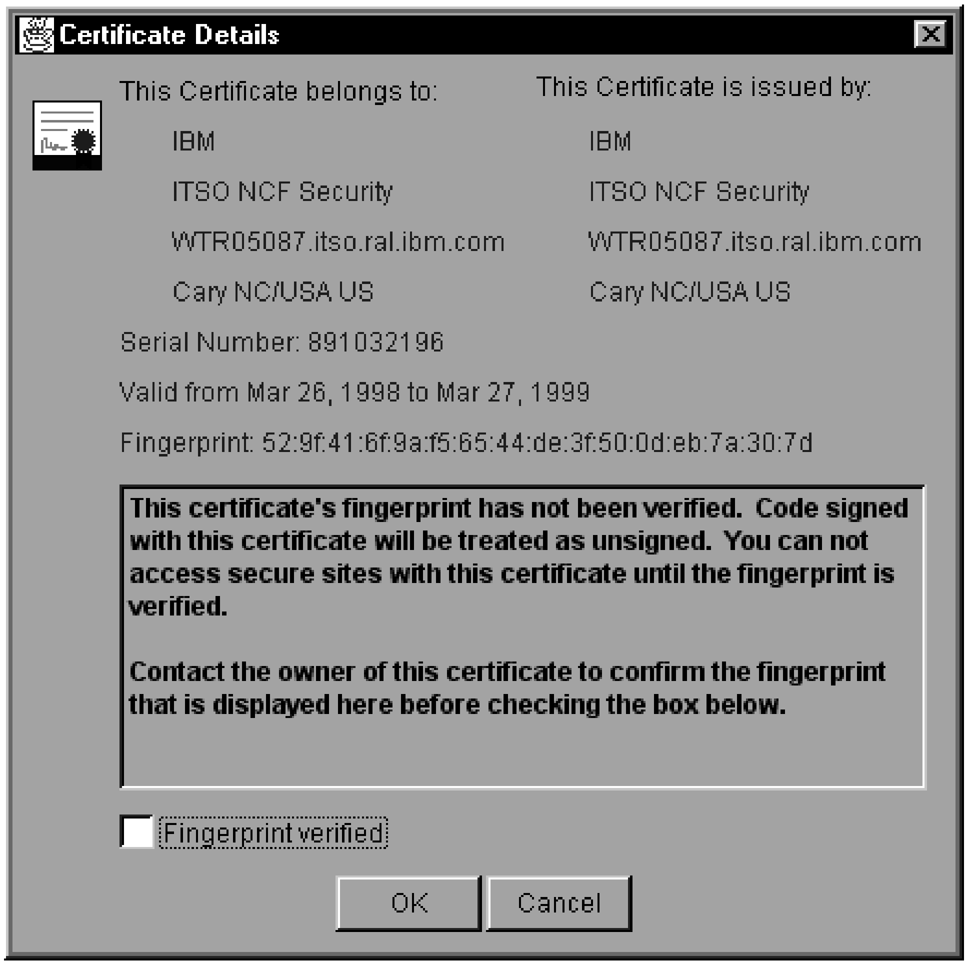
☐ Trust as Certificate Authority

For each certificate listed above, you can mark the certificate to allow SSL connections. If you don't allow SSL connections for a certificate, you'll be prompted when trying to connect to a site which uses that certificate.

You can also mark the certificate as a trusted Certificate Authority. Any certificate issued by a trusted Certificate Authority is automatically trusted to make SSL connections without prompting.

Figure 208. HotJava Browser - New Server Certificate Added

We were advised to verify the certificate, and change the details by clicking on the **Details** button below the list of CAs. The details for our test Web server certificate are shown in the next picture.



This Certificate belongs to:

IBM
ITSO NCF Security
WTR05087.itso.ral.ibm.com
Cary NC/USA US

This Certificate is issued by:

IBM
ITSO NCF Security
WTR05087.itso.ral.ibm.com
Cary NC/USA US

Serial Number: 891032196

Valid from Mar 26, 1998 to Mar 27, 1999

Fingerprint: 52:9f:41:6f:9a:f5:65:44:de:3f:50:0d:eb:7a:30:7d

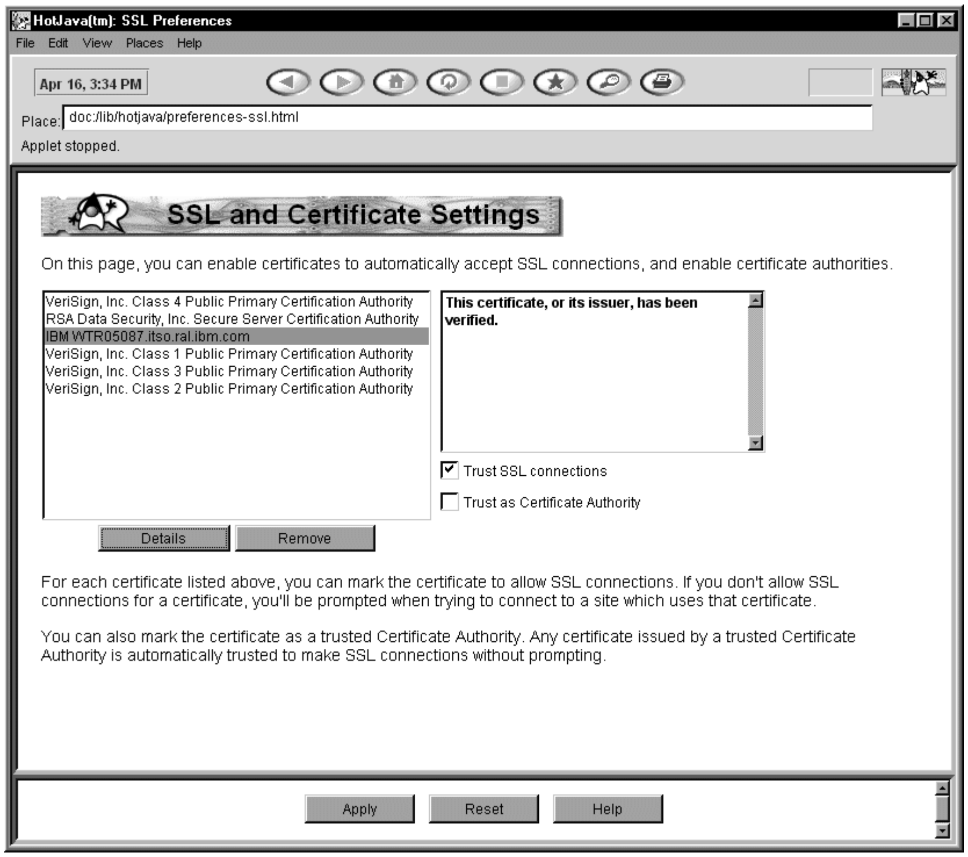
This certificate's fingerprint has not been verified. Code signed with this certificate will be treated as unsigned. You can not access secure sites with this certificate until the fingerprint is verified.

Contact the owner of this certificate to confirm the fingerprint that is displayed here before checking the box below.

☐ Fingerprint verified

Figure 209. HotJava Browser - Certificate Details

There is no program to verify the fingerprint of the certificate. You are expected to contact the owner of the certificate and verify the certificate by some other means. When the fingerprint of the certificate is verified, the box **Fingerprint verified** at the bottom can be checked. Once the fingerprint is marked as verified, the certificate can be marked as trusted for SSL connections, by clicking **Trust SSL connections**. This is shown below.



doc:/lib/hotjava/preferences-ssl.html

Applet stopped.

SSL and Certificate Settings

On this page, you can enable certificates to automatically accept SSL connections, and enable certificate authorities.

VeriSign, Inc. Class 4 Public Primary Certification Authority
RSA Data Security, Inc. Secure Server Certification Authority
IBM WTR05087.itso.ral.ibm.com
VeriSign, Inc. Class 1 Public Primary Certification Authority
VeriSign, Inc. Class 3 Public Primary Certification Authority
VeriSign, Inc. Class 2 Public Primary Certification Authority

This certificate, or its issuer, has been verified.

[✓] Trust SSL connections

[ ] Trust as Certificate Authority

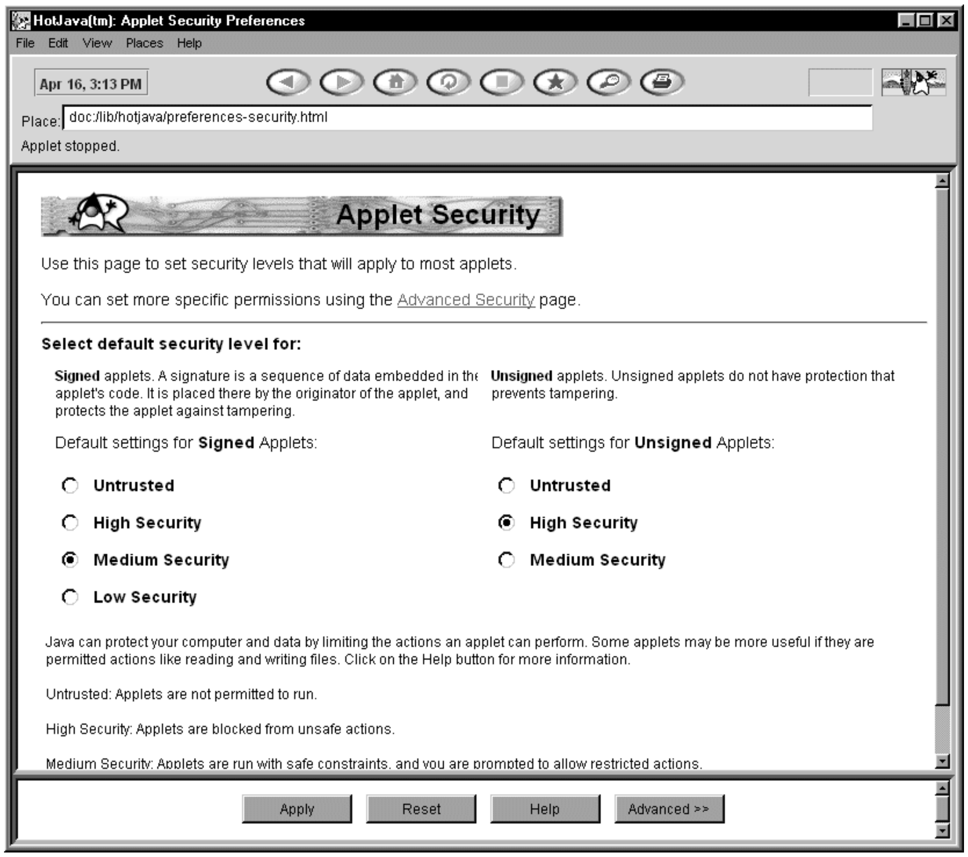
For each certificate listed above, you can mark the certificate to allow SSL connections. If you don't allow SSL connections for a certificate, you'll be prompted when trying to connect to a site which uses that certificate.

You can also mark the certificate as a trusted Certificate Authority. Any certificate issued by a trusted Certificate Authority is automatically trusted to make SSL connections without prompting.

Figure 210. HotJava Browser - New Server Certificate Trusted

The next time an SSL connection is made to this server, there is no warning because the server is trusted.

Next we will look at the Applet Security Preferences window. We will notice that the Sun HotJava browser provides the user with a more extensive capability to configure applet security than Netscape Navigator and Microsoft Internet Explorer. The Applet Security Preferences window is accessed from the Edit pull-down menu as shown in Figure 203 on page 208. The next picture shows the basic Applet Security Preferences page.


HotJava(tm): Applet Security Preferences
File Edit View Places Help
Apr 16, 3:13 PM
Place: doc:/lib/hotjava/preferences-security.html
Applet stopped.
Applet Security
Apply
Reset
Help
Advanced >>

Use this page to set security levels that will apply to most applets.

You can set more specific permissions using the Advanced Security page.

Select default security level for:

Signed applets. A signature is a sequence of data embedded in the applet's code. It is placed there by the originator of the applet, and protects the applet against tampering.

Default settings for **Signed** Applets:

- Untrusted
- High Security
- [✓ Medium Security]
- Low Security

Unsigned applets. Unsigned applets do not have protection that prevents tampering.

Default settings for **Unsigned** Applets:

- Untrusted
- [✓ High Security]
- Medium Security

Java can protect your computer and data by limiting the actions an applet can perform. Some applets may be more useful if they are permitted actions like reading and writing files. Click on the Help button for more information.

Untrusted: Applets are not permitted to run.

High Security: Applets are blocked from unsafe actions.

Medium Security: Applets are run with safe constraints, and you are prompted to allow restricted actions.

Figure 211. HotJava Browser - Basic Applet Security

On this page, you are allowed to set security levels for signed and unsigned applets.

- **Untrusted**

  Applets are not permitted to run.

- **High Security**

  Applets are blocked from unsafe actions.

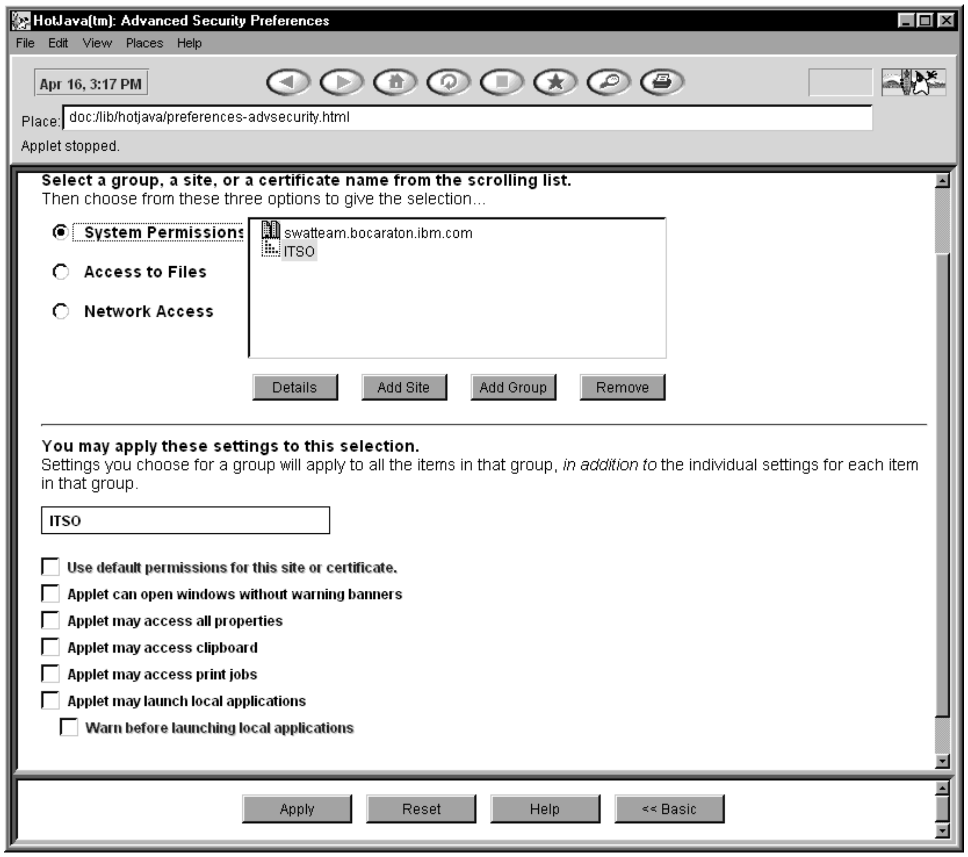
- **Medium Security**

  Applets will run with safe constraints and you are prompted for actions which might otherwise be restricted.

- **Low Security**

  Applets are given the most freedom on the system with a low security setting. You are not prompted for any applet actions. This option can be selected only for signed applets.

The next picture shows the Advanced Security Preferences window. You get to this page by clicking on the **Advanced** button at the bottom of the basic page.



doc:/lib/hotjava/preferences-advsecurity.html

Applet stopped.

Select a group, a site, or a certificate name from the scrolling list.
Then choose from these three options to give the selection...

[✓ System Permissions]
Access to Files
Network Access

swatteam.bocaraton.ibm.com
ITSO

You may apply these settings to this selection.
Settings you choose for a group will apply to all the items in that group, *in addition to* the individual settings for each item in that group.

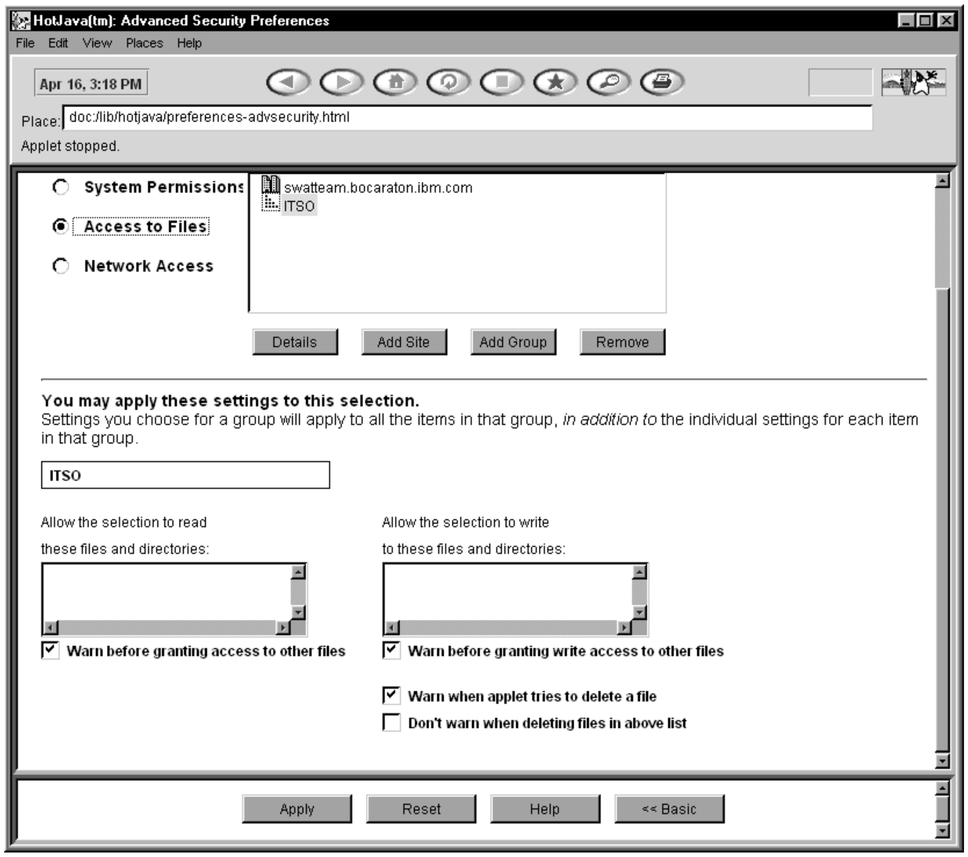
ITSO

- [ ] Use default permissions for this site or certificate.
- [ ] Applet can open windows without warning banners
- [ ] Applet may access all properties
- [ ] Applet may access clipboard
- [ ] Applet may access print jobs
- [ ] Applet may launch local applications
  - [ ] Warn before launching local applications

Figure 212. HotJava Browser - Advanced Security Preferences

Here you are able to set **System Permissions**, **Access to Files** and **Network Access** for applets from specific host names or from groups. In Figure 212 we have added a host name and a group to be able to see the security settings.

You can see the types of system permissions that you can grant to applets for access to the system in Figure 212. Next are the choices for file and directory access.


HotJava(tm): Advanced Security Preferences

File Edit View Places Help

Apr 16, 3:18 PM

Place: doc:/lib/hotjava/preferences-advsecurity.html

Applet stopped.

System Permissions

(●) Access to Files

Network Access

swatteam.bocaraton.ibm.com
ITSO

Details | Add Site | Add Group | Remove

You may apply these settings to this selection.
Settings you choose for a group will apply to all the items in that group, *in addition to* the individual settings for each item in that group.

ITSO

Allow the selection to read these files and directories:

Allow the selection to write to these files and directories:

[✓] Warn before granting access to other files

[✓] Warn before granting write access to other files

[✓] Warn when applet tries to delete a file

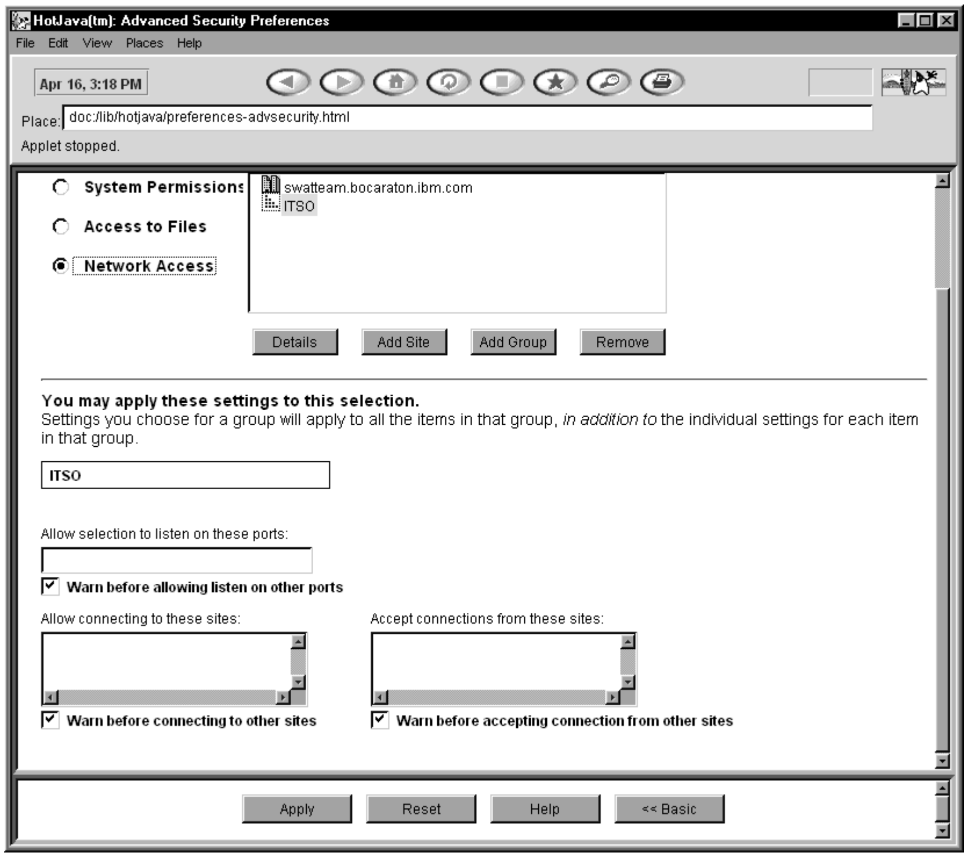
[ ] Don't warn when deleting files in above list

Apply | Reset | Help | << Basic

Figure 213. HotJava Browser - Applet File Control

These settings allow you to specify what files and directories applets from the listed groups or servers can access. File access can be read and write. A particular case of write access is delete. You might want to select **Warn when applet tries to delete a file**, in order to be informed before a file is removed by an applet.

In the next picture we see permissions which may be given applets for network access.


HotJava(tm): Advanced Security Preferences

Place: doc:/lib/hotjava/preferences-advsecurity.html

Applet stopped.

System Permissions
Access to Files
[✓ Network Access]

swatteam.bocaraton.ibm.com
ITSO

You may apply these settings to this selection.
Settings you choose for a group will apply to all the items in that group, *in addition to* the individual settings for each item in that group.

ITSO

Allow selection to listen on these ports:

[✓] Warn before allowing listen on other ports

Allow connecting to these sites:

Accept connections from these sites:

[✓] Warn before connecting to other sites

[✓] Warn before accepting connection from other sites

Figure 214. HotJava Browser - Network Access Control

Allowing an applet to listen to a port allows it to accept an incoming connection. Applets can also be given permission to connect to specific sites, and to accept connections from specific sites. You can also see that warnings are set by default for requests to connect to incoming and outgoing sessions.

5.2 The Java Applet Programming Model for NCF Developers

The Java sandbox defines a set of security restrictions within which all untrusted code runs. Untrusted code that attempts operations outside of these constraints causes a SecurityException to be thrown. Therefore, any code requiring extra-sandbox privileges will need to be trusted.

Trust can be granted to code by various means, usually by being in a signed archive (JAR for Netscape Communicator, Sun HotJava and base JDK platforms; CAB for Microsoft Internet Explorer) or by having the classes be in the system classpath. In Internet Explorer 4.0 and Navigator 4.0, although privileges can be granted to code in signed archives, they are not always enabled. In Navigator, all extra-sandbox permissions must be explicitly enabled when needed, by calling a vendor-specific API. This is also true in Internet Explorer for calls from untrusted sources. This includes calls from script, as well as an applet's start(), stop(), init() and destroy() methods.

In order to enable already granted privileges, you will need to make calls to the Microsoft and Netscape security APIs. This section will help you identify when you need to make those changes, how to make them and how to avoid some security hazards that such changes might inadvertently introduce into your code.

5.2.1 Internet Explorer's Security Model

In Microsoft Internet Explorer 4.0, applets are by default untrusted and have only sandbox permissions. To get out of the sandbox, the classes need to be in a signed CAB or be on the classpath. A CAB can be signed with specific requested permissions, such as file I/O or printing, or one of three sets of permissions, called high, medium and low.

When the HTML file is parsed and the applet is downloaded, its digital signature is examined and, depending on which Security Zone your applet belongs to, and how the security configuration is set for the browser, the applet is either automatically granted these requested permissions, granted/denied based on user acknowledgment, or automatically denied.

Note that these permissions are granted, but not necessarily enabled (ready to use). When an attempt is made to acquire a guarded resource, it is enabled, if a stack walk shows that all classes on the stack have also been granted that particular permission and none has revoked that permission, or if the privilege has been specifically asserted.

This is to prevent less-trusted code from calling more-trusted code and in doing so gain extra privileges, preventing the so-called *luring attack*.

Permissions that have been granted can also be explicitly enabled by calling a method in a class that ships with Internet Explorer 4.0:

```
com.ms.security.PolicyEngine.assertPermission(com.ms.security.PermissionID.MSIE_TARGET);
```

where `MSIE_TARGET` is one of several predefined values, like `EXEC`, `FILEIO`, `NETIO`, `PRINTING`, `MULTIMEDIA`, etc. The complete list is at http://www.microsoft.com/java/sdk/20/packages/security/permissions/default.htm.

When a permission is enabled in this way, access to the guarded resource is permitted in the thread of the calling function, until the function returns.

One important twist to this is that the applet's start(), stop(), init() and destroy() methods are all considered untrusted, since they are called by the browser. So, if you need extra-sandbox permissions in these functions, you will need to call PolicyEngine.assertPermission() to enable them.

5.2.2 Netscape Navigator's Security Model

Similar to Internet Explorer 4.0, Netscape Navigator 4.0 runs unsigned (untrusted) classes in the sandbox. Although being in a signed JAR file can grant the code additional privileges, none are automatically enabled. Netscape requires that the applet in all cases explicitly requests extra-sandbox privileges. For example:

```
netscape.security.PrivilegeManager.enablePrivilege("NN_Target");
```

The first time the above code is executed in a session, a dialog will surface prompting the user to approve or deny the access. The privilege remains enabled for the calling thread until the function that requested the permission returns. Subsequent calls to enablePrivilege() for the same target will be automatically approved or denied (no dialog) based on the user's response to the initial dialog.

In the above method call, `NN_Target` represents a Netscape-defined System target. The complete list of all the Netscape-defined System targets can be found at http://developer.netscape.com/library/documentation/signedobj/targets/index.htm.

5.2.3 In Summary

To escape the sandbox, you need to be on the classpath or in a signed archive. In addition, Netscape requires you to explicitly enable any privileges you need before you can exercise them. With Internet Explorer this is true only in cases where an untrusted class is on the call stack, such as calls from script and calls from the browser (start(), stop(), init(), destroy()).

The following lines of code show what you should do to run your applet under Netscape Navigator and Microsoft Internet Explorer, to exercise all functionality and throw no SecurityException exceptions.

```
try
{
  com.ms.security.PolicyEngine.assertPermission(com.ms.security.PermissionID.MSIE_TARGET);
  netscape.security.PrivilegeManager.enablePrivilege("NN_Target");

  ...  // try to acquire the resource
}

catch(SecurityException e)
{
  ...  // deny access
}

finally
{
  netscape.security.PrivilegeManager.disablePrivilege("NN_Target");
  com.ms.security.PolicyEngine.denyPermission(com.ms.security.PermissionID.MSIE_TARGET);
}
```

Figure 215. How to Enable and Disable Privileges in Navigator and Internet Explorer

Remember, as a security guideline, that you need to clean up all the privileges you enabled. Don't have a path that returns from your function and leaves a privilege enabled. Always match a call to enable with a call to disable, as shown in Figure 215.

Moreover, you should never ask for all permissions if you only need access to a single file, or assert permission randomly in the code, *just to be safe*.

Other security guidelines to get the applets to work in a trusted environment will be shown in the next section.

5.2.4 Avoiding Security Hazards

Security hazards can be broken into four categories:

1. Integrity attacks, such as unauthorized modifications to files and threads
2. Disclosure attacks, such as revealing ordinarily private information or files

3. Denial of service attacks, such as crashing or hanging our applets, the JVM, the underlying operating systems or the server
4. Annoyance attacks, such as displaying unwanted files or other mischievous behavior

We will show now how it is possible to ensure that our applets avoid the more common and more dangerous security hazards. This means that we will concentrate our efforts on shallow integrity and disclosure attacks.

An attack is likely to come through the front door, meaning through public methods on your applet's class (and public static methods on any public class). At minimum we need to examine all such methods and classify them in one of the following four ways:

1. The method does not need to be public, so it could be changed to private, protected or default access.
2. The method is absolutely safe or, in other words, it does not use directly or indirectly any extra-sandbox privileges and does not reveal private data.
3. The method is reasonably safe, so that it uses extra-sandbox privileges in a limited fashion, constrained so as to be, in practice, safe.
4. The method allows straightforward malicious use.

You can use the `javap` command line utility from the JDK, with the option `-public`, to generate a list of all public methods on your applet's class. If your applet is like most, you have too many public methods. You can probably save yourself a lot of work by simply changing some of the methods to private, protected or default access.

For example a method like this is very dangerous:

```
public void deleteFile(String path)
{
  try
  {
    com.ms.security.PolicyEngine.assertPermission(com.ms.security.PermissionID.FILEIO);
    netscape.security.PrivilegeManager.enablePrivilege("UniversalFileAccess");

    File file = new File(path)
    file.delete();
  }

  catch(SecurityException e)
  {
    e.printStackTrace();
  }

  finally
  {
    netscape.security.PrivilegeManager.disablePrivilege("UniversalFileAccess");
    com.ms.security.PolicyEngine.denyPermission.(com.ms.security.PermissionID.FILEIO);
  }
}
```

Figure 216. A Dangerous Public Method

Because the deleteFile() method is both public and enables its own privileges, it can be called from untrusted malicious code, such as a script or another Java applet on the page to get permissions it itself did not have. Because the function is so general and powerful, it is especially dangerous. Better is to make the deleteFile() method private or move the privilege assertions to a private function which then calls a public deleteFile().

In case you are developing a Java bean (see the IBM redbook *Network Computing Framework Component Guide*, SG24-2119), the BeanInfo interface is used to list all the methods that your bean advertises to a builder, so these methods need to be public. Otherwise, you may be able to reduce your exposure considerably by constraining access for the remaining methods. Next, take a look at the remaining public methods, identify those methods that can run in the sandbox and only call methods that run in the sandbox. These may be considered absolutely safe, provided these additional conditions are met:

- The method should not return sensitive data, such as the user's name, IP address or any other personal information.
- The method should not return an object, unless all of its public methods are also absolutely safe.
- The methods should only trust its input parameters, since they may be tainted. For example if

  ```
  setFileName(String name)
  ```

  is a public method that sets an internal file name variable, but that variable is used at shutdown as the name to save the file to, then the public method:

  ```
  setFileName(String name)
  ```

  is equivalent to:

  ```
  saveFile(String name)
  ```

  and should be treated as such.

Methods that have been deemed absolutely safe require no further treatment. For the remaining public methods, you will need to make a judgement call and estimate what would be the overall security risk of allowing unrestricted access to this function from untrusted users. A security guideline could be the following: a method is reasonably safe if it uses extra-sandbox permissions, but it can do nothing malicious, regardless of the state of the applet as configured through input parameters to that method, or via other public methods.

It is generally believed that a function like this:

```
boolean loadDocument(String name)
```

is reasonably safe, if all it does is make an URL from `name` and load it into the applet's workspace. There is a slight privacy concern, since a malicious applet could use it to look at your file system to see whether a file of a given name exists, but even that could be acceptable. Other safe variations on loadDocument() are:

- `boolean loadDocument()`

  This variation is reasonably safe, if it brings up a file dialog, so the user has to choose the file.

- `boolean loadDocument(InputStream in)`

This variation is absolutely safe, since the security check occurs when the stream is opened.

A variation on the loadDocument() method that is not reasonably safe is the following:

```
InputStream loadDocument(String name)
```

A malicious program could use this method to send data back to the server.

Once you have identified the reasonably safe methods, you have two tasks:

1. Make calls to the browser's security APIs to enable privileges for the method (see Figure 215 on page 220).
2. Add a line to the `javadoc` comment for the method to indicate what privileges the method is assuming. For example:

```
/**
 *  Loads a document into the workplace
 *
 *  @param name the URL of the document to open
 *  @return true if success, false if failure
 *
 *  @exception SecurityException if you do not have permissions to read from the file
 */

public boolean loadDocument(String name)
```

Figure 217. Describing the Method with in a javadoc Comment

The idea is this: if we have a signed applet with all the privileges, we want to enable these privileges for all the reasonably safe scripting methods. However, if the applet is untrusted, we will throw a SecurityException.

Once the reasonably safe methods have been taken care of, we move onto those remaining methods, the potentially dangerous ones. For these, we will need to document them in the `javadoc` comment, as described in Figure 217, but we will not make calls to enable the privileges. Instead, we will require that the caller of these methods does the enabling. Any method that could be used maliciously in a rather transparent manner should be treated in this way. For example, as we have already discussed, a setFileName() method that takes the name of a file and saves to it is dangerous and falls in this category.

5.2.5 How to Test

You will want to test that, in the process of making these changes, you haven't broken the full functionality of the applet. The following should be verified at this point:

- In a trusted environment, all functionality should work from the user interface with no SecurityException exceptions.
- In an untrusted environment, all script methods documented as absolutely safe should work.
- In a trusted environment, all script methods documented as absolutely or reasonably safe should work.

- In a trusted environment, all script methods, including the dangerous ones, should work when called from another trusted (signed) applet on the page.

You also need to define exactly what, if anything, your applet does in an untrusted environment. Once you have a functionality definition, then you will need to verify that your applet can provide this functionality and prevent or handle user interface requests for functionality that cannot be done in the sandbox.

The first step is to verify that all attempts to acquire a guarded resource are done in a `try{}` block, with SecurityException being caught and handled (see Figure 215 on page 220). The rest is user interface work, targeted at how well you communicate to the user, through the user interface, what functionality is available in an untrusted environment.

5.2.6 Code Signing on Different Platforms

We want to conclude this section mentioning another problem that Java developers must consider, in terms of code portability. The developer of a Java applet who wants to use the trusted Java applet security model must be aware that signature formats vary from platform to platform.

If the signed applet needs to run on Netscape's JVM implementation, then the applet developer should put the code in a JAR file (see 2.5.1, “The jar Utility” on page 29), signed using a VeriSign Certificate for Object Signing. The signature format for Netscape is RSA.

If the applet is to run on a HotJava-based environment, the code must be put in a JAR file as well, but it is signed using a self-signed certificate created by the `javakey` tool. The signature format in this case would be DSA and not RSA. Similar considerations apply for base JDK 1.1 and 1.2, unless you add the providers to support RSA (see our discussion in 2.5.2, “The keytool Utility” on page 29 and 4.1.2, “Users” on page 128).

If the platform where the applet will run is Microsoft Internet Explorer, the code is put in a CAB file for Internet Explorer and it is signed using a VeriSign Certificate for Authenticode. Note that this certificate is different from the one used with Netscape. The signature format in this case is RSA.

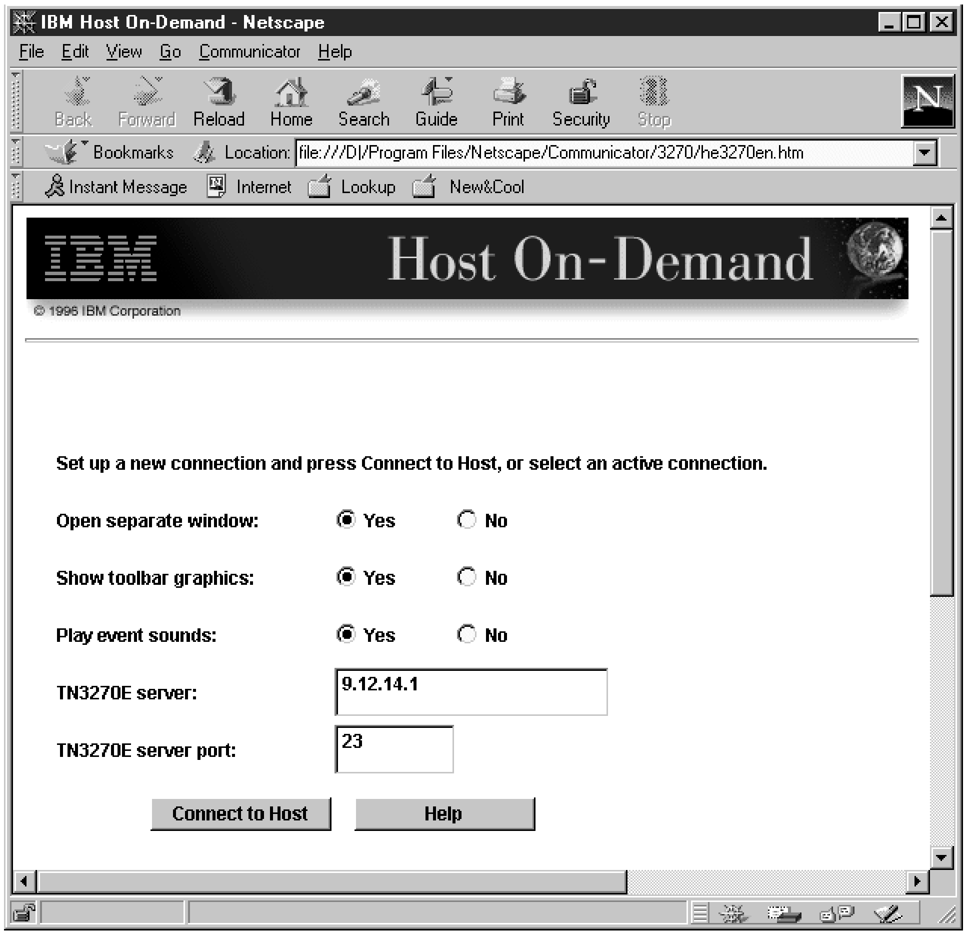
These differences generate several problems for applet developers who want to write code that is portable across the platforms. They often have to deploy three different versions of any page hosting a trusted (or even untrusted) Java applet.

5.3 Interacting with Signed Java Applets

What we will say in this section is strictly related to Java 1.1, since at the time this book went to print JDK 1.2 was still in beta and no Web browser supported the new version yet. Detailed information on the Java 1.2 security model, even if based on the beta version we worked on, can be found in Chapter 2, “The New Java 1.2 Security Model” on page 9.

We have shown in Figure 179 on page 182 that Navigator offers a simple ability to enable or disable Java and JavaScript. Similar settings are present in Microsoft Internet Explorer too. In the JDK 1.1 security model (see 2.2, “The Evolution of the Java Security Model” on page 10), unsigned applets can only run in a sandbox and cannot perform most file system access or file I/O routines. There are times, however, when you might want to grant a Java applet more access to your system. For instance you would like a specific Java applet to be granted the ability to update certain files stored in the hard disk of your computer. For security precautions, Netscape Communicator, Microsoft Internet Explorer and Sun HotJava support the trusted Java applet security model, which is based on JDK 1.1. A signed applet is one that will not be downloaded to your system unless you approve a security digital certificate that identifies the sender of that applet. Similarly, signed JavaScript files will not be executed until you approve the sender's digital certificate. The digital signature process is used to show integrity and ownership of the signed code when permission is requested. Moreover, if a Java applet or a JavaScript file wants to perform some particular function that could be considered risky, since it could compromise the security of your system, you will be asked to approve this action too. A special security alert asks you to approve a digital certificate and also to permit a specific type of access to your system.

The end user who executes the applet is given the chance to grant or deny the permissions needed for the applet to carry out its tasks. This permission can be granted on a one-time basis, or the JVM can remember the permissions on a permanent basis. An example of use on a permanent basis might be IBM Host On-Demand, the telnet 3270 client included with Netscape Communicator.


IBM Host On-Demand - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks
Location: file:///D|/Program Files/Netscape/Communicator/3270/he3270en.htm
Instant Message Internet Lookup New&Cool
IBM
Host On-Demand
© 1996 IBM Corporation
Connect to Host
Help

Set up a new connection and press Connect to Host, or select an active connection.

Open separate window: [✓ Yes] No

Show toolbar graphics: [✓ Yes] No

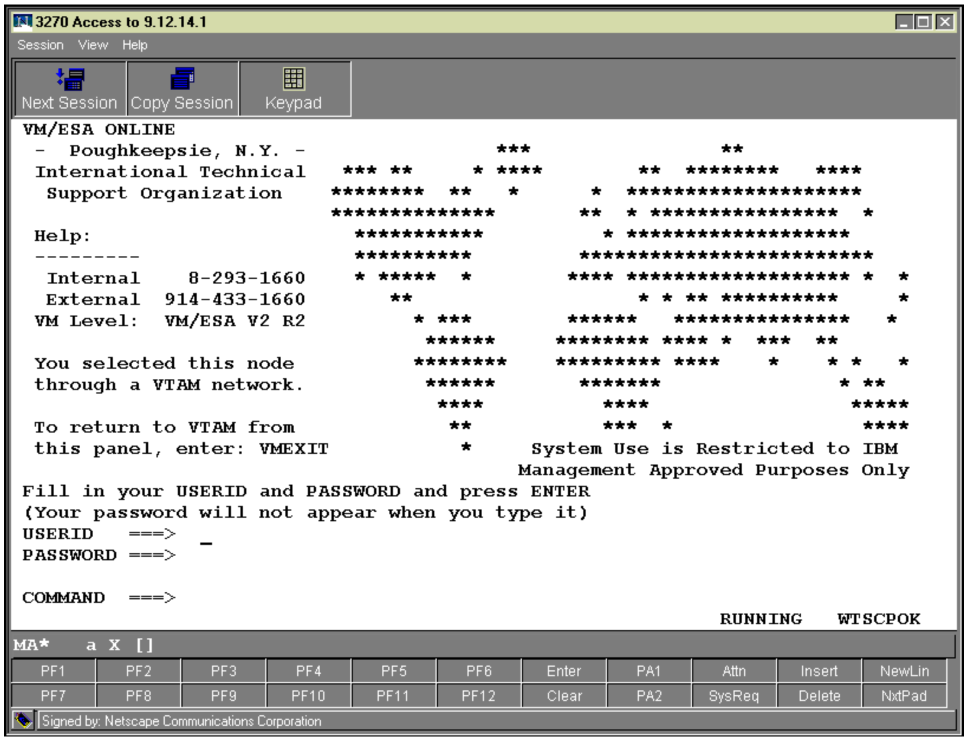
Play event sounds: [✓ Yes] No

TN3270E server: 9.12.14.1

TN3270E server port: 23

Figure 218. IBM Host On-Demand Connection Screen

The first time Host On-Demand is loaded using Netscape Communicator, permission is asked (you will see a request similar to Figure 221 on page 227) because the Java terminal emulator is being used to connect to another host. If you grant the requested permission, the communication is established.

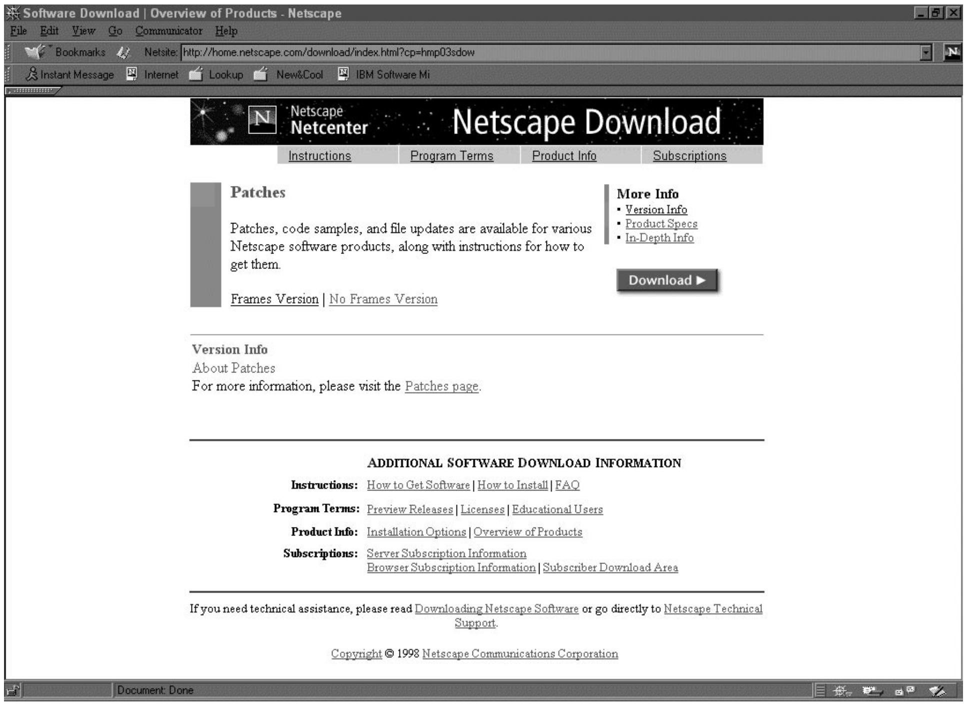


```
VM/ESA ONLINE
 -  Poughkeepsie, N.Y. -                ***                **
 International Technical   *** **     * ****        **  *********  ****
  Support Organization    ********* **   *      *  *********************
                          **************       **  * ***************** *
 Help:                      ************         * ********************
 ---------                  **********         **************************
  Internal    8-293-1660    * *****  *        **** ******************** * *
  External  914-433-1660       **                    * * ** **********    *
 VM Level:  VM/ESA V2 R2         * ***        ******   ****************  *
                                  ******     ******** **** *  ***  **
 You selected this node          *********   ********* ****    *    * *   *
 through a VTAM network.          ******       *******               * **
                                   ****          ****                 *****
 To return to VTAM from             **           ***  *                ****
 this panel, enter: VMEXIT           *     System Use is Restricted to IBM
                                          Management Approved Purposes Only
Fill in your USERID and PASSWORD and press ENTER
(Your password will not appear when you type it)
USERID   ===>  _
PASSWORD ===>

COMMAND  ===>
                                                           RUNNING   WTSCPOK
MA*    a X []
```

Figure 219. IBM Host On-Demand Running After Granting Permissions

The trusted Java applet or JavaScript file security model would allow a Java applet on a Web page to be used to install or update an application on the end user's hard disk.

Netscape has implemented this model for an installation program used to upgrade the JVM level in Netscape Communicator. We connected to the Netscape Web site and followed the links to SmartUpdate at http://home.netscape.com/download/su1.html.



Software Download | Overview of Products - Netscape

File Edit View Go Communicator Help

Bookmarks Netsite: http://home.netscape.com/download/index.html?cp=hmp03sdow

Instant Message Internet Lookup New&Cool IBM Software Mi

Netscape Netcenter Netscape Download

Instructions | Program Terms | Product Info | Subscriptions

Patches

Patches, code samples, and file updates are available for various Netscape software products, along with instructions for how to get them.

Frames Version | No Frames Version

More Info
- Version Info
- Product Specs
- In-Depth Info

Download ▶

Version Info
About Patches
For more information, please visit the Patches page.

ADDITIONAL SOFTWARE DOWNLOAD INFORMATION

Instructions: How to Get Software | How to Install | FAQ
Program Terms: Preview Releases | Licenses | Educational Users
Product Info: Installation Options | Overview of Products
Subscriptions: Server Subscription Information
Browser Subscription Information | Subscriber Download Area

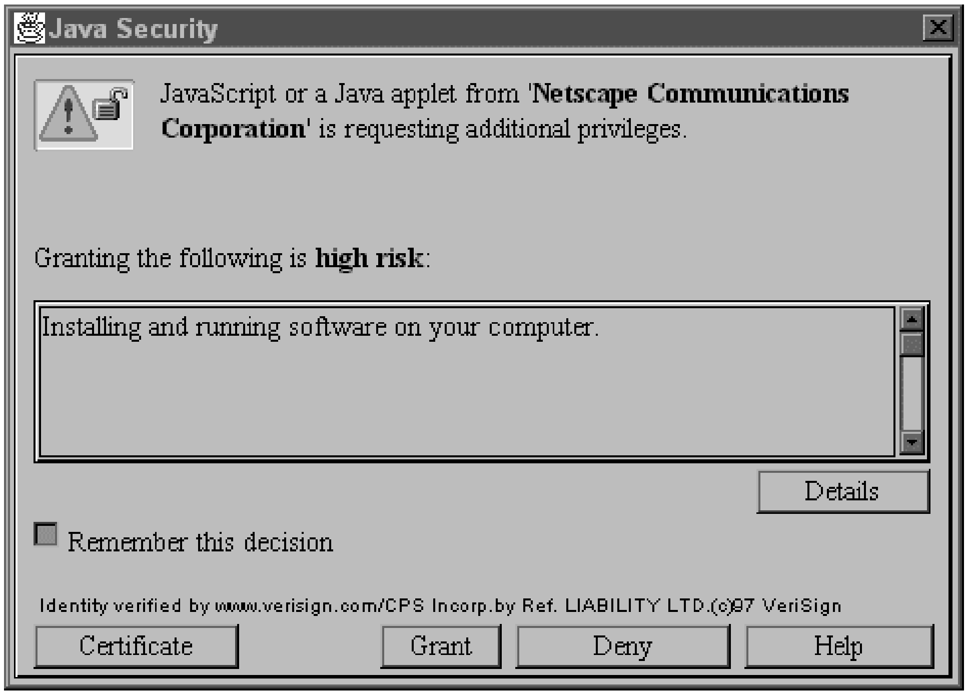
If you need technical assistance, please read Downloading Netscape Software or go directly to Netscape Technical Support.

Copyright © 1998 Netscape Communications Corporation

Document: Done

Figure 220. Netscape Patches Download Page

SmartUpdate is an application that requires client authentication, and so you must register before it can be used. When we selected to upgrade the Java on our system we received the following warning.



Java Security

JavaScript or a Java applet from '**Netscape Communications Corporation**' is requesting additional privileges.

Granting the following is **high risk**:

Installing and running software on your computer.

Details

Remember this decision

Identity verified by www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.(c)97 VeriSign

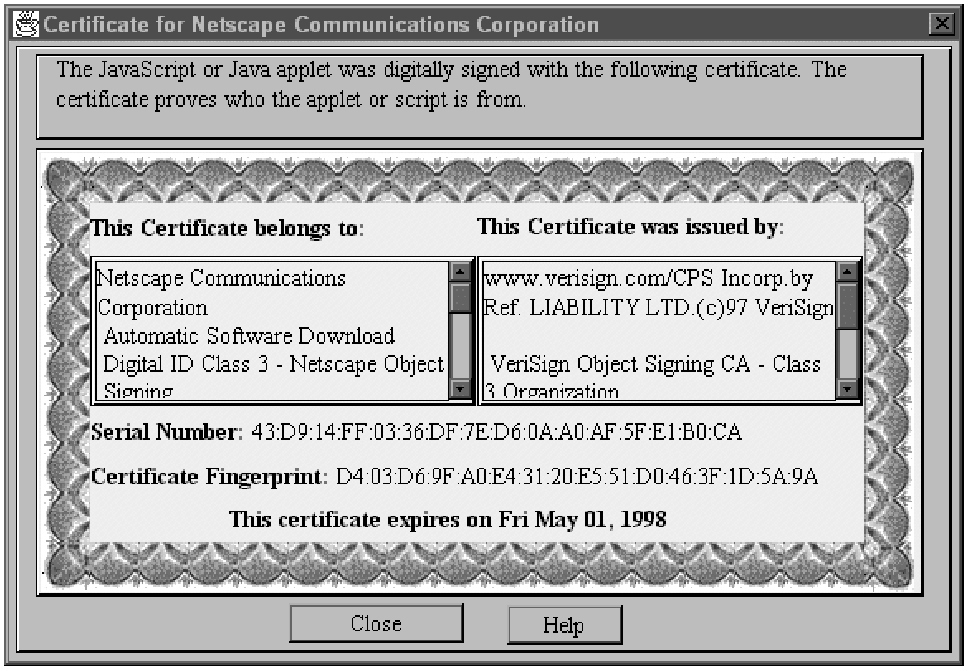
Certificate | Grant | Deny | Help

Figure 221. Trusted Code Warning

In the picture above, notice that the applet will tell you what it wants to do. In this case, the alert panel will let you select among the following options:

- **Certificate**

Click on this button to view the details about the digital certificate of the entity that signed the code. All the information about that digital certificate will be displayed:



Certificate for Netscape Communications Corporation

The JavaScript or Java applet was digitally signed with the following certificate. The certificate proves who the applet or script is from.

This Certificate belongs to:

Netscape Communications Corporation
Automatic Software Download
Digital ID Class 3 - Netscape Object Signing

This Certificate was issued by:

www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.(c)97 VeriSign
VeriSign Object Signing CA - Class 3 Organization

Serial Number: 43:D9:14:FF:03:36:DF:7E:D6:0A:A0:AF:5F:E1:B0:CA

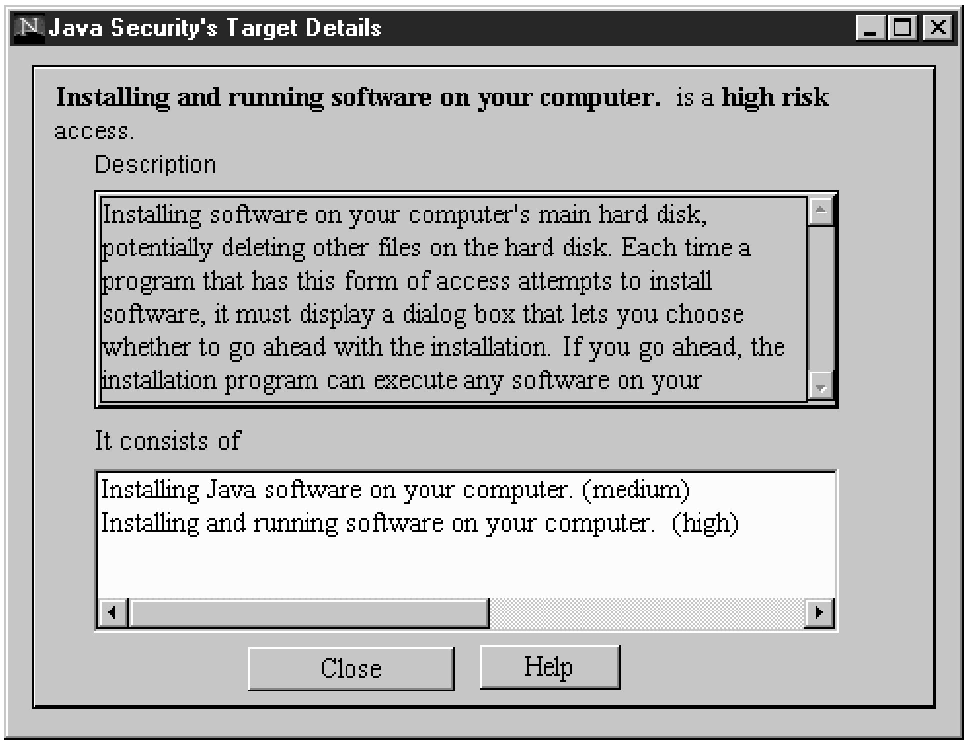
Certificate Fingerprint: D4:03:D6:9F:A0:E4:31:20:E5:51:D0:46:3F:1D:5A:9A

This certificate expires on Fri May 01, 1998

Figure 222. Digital Certificate

- **Details**

  Click on this button to view more information about the special access this signed code is requesting. The Java Security's Target Details window will be brought up:


Java Security's Target Details

Installing and running software on your computer. is a **high risk** access.

Description

Installing software on your computer's main hard disk, potentially deleting other files on the hard disk. Each time a program that has this form of access attempts to install software, it must display a dialog box that lets you choose whether to go ahead with the installation. If you go ahead, the installation program can execute any software on your

It consists of

Installing Java software on your computer. (medium)
Installing and running software on your computer. (high)

Figure 223. Java Security's Target Details Window

As you can see, this box lists exactly what access privileges the application is requesting. It is a good security measure to read these privilege requests before granting them.

- **Grant**

  Click on this button to grant these specific privileges only for this Communicator session.

- **Remember this decision**

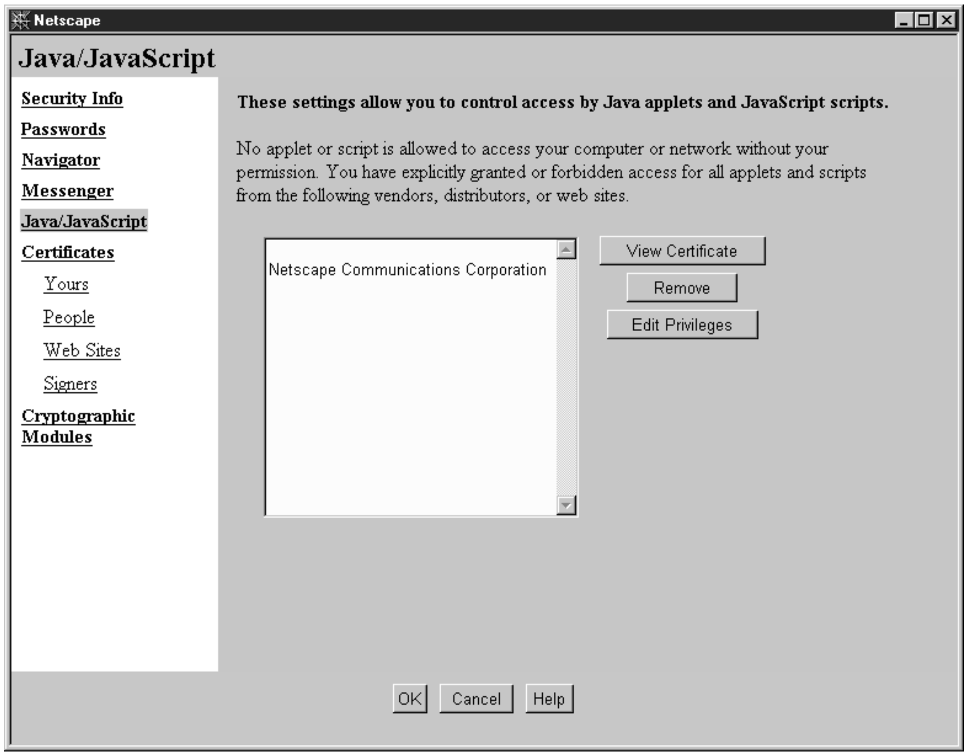
  Mark this check box at the bottom of the page before clicking **Grant**, in order to grant any signed Java applets or JavaScript files from this same originator the specific privileges requested. In this case the digital certificate is saved to your system and these same privileges will be granted in future Communicator sessions as well.

- **Deny**

  Click this button to deny the specifically requested privileges. No Java applet will be downloaded to your system and no JavaScript script will be executed.

We pressed the **Grant** button after viewing the certificate and reading the permissions requested. Then the code proceeded to transfer the upgrade and perform the installation. As we showed in Chapter 2, “The New Java 1.2 Security Model” on page 9, the permissions model in the JDK 1.2 will have more granular controls for granting or denying specific permissions to applets. Unsigned applets also will be enabled to receive privileges, and things will be different for local code as well, since by default it will no longer be granted all privileges.

We have seen in Figure 185 on page 188 how you can grant or deny specific privileges to Java applets or JavaScript files signed by a particular entity, whose certificate appears in the certificate list. That list was empty since no Java applet or JavaScript file signer had been registered to our system yet. If you mark the **Remember this decision** check box before granting the permission, the digital certificate of the organization or individual distributing this signed code is saved to your system. We tried this and we noticed that, from that moment, the digital certificate of Netscape Communications Corporation was visible in the digital certificate list of the Java/JavaScript panel.


Netscape

Java/JavaScript

These settings allow you to control access by Java applets and JavaScript scripts.

No applet or script is allowed to access your computer or network without your permission. You have explicitly granted or forbidden access for all applets and scripts from the following vendors, distributors, or web sites.

Netscape Communications Corporation

Figure 224. Java/JavaScript Panel - Digital Certificates Already Approved

This list gives you the opportunity to change access privileges you granted, on a certificate-by-certificate basis at any time. If you select a certificate from that list, you can use the buttons on the right to do one of the following tasks:

- **View Certificate**

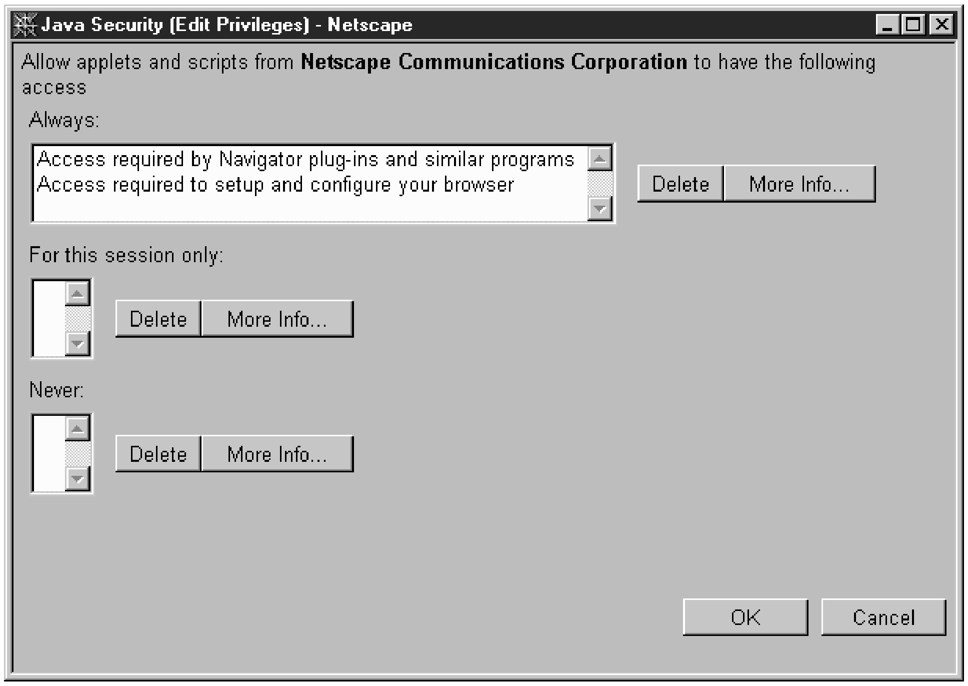
  Click this button and you will have a detailed view of the certificate, similar to Figure 222 on page 228.

- **Remove**

  Click this button if you want to remove the highlighted certificate from the list. Java applets and JavaScript files signed from this point of origin will no longer be downloaded and executed on your system. Instead, you will be presented again with the option of approving a digital certificate.

- **Edit Privileges**

  Click this button for the Edit Privileges page, shown next:



Allow applets and scripts from **Netscape Communications Corporation** to have the following access

Always:

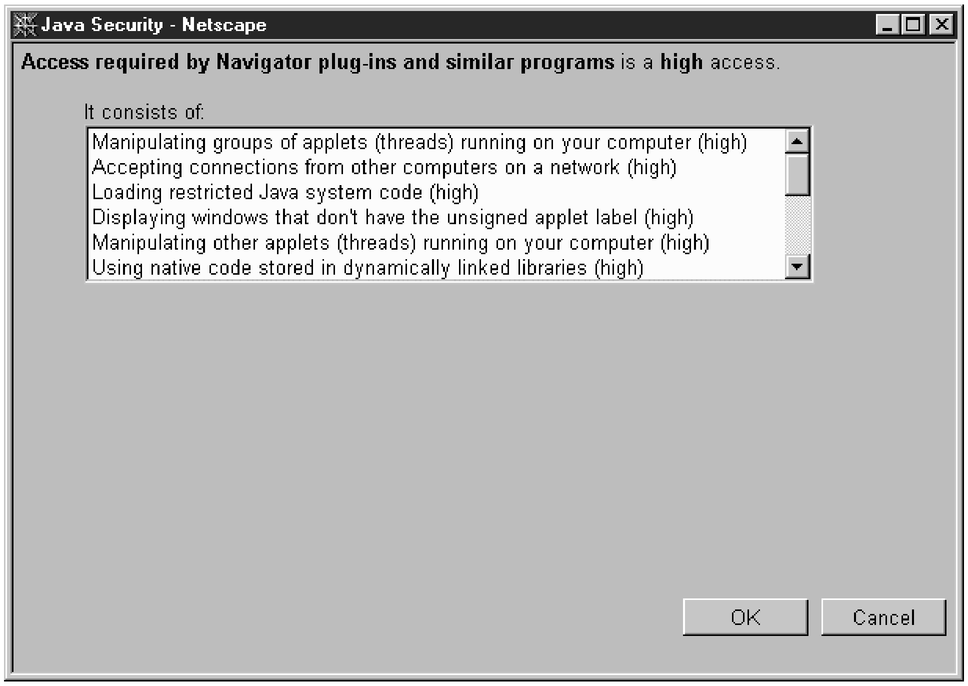
Access required by Navigator plug-ins and similar programs
Access required to setup and configure your browser

For this session only:

Never:

Figure 225. Edit Privileges Page

In this dialog box, you can select which specific privileges you want to grant for this session only or for any future sessions. You may also be able to deny specific privileges. Clicking the **More Info...** button, you will be able to get more specific information about the permission requested:



Access required by Navigator plug-ins and similar programs is a **high** access.

It consists of:

Manipulating groups of applets (threads) running on your computer (high)
Accepting connections from other computers on a network (high)
Loading restricted Java system code (high)
Displaying windows that don't have the unsigned applet label (high)
Manipulating other applets (threads) running on your computer (high)
Using native code stored in dynamically linked libraries (high)

Figure 226. More Information About the Access Required

Let's see again Figure 221 on page 227, Figure 223 on page 229 and Figure 226. You can read that the security risks that we ran allowing a signed code to install a new product on our system are considered by Netscape `high`.

Netscape provides three risk level categories:

1. High risk

   A major security attack is possible, permitting severe damage to your system or data. Major violation of privacy is possible, such as reading any information from hard disks connected to your computer. Very significant permissions may be requested, such as establishing a connection over the network to a remote computer.

2. Medium risk

   Major violation of privacy is possible, such as reading any information from a hard disk connected to your computer. Some significant services may be requested, such as writing files on a hard disk or sending e-mail on your behalf.

3. Low risk

   Minor violation of privacy is possible, such as reading your user ID. Relatively minor services may be requested, such as writing a single file to a specified non-critical directory on a hard disk connected to your computer.

Unfortunately this differentiation is often useless, since almost all applets want to read or write to the hard disk, and they all would fall into the high risk category.

We recommend at this point that you read the detailed explanation on signed and unsigned applets in the new Java 1.2 security model, that you can find in 2.7, “Applets in the New JDK 1.2 Security Model” on page 52.

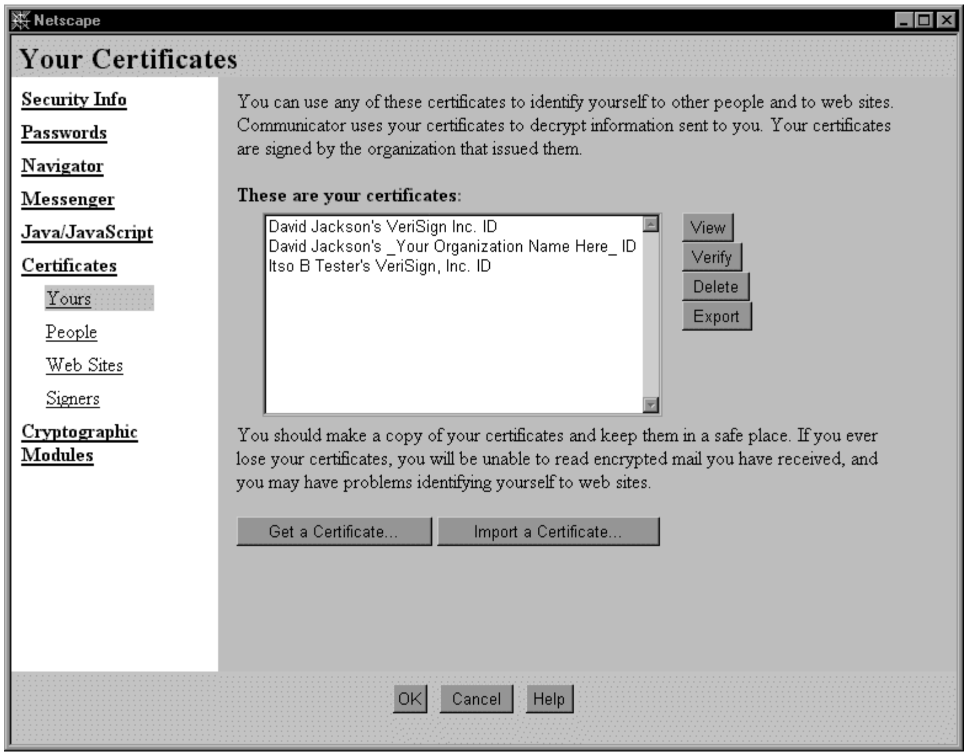
5.4 How to Manage Client Certificates

In this section we will see the main uses of client certificates.

5.4.1 Obtaining a VeriSign Evaluation Certificate

If you need a certificate to authenticate yourself on the Web and in electronic mail messages, VeriSign offers a service that allows you to obtain free evaluation certificates. The same certificate can also be used for form signing (see 5.4.5, “Using Certificates with Form Signing” on page 243). Follow the steps in this section to obtain your own personal Class 1 evaluation certificate (a discussion on different classes of VeriSign digital IDs is in 5.1.1, “Netscape Navigator” on page 181).

1. Click on the **Security** button from Netscape Navigator's main menu.
2. Click on **Yours** under the Certificates item in the Security Info list.



Your Certificates

You can use any of these certificates to identify yourself to other people and to web sites. Communicator uses your certificates to decrypt information sent to you. Your certificates are signed by the organization that issued them.

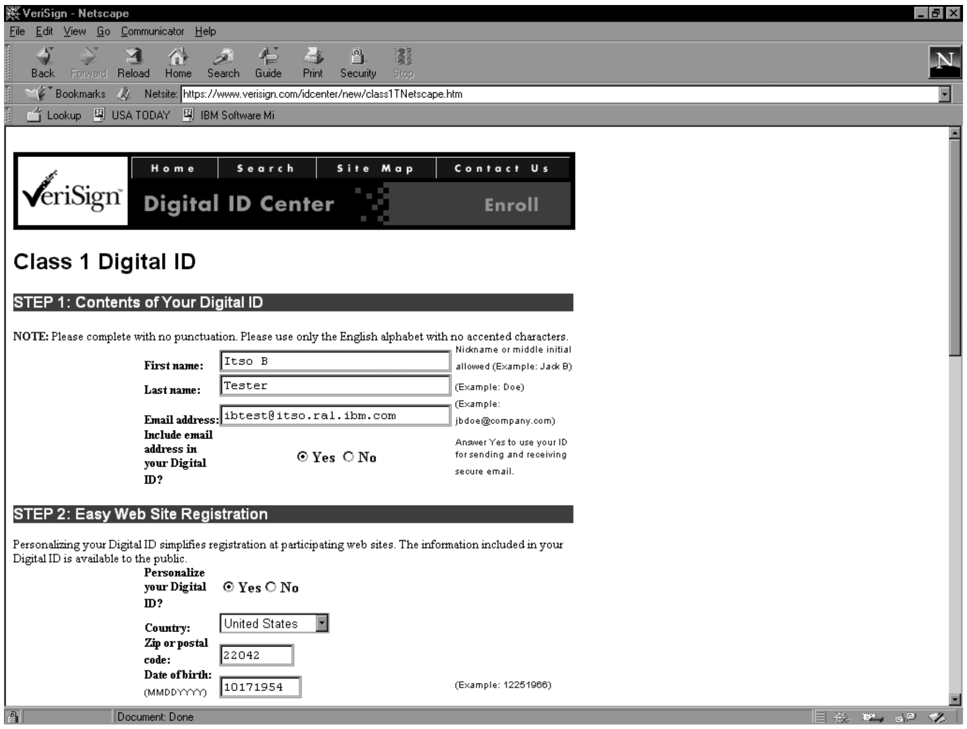
These are your certificates:

David Jackson's VeriSign Inc. ID
David Jackson's _Your Organization Name Here_ ID
Itso B Tester's VeriSign, Inc. ID

You should make a copy of your certificates and keep them in a safe place. If you ever lose your certificates, you will be unable to read encrypted mail you have received, and you may have problems identifying yourself to web sites.

Figure 227. Netscape Security Configuration Panel

3. Click on **Get a Certificate**. Communicator connects you to https://certs.netscape.com/client.html. The Certificate Authority Services page offers several choices to obtain a digital certificate.
4. Click on the **VeriSign** link. Communicator connects you to the VeriSign home page.
5. Select **Digital ID - Browsers** and then click on **Go**.
6. Communicator takes you to the Digital ID Center at VeriSign. Click on **Enroll Now** to enroll for your trial certificate and get your VeriSign digital ID.
7. On the next Web page, you select the type of digital ID you require. Click on the **Netscape** icon.
8. Communicator sends you to the Digital IDs for Browser Web page. Click on **Class 1 Digital ID** to get a Class 1 evaluation certificate (valid for 60 days). If this is not offered, try https://digitalid.verisign.com/class1Netscape.htm.
9. Next you must provide the information necessary to enroll for your certificate.


VeriSign - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Netsite: https://www.verisign.com/idcenter/new/class1TNetscape.htm
Lookup USA TODAY IBM Software Mi
Home Search Site Map Contact Us
VeriSign
Digital ID Center
Enroll
Document: Done

Class 1 Digital ID

STEP 1: Contents of Your Digital ID

NOTE: Please complete with no punctuation. Please use only the English alphabet with no accented characters.

First name:	Itso B	Nickname or middle initial allowed (Example: Jack B)
Last name:	Tester	(Example: Doe)
Email address:	ibtest@itso.ral.ibm.com	(Example: jbdoe@company.com)
Include email address in your Digital ID?	[✓ Yes] No	Answer Yes to use your ID for sending and receiving secure email.

STEP 2: Easy Web Site Registration

Personalizing your Digital ID simplifies registration at participating web sites. The information included in your Digital ID is available to the public.

Personalize your Digital ID?	[✓ Yes] No	
Country:	United States	
Zip or postal code:	22042	
Date of birth: (MMDDYYYY)	10171954	(Example: 12251966)

Figure 228. VeriSign Certificate Registration Page

10. We recommend that you do not use your preferred name. Instead, use a similar name, but one you do not mind discarding at a later date. The reason: when you decide to upgrade to a full-service digital ID, you will *not* be able to upgrade a trial ID. Instead, you will have to enroll for a new digital ID. And because names must be unique, you cannot use the same name for your full-service digital ID.

11. Enter your correct e-mail address in order to receive the required reply from VeriSign.

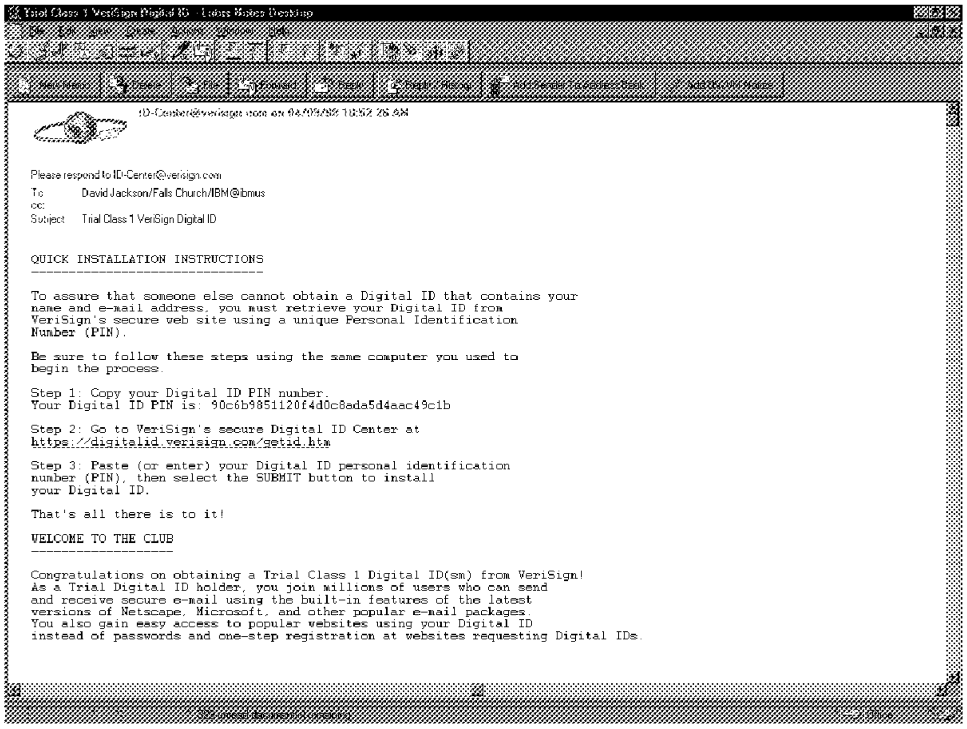
12. Read the Digital ID Subscriber Agreement. If you agree to be bound by the terms and conditions, click on **Accept**.

13. Communicator now wants to generate a private key for your certificate. This private key will be used along with the certificate you are requesting to identify you to Web sites and via e-mail. During the process of generating the private key, Communicator offers you the possibility to protect your private key with a password.

14. Click on **OK** to begin the process of generating your certificate. You must await an electronic mail message from VeriSign with your digital ID PIN. Meanwhile, Communicator takes you to a Web page that indicates that your enrollment is complete.

15. Wait now for an e-mail from VeriSign.

16. Reading the e-mail, when it arrives, you may now connect to https://digitalid.verisign.com/getid.htm as instructed.



ID-Center@verisign.com on 04/09/98 10:52 26 AM

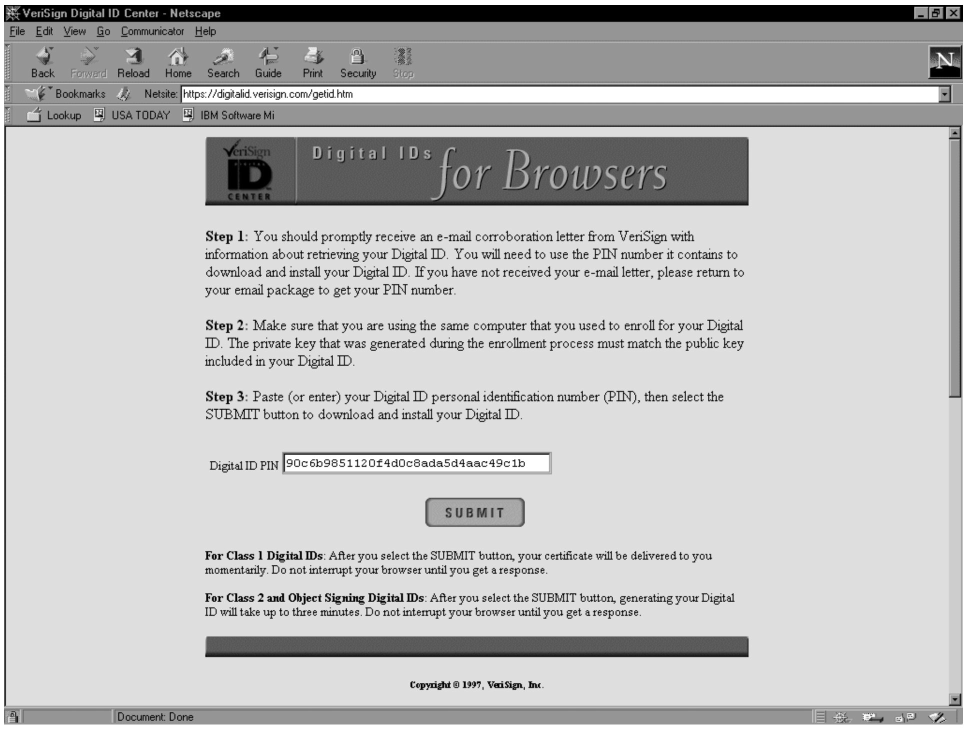
Please respond to ID-Center@verisign.com

To David Jackson/Falls Church/IBM@ibmus
cc:
Subject Trial Class 1 VeriSign Digital ID

```
QUICK INSTALLATION INSTRUCTIONS
-------------------------------

To assure that someone else cannot obtain a Digital ID that contains your
name and e-mail address, you must retrieve your Digital ID from
VeriSign's secure web site using a unique Personal Identification
Number (PIN).

Be sure to follow these steps using the same computer you used to
begin the process.

Step 1: Copy your Digital ID PIN number.
Your Digital ID PIN is: 90c6b9851120f4d0c8ada5d4aac49c1b

Step 2: Go to VeriSign's secure Digital ID Center at
https://digitalid.verisign.com/getid.htm

Step 3: Paste (or enter) your Digital ID personal identification
number (PIN), then select the SUBMIT button to install
your Digital ID.

That's all there is to it!

WELCOME TO THE CLUB
-------------------

Congratulations on obtaining a Trial Class 1 Digital ID(sm) from VeriSign!
As a Trial Digital ID holder, you join millions of users who can send
and receive secure e-mail using the built-in features of the latest
versions of Netscape, Microsoft, and other popular e-mail packages.
You also gain easy access to popular websites using your Digital ID
instead of passwords and one-step registration at websites requesting Digital IDs.
```

Figure 229. VeriSign Certificate e-mail

17. The secure Web page that you get is the Digital ID window. Notice that the e-mail that VeriSign sends you does not contain your certificate. It wouldn't be safe to send it via e-mail. It contains instead a unique, personal 32-character digital ID PIN, that you must cut and paste or type in the Digital ID PIN window. Security and confidentiality are granted by the fact that you access this Web page through the SSL protocol and that the certificate is not issued unless you use the same computer with which you started the process.


VeriSign Digital ID Center - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Netsite: https://digitalid.verisign.com/getid.htm
Lookup USA TODAY IBM Software Mi
Digital IDs for Browsers

Step 1: You should promptly receive an e-mail corroboration letter from VeriSign with information about retrieving your Digital ID. You will need to use the PIN number it contains to download and install your Digital ID. If you have not received your e-mail letter, please return to your email package to get your PIN number.

Step 2: Make sure that you are using the same computer that you used to enroll for your Digital ID. The private key that was generated during the enrollment process must match the public key included in your Digital ID.

Step 3: Paste (or enter) your Digital ID personal identification number (PIN), then select the SUBMIT button to download and install your Digital ID.

Digital ID PIN: 90c6b9851120f4d0c8ada5d4aac49c1b

SUBMIT

For Class 1 Digital IDs: After you select the SUBMIT button, your certificate will be delivered to you momentarily. Do not interrupt your browser until you get a response.

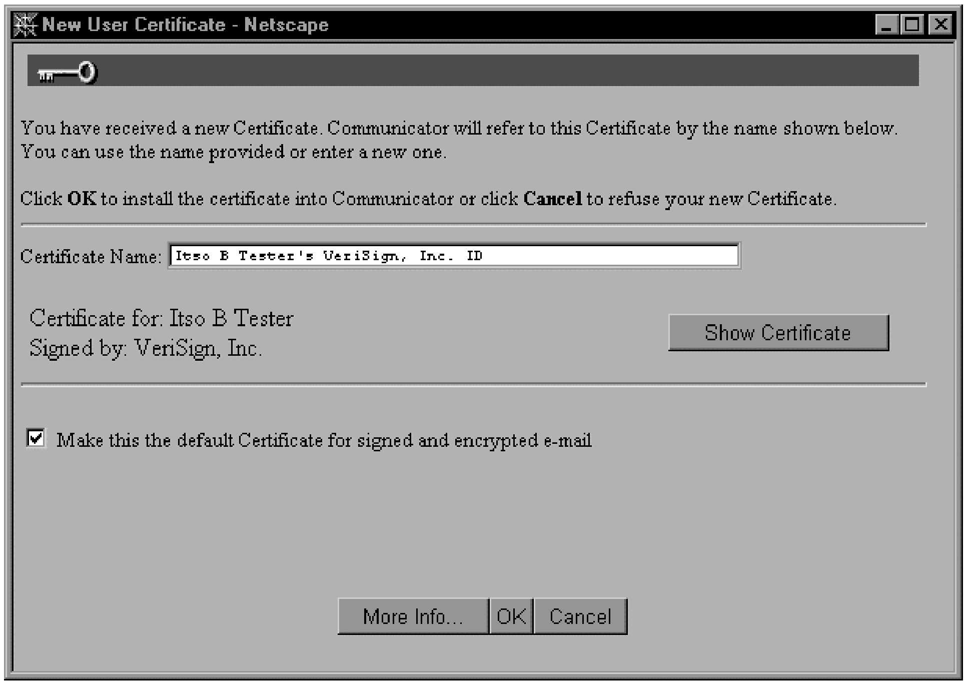
For Class 2 and Object Signing Digital IDs: After you select the SUBMIT button, generating your Digital ID will take up to three minutes. Do not interrupt your browser until you get a response.

Copyright © 1997, VeriSign, Inc.

Document: Done

Figure 230. VeriSign Certificate Get ID Page

18. Then you can click **SUBMIT**. When your new user certificate is issued successfully, you see a screen that says you have received your new certificate.



You have received a new Certificate. Communicator will refer to this Certificate by the name shown below. You can use the name provided or enter a new one.

Click **OK** to install the certificate into Communicator or click **Cancel** to refuse your new Certificate.

Certificate Name: Itso B Tester's VeriSign, Inc. ID

Certificate for: Itso B Tester
Signed by: VeriSign, Inc.

Show Certificate

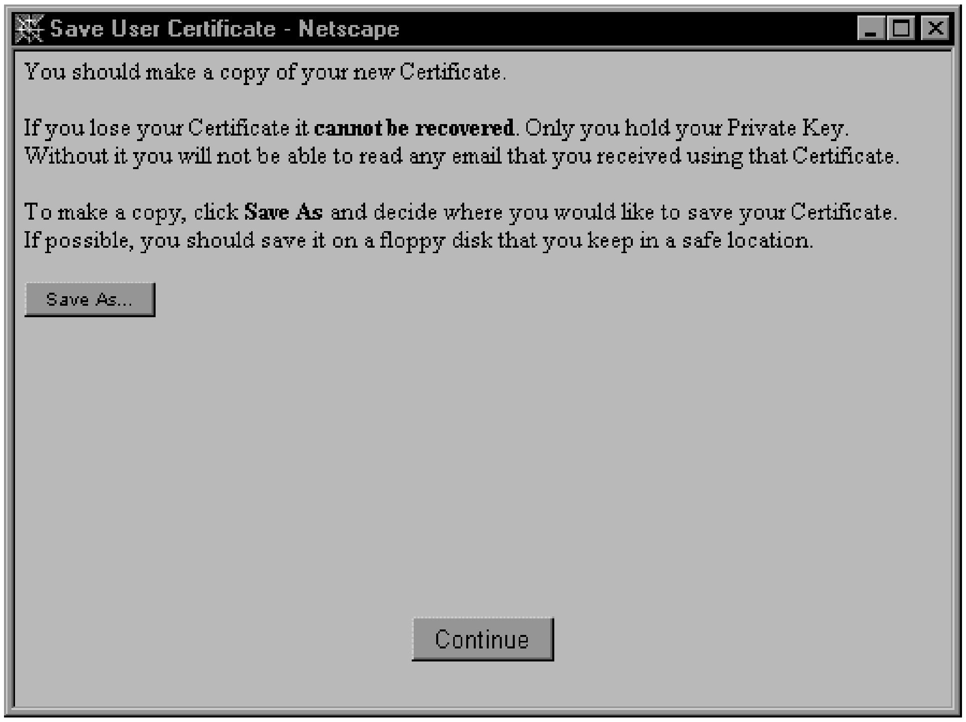
☑ Make this the default Certificate for signed and encrypted e-mail

More Info... OK Cancel

Figure 231. Netscape Certificate Receipt

19. You can click on **Show Certificate** to see the particulars or **OK** to exit this window.

20. Communicator suggests that you should make a copy of your certificate. You can accept this security measure or click **Continue** to bypass this step.



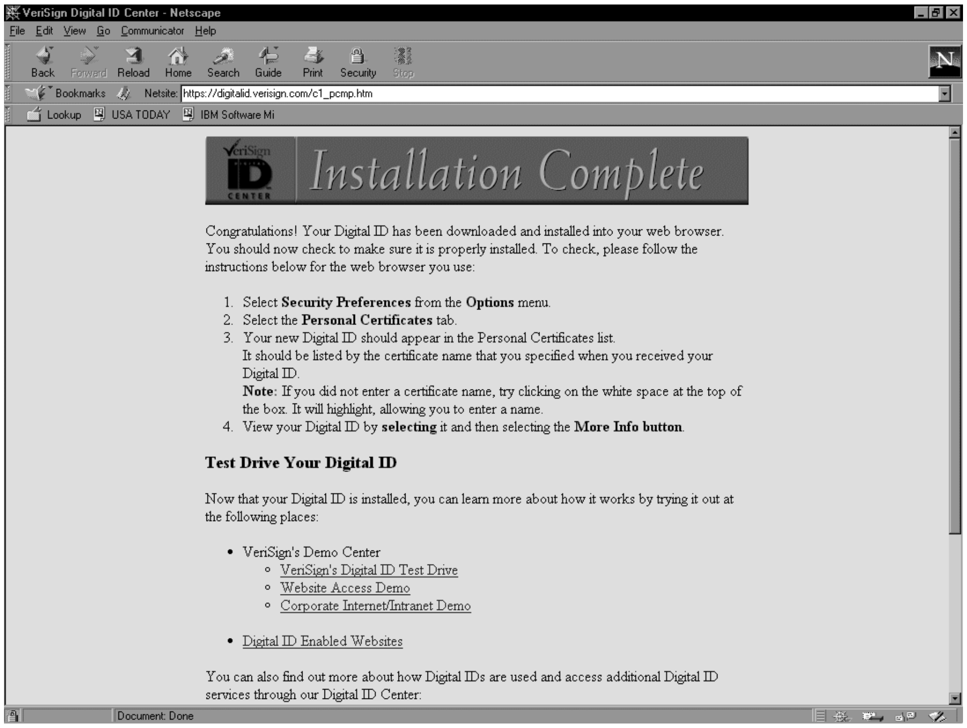
You should make a copy of your new Certificate.

If you lose your Certificate it **cannot be recovered**. Only you hold your Private Key. Without it you will not be able to read any email that you received using that Certificate.

To make a copy, click **Save As** and decide where you would like to save your Certificate. If possible, you should save it on a floppy disk that you keep in a safe location.

Figure 232. Netscape Certificate Save Information

21. Your digital ID is now established.


VeriSign Digital ID Center - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Netsite: https://digitalid.verisign.com/c1_pcmp.htm
Lookup USA TODAY IBM Software Mi
VeriSign Digital ID Center
Installation Complete

Congratulations! Your Digital ID has been downloaded and installed into your web browser. You should now check to make sure it is properly installed. To check, please follow the instructions below for the web browser you use:

1. Select **Security Preferences** from the **Options** menu.
2. Select the **Personal Certificates** tab.
3. Your new Digital ID should appear in the Personal Certificates list. It should be listed by the certificate name that you specified when you received your Digital ID. **Note:** If you did not enter a certificate name, try clicking on the white space at the top of the box. It will highlight, allowing you to enter a name.
4. View your Digital ID by **selecting** it and then selecting the **More Info button**.

Test Drive Your Digital ID

Now that your Digital ID is installed, you can learn more about how it works by trying it out at the following places:

- VeriSign's Demo Center
  - VeriSign's Digital ID Test Drive
  - Website Access Demo
  - Corporate Internet/Intranet Demo
- Digital ID Enabled Websites

You can also find out more about how Digital IDs are used and access additional Digital ID services through our Digital ID Center:

Figure 233. VeriSign Certificate Installation Complete

Now, return via the **Security** button to the Security Info window and once more click on **Yours** under the Certificates item. You should see the certificate just obtained now in the list box. You can also **View** or **Verify** your certificate.

5.4.2 Using a Certificate to Access Secure Web Sites

Now that you have a digital certificate installed on your browser, you can easily access secure Web sites using the new certificate. The following Web sites are non-secure, but they contain links to secure Web sites, that require that both client (your browser) and server (the Web site) present their digital IDs. If you try to access them, you can experiment with your new certificate. Then, clicking on the **Security** icon on the Navigator window, you can see the certificate provided by the Web site you accessed.

- http://www.verisign.com/demos/index.html
- http://www.verisign.com/showcase/index.html
- http://www.verisign.com/products/corporate/hr_demo/index.html
- http://www.verisign.com/products/sites/job_demo/index.html
- http://www.verisign.com/demos/inbox/index.html
- http://www.virtualvin.com
- http://www.bassandco.com/
- http://form.netscape.com/ibd/html/ibd.html

Notice that Netscape Navigator caches security information after you access a secure Web site and uses the cached information the next time you access the same secure Web page. If you remove the digital certificate from your browser, when accessing secure Web sites, you must restart Communicator to be denied access to secure Web pages that you previously visited.

5.4.3 Using Certificates for E-Mail

Follow these steps to send a signed message to yourself:

1. Click on the **Mailbox** icon in the bottom-right corner of Communicator.
2. Click on the **New Msg** button.
3. Click on the **Message Sending Options** button and check the Signed check box only.
4. Complete a message to yourself.
5. Click on the **Send** button.
6. When the message lands in your Inbox - Netscape Folder, open it and click on the Signed stamp in the top, right corner of the message. Note that your message was not encrypted but was signed as indicated in the resulting Communicator screen.

Follow these steps to send a signed and encrypted message to yourself:

1. Once more, click on the **New Msg** button from within the Mailbox.
2. Through the **Message Sending Options** button, check the Encrypted as well as the Signed check box.
3. Complete and send a message to yourself.

4. When the message lands in your Inbox - Netscape Folder, open it and click on the Encrypted and Signed stamp.
5. Now you will see an indication that your message was encrypted and the algorithm used to do so.

Follow similar steps to send messages to other persons.

5.4.4 How to Export and Import Certificates

When the key pair for a digital certificate is generated in the browser, the private key of the key pair is stored in the browser's directory on the hard disk of the PC that the browser is running on. If the user has both Netscape and Microsoft browsers, and switches between them, a certificate created in one browser will not be available in the other browser. There is a requirement on the user to understand why they are not able to use an application with Netscape that they are able to use with Internet Explorer because they received their certificate for the application while using Internet Explorer.

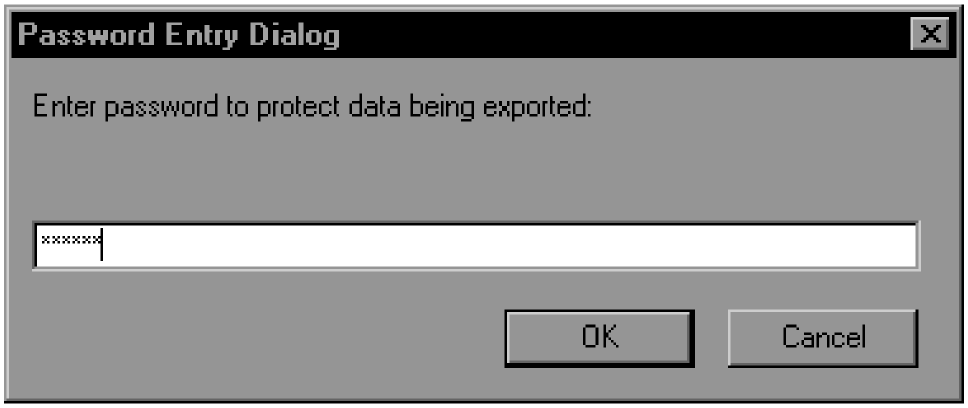
However it is possible to export a certificate to a password protected file and import that certificate to another browser. Of course the other browser can also be on another computer, and in this case the file where you export the certificate must be simply transferred to the other computer, and then imported to the new browser. The tasks encountered here include:

- Exporting the certificate to a file (password required)
- Deleting the certificate export file after copying it to a diskette
- Deleting the certificate from the browser the user last left so that their certificate is kept private
- At the new machine, gaining access to the browser Key Ring Organizer (password required)
- Properly importing the certificate

As we will see, the certificate must be exported in the certificate format PKCS #12.

The Certificates section of the Netscape Communicator security page is used to perform functions related to receiving and using digital certificates. You can see 5.4.1, “Obtaining a VeriSign Evaluation Certificate” on page 232 for information on getting a certificate with the section labeled Yours. Figure 227 on page 233 also shows the feature used to export a digital certificate to a file by selecting a certificate and clicking on the **Export** button. This file is password protected so that on the import operation, the password would be required as an additional security measure for the certificate file. We will show the export steps next and show the import steps with Internet Explorer 4.0, which we had installed on the same machine. This export/import function might also be applied if you use more than one browser, or if you must move from PC to PC.

Once you have selected the certificate to be exported and clicked on the **Export** button, you may be asked for the password to the Netscape certificate database (also known as Key Ring Organizer or KRO), depending on the options you set for password access as shown in the configuration window in Figure 181 on page 184. Next you will be asked for a password for the export file. This password is very important because it will be also used to import the certificate in Microsoft Internet Explorer.

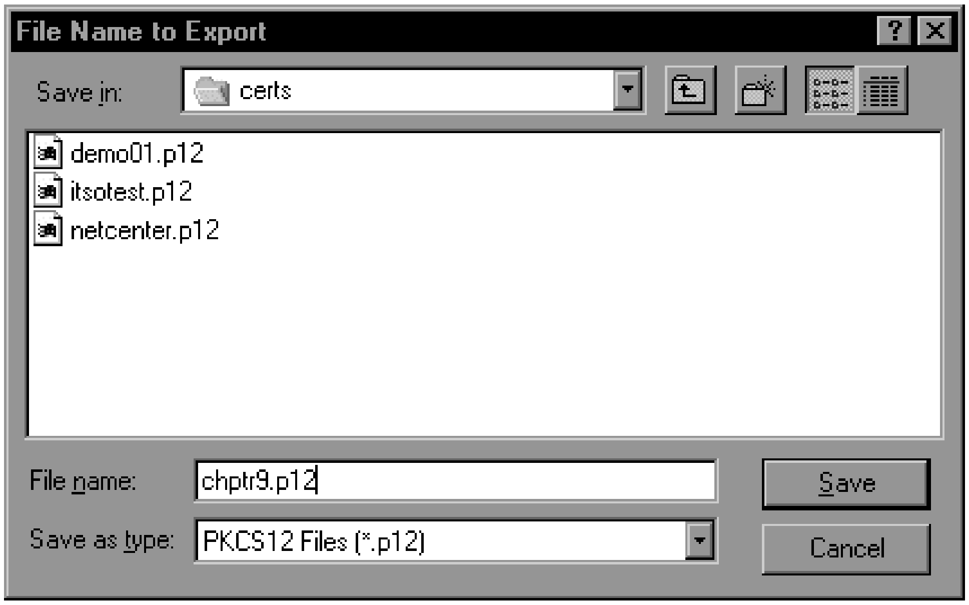

Password Entry Dialog

Enter password to protect data being exported:

******

Figure 234. Certificate Export Password

Then you will be asked to re-enter the password to confirm it. Next you will be asked for a file name to export. The export files are given the .p12 extension by default. This extension, that indicates that the certificate is in the format PKCS #12, is the one that the import function will search for when you import your certificate, so it is recommended that you not use a different extension.


File Name to Export

Save in: certs

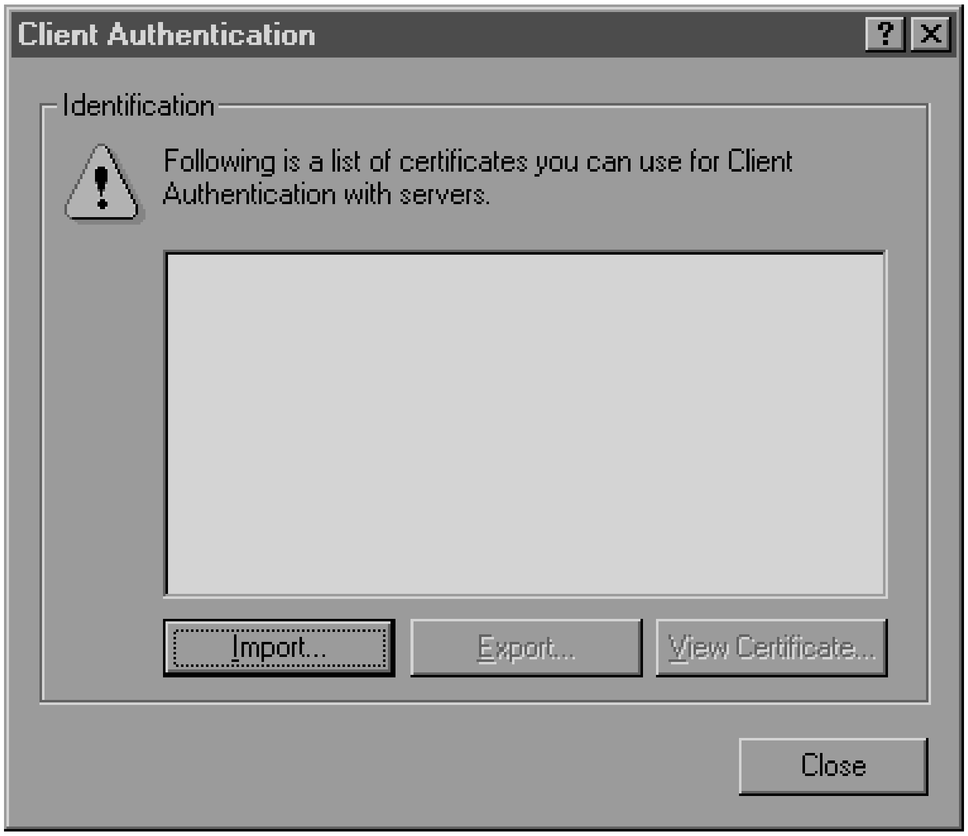
demo01.p12
itsotest.p12
netcenter.p12

File name: chptr9.p12

Save as type: PKCS12 Files (*.p12)

Figure 235. Certificate Export File - Filename

When you click on the **Save** button the file is written, and you will see a notification informing you that your certificate has been successfully exported. We see now how that certificate can be imported into Microsoft Internet Explorer (MSIE). First of all, you should open the Content page for the Internet Options configuration dialog, as shown in Figure 198 on page 201. Then, you should click the **Personal** button, found in the Certificates section, in order to see the list of all your personal digital certificates. As you can see, we had none installed in the MSIE browser because we had applied for and received all our certificates in the Netscape browser.

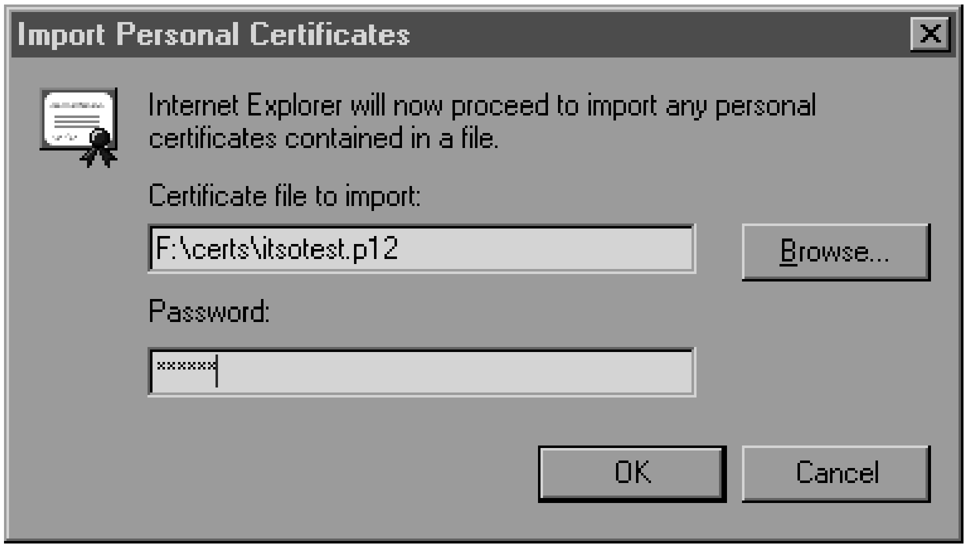

Client Authentication

Identification

Following is a list of certificates you can use for Client Authentication with servers.

Figure 236. Client Authentication - Personal Certificates

When you click on the **Import...** button on the screen above, you are presented with the following dialog. You are asked for the file name of the PKCS #12 file that was created when you exported your certificate, and the password associated with that file.


Import Personal Certificates

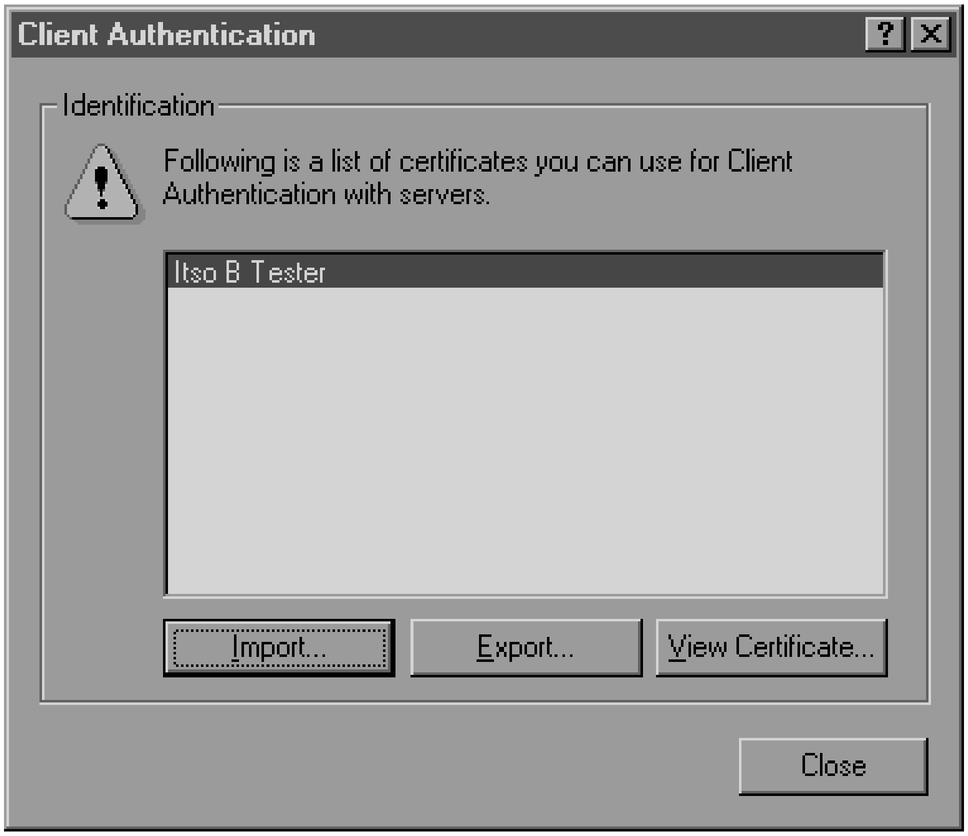
Internet Explorer will now proceed to import any personal certificates contained in a file.

Certificate file to import: F:\certs\itsotest.p12

Password: ******

Figure 237. Import Personal Certificates

When you click on **OK** in the screen above, the certificate is imported and displayed in the list of all your personal digital certificates.

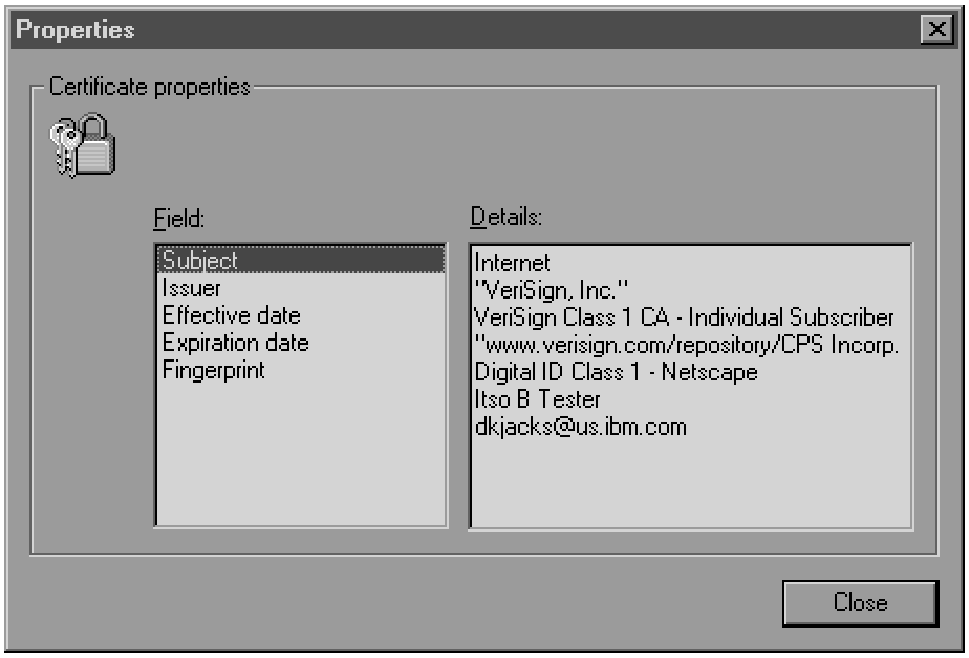


Following is a list of certificates you can use for Client Authentication with servers.

Itso B Tester

Figure 238. Imported Certificate

If you click on the **View Certificate...** button, you can see the details of the certificate in the following panel.



Internet
"VeriSign, Inc."
VeriSign Class 1 CA - Individual Subscriber
"www.verisign.com/repository/CPS Incorp.
Digital ID Class 1 - Netscape
Itso B Tester
dkjacks@us.ibm.com

Figure 239. Certificate Properties

5.4.5 Using Certificates with Form Signing

We know that we can secure the data stream from the browser to the Web server by using SSL to encrypt the data (see 3.1.2, “Lotus Domino Go Webserver SSL Setup” on page 71). We can add additional authentication in this process by requesting client side authentication and we can set Web server directives to use client side authentication for access control to areas and pages on our Web server (see 3.1.4, “SSL Client Authentication” on page 84). There is an overhead penalty to pay for setting up and using SSL. The browser and server must go through the process of authentication, and they must exchange a session encryption key. SSL has an impact because additional data must be exchanged in terms of number of packets and number of bytes to be transferred, which increases network traffic flow. Recent research that shows the performance cost of implementing SSL instead of a non-secure connection is shown at http://www.ics.raleigh.ibm.com/capacity/example8.htm.

An alternative to encrypting the connection with SSL may be to use Form Signing. If the information requested is not necessarily of a sensitive nature, this option may work for you. Form Signing makes sure that the data is not changed between origin and destination and it also ensures that it was sent by the owner of the private key related to the digital certificate used for signing. The data from the form is sent in clear text, and along with the form data, a signature of the data, created in the browser, is also sent.

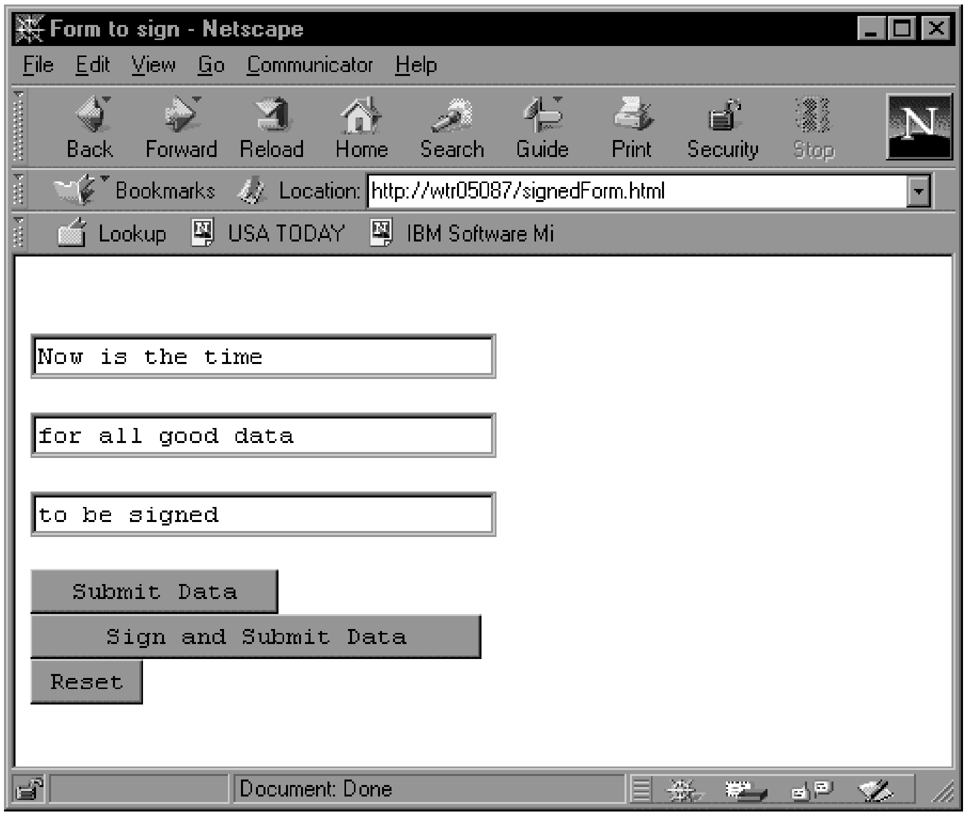
Beginning with Netscape Communicator 4.04, Netscape provided cryptographic functions in the JavaScript interpreter of Communicator. Information about form signing can be seen at http://developer.netscape.com/docs/manuals/security/signver/index.htm.

We downloaded the zip file listed for Windows NT and we received the signature verification tool. In the zip file were some sample HTML forms and Perl scripts to use form signing and verification. We made some modifications in the samples to get them to work on our Lotus Domino Go Webserver. The source of the HTML page used is listed here.

```
<html>
<head>
<title>Form to sign</title>
<script language="javascript">
<!--
function submitSigned(form){
  var signature = "";
  var dataToSign = "";
  var i;

  form.action='/cgi-bin/signedForm.pl';
  for (i = 0; i < form.length; i++)
    if (form.elements[i].type == "text")
      dataToSign += form.elements[i].value;

  // alert("Data to sign:\n" + dataToSign);
  signature = crypto.signText(dataToSign, "ask");
  /* alert("You cannot see this alert");
  alert("Data signature:\n" + signature); */
  if (signature != "error:userCancel") {
    for (i = 0; i < form.length; i++) {
      if (form.elements[i].type == "hidden") {
        if (form.elements[i].name == "dataToSign")
          form.elements[i].value = dataToSign;
        if (form.elements[i].name == "dataSignature")
          form.elements[i].value = signature;
      }
    }
    form.submit();
  }
}
//-->
</script>
</head>

<body>
<form method="POST" action="/cgi-bin/form.pl">
<pre>
<input type=hidden size=30 name=dataSignature>
<input type=hidden size=30 name=dataToSign>
<input type=text size=30 name=p>
<BR>
<input type=text size=30 name=q>
<BR>
<input type=text size=30 name=r>
<BR>
<input type=submit value="Submit Data">
<input type=button value="Sign and Submit Data" onclick=submitSigned(this.form)
<input type=reset value=Reset>
</pre>
</form>
</body>
</html>
```

Figure 240. Source Code for signedForm.html

This form provides a **Submit Data** button and a **Sign and Submit Data** button, as shown in the screen shot below.


Form to sign - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05087/signedForm.html
Lookup USA TODAY IBM Software Mi
Submit Data
Sign and Submit Data
Reset
Document: Done

Now is the time

for all good data

to be signed

Figure 241. Web Page for Form Data Signing

As you can see, this form uses the POST method. The function provided by the **Submit Data** button simply retrieves the data entered in the text fields of the form and sends it to the Web server without signing it. The **Submit Data** button calls the form.pl Perl script listed here and that must be installed in the CGI-Bin directory of the Web server. The function of form.pl just sends back to the browser the data that you entered and some environment variables it collected from the browser. This information is displayed on the client browser.

```
#!/perl/bin
print <<EOM;
Content-type: text/html

<html>
<head>
<title>
Sample of form processing
</title>
</head>
<body>
<pre>
EOM

$std_in = <STDIN>;
#print $std_in;
print "<B>Server Name:</B> , $ENV{'SERVER_NAME'}, <BR>, \n";
print "<B>Server Port:</B> ", $ENV{'SERVER_PORT'}, "<BR>", "\n";
print "<B>Server Software:</B> ", $ENV{'SERVER_SOFTWARE'}, "<BR>", "\n";
print "<B>Server Protocol:</B> ", $ENV{'SERVER_PROTOCOL'}, "<BR>", "\n";
print "<B>CGI Revision:</B> ", $ENV{'GATEWAY_INTERFACE'}, "<BR>", "\n";
print "<B>Browser:</B> ", $ENV{'HTTP_USER_AGENT'}, "<BR>", "\n";
print "<B>Remote Address:</B> ", $ENV{'REMOTE_ADDR'}, "<BR>", "\n";
print "<B>Remote Host:</B> ", $ENV{'REMOTE_HOST'}, "<BR>", "\n";
print "<B>Remote User:</B> ", $ENV{'REMOTE_USER'}, "<BR>", "\n";

print "You typed:\n";
@pairs = split (/&/, $std_in);

foreach $pair (@pairs) {
 ($key, $value) = split (/=/, $pair);
 $form_data{$key} = $value;
}

$firstline = $form_data{"p"};
$secndline = $form_data{"q"};
$thrdline = $form_data{"r"};

print "$firstline\n";
print "$secndline\n";
print "$thrdline\n";

print <<EOM;
</pre>
</body>
</html>
EOM

close OUT;
```

Figure 242. Source Code for form.pl

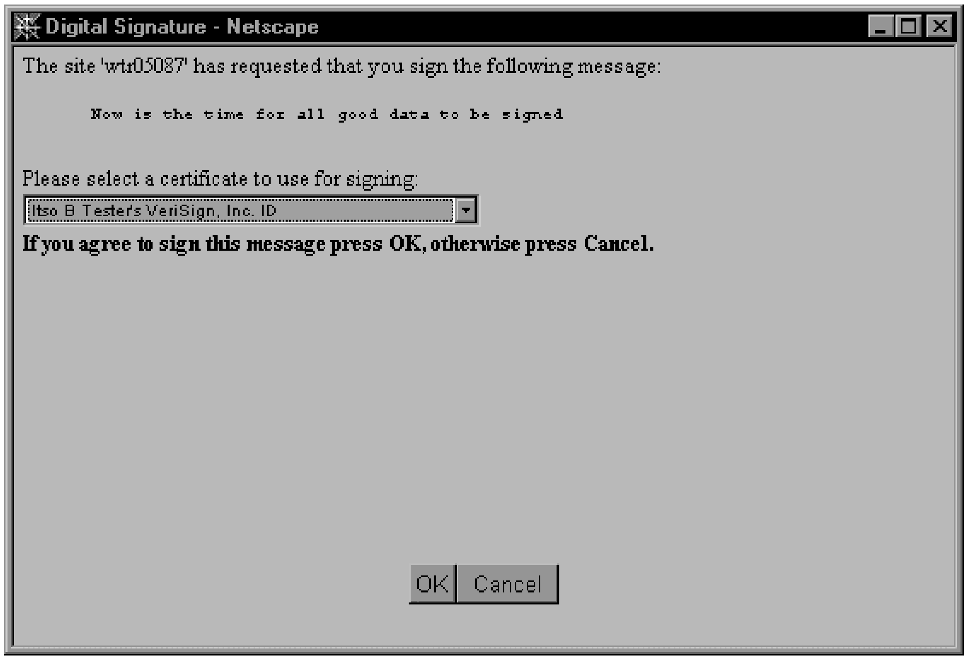
The top of the HTML source in Figure 240 on page 244 is JavaScript. This part of the HTML is processed after pressing the **Sign and Submit Data** button. The important line in the JavaScript section is

```
signature = crypto.signText(dataToSign, "ask");
```

This calls the JavaScript function to sign the form data before it is sent to the signedForm.pl Perl script pointed to in

```
form.action='/cgi-bin/signedForm.pl';
```

This Perl script also must be installed in the CGI-Bin directory of the Web server. When the **Sign and Submit Data** button is clicked, the browser accesses the certificate database to use the private key for the signing operation. If the database has not yet been accessed in this session, the user will be prompted for the database password. The user is presented with the KRO and asked to sign the data.



The site 'wtr05087' has requested that you sign the following message:

Now is the time for all good data to be signed

Please select a certificate to use for signing:

Itso B Tester's VeriSign, Inc. ID

If you agree to sign this message press OK, otherwise press Cancel.

Figure 243. Netscape Key Ring Organizer

The Perl script, signedForm.pl, is listed below.

```
#!/perl/bin
#print "Content-type: text/html\n\n";
print <<EOM;
Content-type: text/html

<html>
<head>
<title>
HTML Page generated by signedForm Script
</title>
</head>

<body>
<h1>HMTL Page dynamically generated by signedForm</h1>
<pre>
EOM

sub decode {
    read (STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
 print $buffer;
    @pairs = split(/&/, $buffer);
    foreach $pair (@pairs)
    {
        ($name, $value) = split(/=/, $pair);
        $value =~tr/+/ /;
#        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
        $FORM{$name} = $value;
        print "name=$name  value=$value<BR>\n";
    }
}


&decode();

$dataSignature = $FORM{'dataSignature'};
$dataToSign = $FORM{'dataToSign'};

unlink("signature");
open(FILE1,">signature") || die("Cannot open file for writing\n");

print FILE1 "$dataSignature";

close(FILE1);
unlink("data");
open(FILE2,">data") || die("Cannot open file for writing\n");

print FILE2 "$dataToSign";

close(FILE2);
```

Figure 244 (Part 1 of 2). Source Code for signedForm.pl

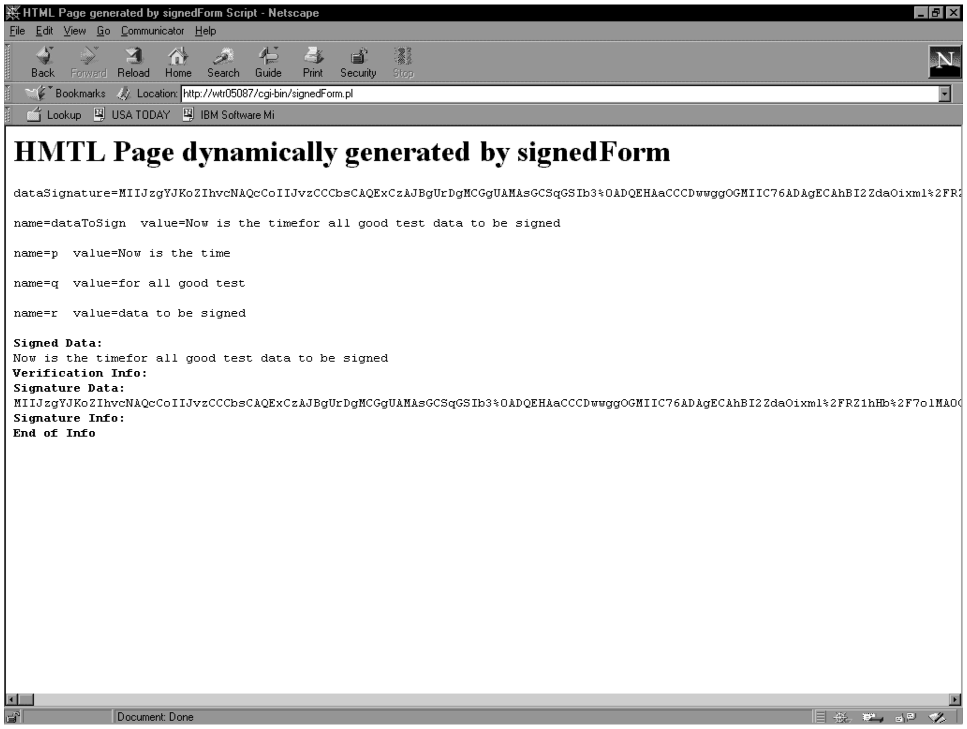
```
print "<BR><B>Signed Data:</B><BR>", "$dataToSign", "<BR>";

print "<BR><b>Verification Info:</b><BR>";

$verInfo = `./signver -D . -s signature -d data -v`;
# $verInfo = `./cgi-bin/signver -D . -s signature -d data -v`;
print "<font color=red><b>$verInfo</b></font><BR>";

print "<BR><B>Signature Data:</B><BR>", "$dataSignature", "<BR>";

print "<BR><b>Signature Info:</b><BR>";

foreach $line (`./signver -s signature -A`) {
# foreach $line (`./cgi-bin/signver -s signature -A`) {

     print "$line<BR>\n";
}

print "<b>End of Info</b><BR>";

print <<EOM;
</pre>
</body>
</html>
EOM

close OUT;
```

Figure 244 (Part 2 of 2). Source Code for signedForm.pl

It takes the data from STDIN and parses it into variable and data elements with the subroutine decode(). Next it writes the signature out to a file, and the data out to a second file. This is necessary to use the verification tool. The line

```
$verInfo = `./signver -D . -s signature -d data -v`;
```

invokes signver.exe, the signing verification tool, with the files signature and data as input. The other input variable, -D, points to the certificate data repository so that the verification tool can retrieve the sender's certificate to use the public key. We did get the signature and the data returned in a Web page.


HTML Page generated by signedForm Script - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05087/cgi-bin/signedForm.pl
Lookup USA TODAY IBM Software Mi
Document: Done

HMTL Page dynamically generated by signedForm

```
dataSignature=MIIJzgYJKoZIhvcNAQcCoIIJvzCCCbsCAQExCzAJBgUrDgMCGgUAMAsGCSqGSIb3%0ADQEHAaCCCDwwggOGMIIC76ADAgECAhBI2ZdaOixml%2FRZ

name=dataToSign  value=Now is the timefor all good test data to be signed

name=p  value=Now is the time

name=q  value=for all good test

name=r  value=data to be signed

Signed Data:
Now is the timefor all good test data to be signed
Verification Info:
Signature Data:
MIIJzgYJKoZIhvcNAQcCoIIJvzCCCbsCAQExCzAJBgUrDgMCGgUAMAsGCSqGSIb3%0ADQEHAaCCCDwwggOGMIIC76ADAgECAhBI2ZdaOixml%2FRZ1hHb%2F7olMAO
Signature Info:
End of Info
```

Figure 245. Signed Form Data

Form signing is a powerful technique. It can for example be used together with a servlet that would receive the PKCS #7 object created by the form signing JavaScript to verify the form data as being genuine, and then act on the form. If the digital certificate of the originator of the form was available, the public key could be used to encrypt the information being returned. Now we could be sure that only the person who has control of the private key related to the certificate, could view the data.

5.5 Cookie Security Implications

In this section we will show you how cookies can be considered as a privacy exposure rather than a security risk. What we will say in this section about cookie security is based on information available to date. It could happen that new bugs in some Web browsers are discovered that could make accepting cookies a danger for your system.

Web cookies are simple pieces of information passed between the client Web browser and the Web server during an HTTP transaction. The maximum size a cookie can have is 4 KB.

Cookies do not contain any information about the client that the server does not already know and they cannot do anything on the client machine that the client itself cannot already do, provided the browser is within the specifications.

Cookies were introduced as an answer to a fundamental problem of the Web's underlying HTTP 1.0 protocol: the lack of a state, or a persistent connection. The way the HTTP 1.0 protocol operates is that the client initiates an HTTP transaction by sending a request to a Web server (for example, it asks for a Web page), then the server responds and after that the HTTP transaction is automatically closed. If the client sends another request, there is no relation between the new request and the previous one. This approach has its advantages, in that it allows a Web server to serve many clients simultaneously, without incurring the overhead generated by keeping several sessions opened with all the clients that initiated a request. Moreover, since each request is independent of others, a secondary benefit is that a page on a machine can link directly to another page on a completely different machine, without any need for the user to log in or otherwise establish a session. In other words, in a stateless connection model it does not matter where the request came from or how the client found its way there. All the requests are treated in the same way.

Unfortunately, a stateless connection does not allow a server to gather information about the client it is responding to. For instance, in the context of a password-protected session, a Web server cannot establish if a client has permission to access restricted information; in the context of a shopping cart for an online store, the Web server cannot know whether the client has selected items to purchase; in the context of an online newspaper that allows users to specify which types of articles they are most interested in, the Web server cannot establish if the client has previously specified preferences.

Several solutions were constructed in order to solve this problem. Basic authentication, for instance, enables password protection by utilizing the HTTP header for the client to send additional information to the Web server. Various Common Gateway Interface (CGI) scripts insert hidden information into a form created on the fly, so that the form will return information specific to the particular client's request. Several disadvantages exist for these approaches. For example basic authentication is relatively easy to spoof, CGI scripts can get confused if the user on the client machine presses the browser's **Back** button, etc.

A solution to this problem would be for the Web server to store the information gathered from all the clients on some permanent storage. However this solution was considered too expensive if implemented on the server-side, especially if a Web site is contacted by a large number of users.

Cookies were developed to establish state by allowing the client and the server to share exchanged information about each other. The first time a client visits a Web site that serves cookies, the Web server sends a cookie to the client along with some information about which URLs the cookie is valid for. The next time the client visits one of those URLs, it knows that it has to include the current value of the cookie in its request. This enables the server to possibly update the value of the cookie and customize its response to the client. Since the cookie is stored on the client machine, this solves, on the server-side, the problem of a too expensive storage of information coming from multiple users. The cookie sent on the client is information that allows the client and the server to establish a session. Notice that this description is a simplification, since for example it is sometimes a JavaScript file, running on the client machine itself, that sets the cookie on the client. In this case, all the activity occurs on the client machine and the JavaScript file acts like a proxy for the server.

There are several types of cookies. The first type was introduced by Netscape with Version 2.0 of Navigator. This type of cookie is currently the most used. A slightly different cookie is described by D. Kristol and L. Montulli in the Request For Comments RFC2109 *HTTP State Management Mechanism*, that you can find at http://info.internet.isi.edu/in-notes/rfc/files/rfc2109.txt. The basic process is the same in each case: the server will set the cookie, send it to the client and then look for it the next time the client issues a request to the same Web server. As we mentioned, the same job done by the server can be done by a client-side script on behalf of the server.

So, what can the security exposures be for a client when it receives a cookie from a Web server? Cookies are pieces of information that the Web server stores in a text file, named cookies.txt, that you can easily find under the browser directory. In our Windows NT system, where both Netscape Communicator and Microsoft Internet Explorer were installed, we found two files, both named cookies.txt, stored below the Netscape and Internet Explorer root directories respectively. Cookies are not programs that can cause the client machine to shut down, erase its hard disk, corrupt its files or throw a virus. Furthermore, because a cookie is set by a Web server or by a client-side script on behalf of a Web server, it doesn't contain any information that the server (or the client-side script) does not already know or have access to. So cookies are simply used to save some piece of information for later retrieval. Of course, as we mentioned at the beginning of this section, what we are saying now about the security implications of cookies is based upon current information. It could be possible that bugs discovered in a browser, in the future, can create problems. For example a concern about cookies was that Netscape Navigator 2.0 allowed JavaScript to store a user's e-mail address in a cookie, which could then be sent to a server. This was really a security exposure, since the rule that cookies do not have to provide a server with any information about the client that the server does not already know was bypassed. A Web server usually does not know the e-mail address of the user on the client machine. This loophole enabled the Web server to know something new about you and for example your e-mail address could then be added to e-mail distribution lists you never asked to belong to. However, this particular loophole was present in JavaScript and not in the cookie mechanism, and it was closed with Netscape 2.01. However some books still recommend to remove personal information, like your name and your e-mail address, from within your Web browser before browsing the Web. If you use your browser to access your e-mail, then you should re-enter that information into your browser before you try to send or receive e-mail.

Problems like the one shown above have been solved, and now a cookie can only contain the following:

- Standard information from the HTTP header, such as your IP address (that the server already knows, since the Web server needs it to direct its response back to the client), browser type and version and page previously visited.
- Information generated by the server or client-side script, such as a unique session ID.
- Any additional information you have supplied in response to questions or forms, such as items already ordered from an online shop.

Notice that all this information the server could construct with or without cookies. Passing cookies is simply more easy, fast and convenient than maintaining and analyzing large server logs. The convenience is for Web server administrators, who can easily and quickly retrieve information about you and construct a customized Web page based upon the information they retrieve. There is a convenience for you as well, since next time you visit the same Web site that, for example, asks you to register, you will not have to re-enter the same data: the Web server simply retrieves the cookie it set on your computer and you are automatically recognized. This happens for example with Microsoft's Web site, http://www.microsoft.com. Before downloading code, you must register. Microsoft stores user information in a cookie on the client machine. That information is then retrieved the next time the user accesses their Web site.

Cookies offer another big advantage, because for multi-user machines, where a number of people share the same IP address, cookies enable the server to distinguish between the various browsers using that address, since each browser maintains its own table of cookies. The same operation would be really more complicated without the cookie support.

Most of the current concerns about cookies are related to an intrusion of privacy that cookies seem to allow, since some advertising groups have started constructing user activity profiles using cookies. On the basis of the data entered by a user on a form the previous times the user visited a specific Web site, these groups can quickly construct the user profile and send to the user customized Web pages. For example, whenever you are at the AltaVista Web site http://www.altavista.com and receive a cookie, it is probable that one of these advertising groups is at work.

As mentioned above, however, all of this can be done anyway, with or without cookies. The only difference is that with cookies this operation becomes easier and faster.

We can conclude that cookies present a privacy exposure more than a security risk. In order to prevent problems we recommend the following:

- Keep your browser version up to date. This is a general rule that can be repeated in particular for cookie-related problems. Java and JavaScript, for example, are evolving very fast and the leading browser manufacturers solve step-by-step all the problems that arise.
- If you are concerned about your private information being distributed, do not worry about cookies, worry about what information you provide in response to forms or questions. Cookies will contain only that information that the server already knows, so it's up to you not to provide untrusted Web servers with private information.
- Rather than setting your browser to accept all cookies (which could be a privacy exposure) or deny all cookies (which could cause you to re-enter several times the same information, filling the same forms, etc.), you could configure it to alert you before accepting a cookie. Notice that by default Netscape Navigator enables all cookies. We showed how to modify the settings for cookies in Figure 179 on page 182 for Netscape Navigator and Figure 201 on page 205 for Microsoft Internet Explorer.

  Of course it can become very annoying to receive so many warning messages coming from all the Web sites that would like to set a cookie on your system and ask your permission to do that. Since so many Web sites require that you accept a cookie to access certain services, programs exist that can perform filtering for you, allowing you to specify exactly what servers you will receive cookies from. See for example Cookie Jar at http://www.lne.com/ericm/cookie_jar.
- A privacy exposure for you is that if a Web site sends a cookie to your computer, you cannot deny to have visited that Web site. The file cookies.txt can be opened with a regular text editor. Although it carries a warning not to edit it directly, you can safely delete entire lines and then save it if you see any

you do not want to keep. If you are using Netscape Communicator, just remember to exit all the copies of Communicator currently running before saving that file, or Communicator will overwrite the cookies.txt file when it shuts down, deleting all your modifications.

If you edit and modify a single line of the cookies.txt file, but you do not remove it completely, you may cause errors when you try to access Web sites that use that cookie. It is interesting to know that if you delete the cookies.txt file, both Netscape Communicator and Microsoft Internet Explorer will create a new copy of the file on their next execution.

If privacy is your concern, then you should remember that not only the file cookies.txt can keep track of all the Web sites you visited. It is well known, for example, that to retrieve the sites you have most recently visited, it is enough to enter

```
about:cache
```

in the Location box in Netscape Navigator. A new page will appear, showing all the Web pages you have recently visited. If you visited Web sites containing private information that you do not want to share with other people who use your same machine, you should clear memory cache and disk cache before leaving your desk or you can also set their values to `0 KBytes`. This operation can be done through the Advanced Preferences window, as shown in Figure 247 on page 256. You can also change the values for memory cache and disk cache by manually editing the prefs.js file (see 5.6, “Netscape Navigator Hidden Security Preferences”).

5.6 Netscape Navigator Hidden Security Preferences

Netscape Navigator is the browser part of Netscape Communicator. We have shown in 5.1.1, “Netscape Navigator” on page 181 how security-related configuration information can be found in two places in Navigator. Some items are found in the Preferences settings and some in the Security Information panels. We present now a description of the so-called *hidden* security preferences, this name coming from the fact that such security preferences cannot be configured through any GUI.

For each user running Communicator, a local preferences file, named prefs.js, is automatically installed in the directory *Netscape_root*\Users*user_name*. This file is a plain-text record of the user's current preferences.

The following figure shows the file prefs.js that we found under the directory D:\Program Files\Netscape\Users\jsmith after having installed Netscape Communicator and configured the profile for the new user John Smith:

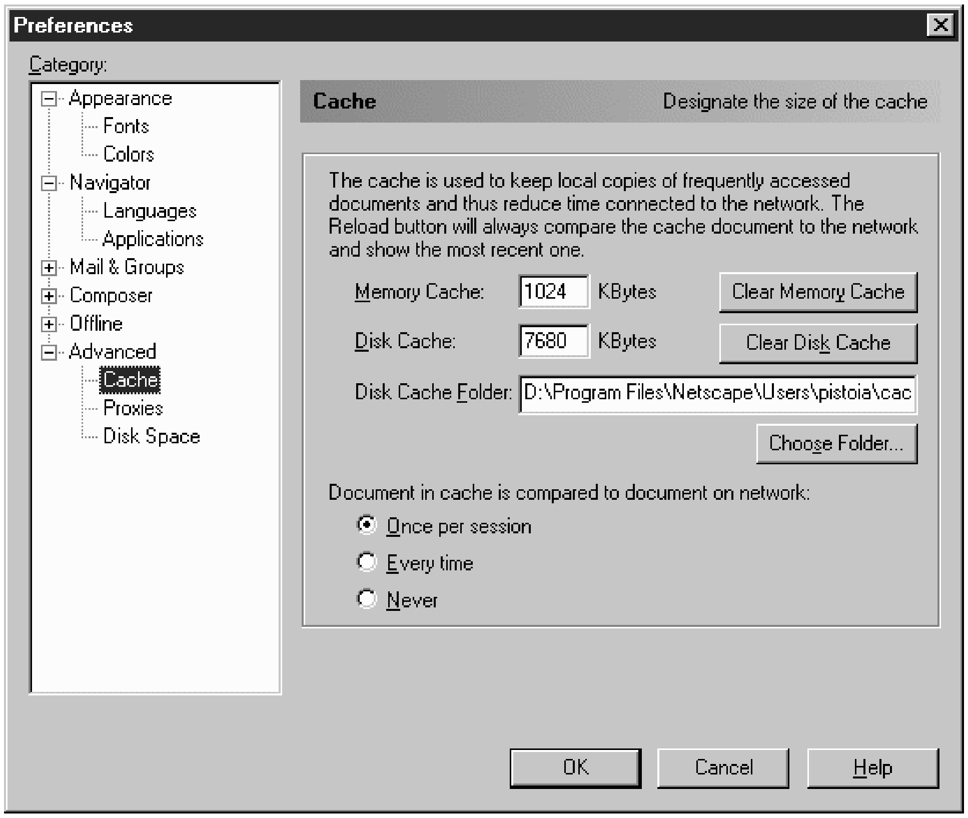
```
// Netscape User Preferences
// This is a generated file!  Do not edit.

user_pref("browser.startup.homepage_override", false);
user_pref("browser.window_rect", "0,0,632,480");
user_pref("custtoolbar.personal_toolbar_folder", "Personal Toolbar Folder");
user_pref("editor.author", "John Smith");
user_pref("ldapList.version", 1);
user_pref("ldap_1.directory1.filename", "abook.nab");
user_pref("ldap_1.directory2.filename", "X0HCMB9M.nab");
user_pref("ldap_1.directory3.filename", "XV496L8B.nab");
user_pref("ldap_1.directory3.searchBase", "c=US");
user_pref("ldap_1.directory4.filename", "X044F5OH.nab");
user_pref("ldap_1.directory5.filename", "X0LC7LJ0.nab");
user_pref("ldap_1.directory6.filename", "XVS29DR1.nab");
user_pref("ldap_1.end_of_directories", "8778368");
user_pref("mail.identity.useremail", "jsmith@company.com");
user_pref("mail.identity.username", "John Smith");
user_pref("mail.pop_name", "jsmith");
user_pref("news.show_pretty_names", true);
user_pref("taskbar.floating", false);
user_pref("taskbar.x", 634);
user_pref("taskbar.y", 8);
```

Figure 246. prefs.js Preferences File for the User John Smith

Notice, however, that this figure shows the prefs.js file as soon as Communicator has been installed and John Smith has defined himself as a Communicator user. Once John Smith starts to configure his Communicator, this file becomes pretty huge. For example it will contain the names or IP addresses of mail, discussion or proxy servers and their secure port numbers. To an intruder these preferences could offer information on both the client itself and the network. Moreover, if an intruder were able to access this file, he could disable password protection or automatically copy sent mail to an accessible location on the network. For this reason, the prefs.js file should be kept protected in all the ways offered by the operating system where Communicator has been installed.

Each preference that the user sets through the GUI of Communicator is automatically registered in the corresponding user's prefs.js file. This is the reason why in the first line we can read that it is a generated file. The hidden security preferences, however, since for them no GUI was built into Communicator, can be configured *only* through the prefs.js file, even if Netscape recommends that we shouldn't edit that file directly.

For example, some network administrators consider it a security matter to make all users in the network disable memory cache and disk cache, since cached pages could contain copies of documents recently viewed on secure sites (such as, for example, Payroll or Human Resources). Memory cache and disk cache can be disabled through the Advanced Preferences window provided by Communicator:


Preferences

Category:

Appearance
Fonts
Colors
Navigator
Languages
Applications
Mail & Groups
Composer
Offline
Advanced
Cache
Proxies
Disk Space

Cache Designate the size of the cache

The cache is used to keep local copies of frequently accessed documents and thus reduce time connected to the network. The Reload button will always compare the cache document to the network and show the most recent one.

Memory Cache: 1024 KBytes Clear Memory Cache

Disk Cache: 7680 KBytes Clear Disk Cache

Disk Cache Folder: D:\Program Files\Netscape\Users\pistoia\cac

Choose Folder...

Document in cache is compared to document on network:

● Once per session
○ Every time
○ Never

OK Cancel Help

Figure 247. Netscape Advanced Preferences Window - Cache Settings

Default values are `1024 KBytes` for Memory Cache and `7680 KBytes` for Disk Cache and to disable them the user must set both those values to `0 KBytes`. As soon as the default values are changed, the following two entries immediately appear in the prefs.js file:

```
user_pref("browser.cache.disk_cache_size", 0);
user_pref("browser.cache.memory_cache_size", 0);
```

Most of the preferences that you can set through the prefs.js file have no interface built into Communicator, since they are not considered useful for a common end user. But in reality there are cases where they are useful, and we also needed to experiment with one of these hidden properties when we tried to develop an applet communicating with its servicing Web server through a firewall.

The following list shows the preferences that Java developers should know when they test their applications:

1. `signed.applets.local_classes_have_30_powers`

   Setting this preference to `true` simulates Navigator 3.0 security behavior for local classes. This preference does not affect downloaded classes. Moreover it does not provide local classes with file access privileges, since such privileges are denied on Netscape Navigator 3.0. But, if that preference is enabled, classes on the CLASSPATH can establish universal network connections and load digital libraries.

2. `signed.applets.codebase_principal_support`

Netscape security model for Java considers a *principal* as the originator of a particular class. By default, Communicator only accepts principals that are based on cryptographic certificates or that are based on `file:///` URLs. If you set this preference to `true`, an `http://` codebase (which is the URL where the class file is located) will be allowed to be a principal, meaning that the signing stage when developing secure code is bypassed.

3. `signed.applets.verbose_security_exception`

   When it is set to `true`, this preference allows the method printStackTrace() in class java.lang.Throwable to print a complete stack trace, and not only the name of the exception. But in order for this preference to work properly, also you need to disable the Just In Time (JIT) compiler. To do this, you should rename the JIT library file Program\Java\Bin\jit3240.dll to a different name, then shut down all the copies of Communicator already active on your platform. Since JIT when it is active displays a message on the Java Console, next time you start Communicator you could verify that such a message is no longer present. If it turns out that JIT is still active, you should restart your computer.

4. `signed.applets.simulate_signature_on_system_classes`

   This preference must be used very carefully. If it set to `true` it permits the system classes shipped with Communicator to be modified without a valid signature is placed on the modified classes.

5. `security.lower_java_network_security_by_trusting_proxies`

   When an applet tries to connect back to its servicing Web server, the JVM automatically passes the URL request to the regular browser connection for processing. For this reason, even if a proxy server is defined in the Communicator proxy configuration, the Java code will automatically use it. Of course Communicator's SecurityManager only permits a running applet to connects to its originating Web server and on the same port. For this reason, if the Web server is invoked through its host name, the SecurityManager first of all translates the host name of the Web server to its IP address. If this check were not performed, it would be possible for an intruder to attack your system associating the host name with a false IP address. There are cases, anyway, where you do not want that the DNS look-up is done, even if such a choice can reduce security. For example, if your Web browser is located behind a firewall, and DNS host name resolution is not provided for names of hosts outside the firewall, the applet will fail to make the desired connection to any Web server, even to its servicing one, if this is located outside the firewall. Setting this preference to `true`, the SecurityManager allows the applet to establish its connection, even if DNS host name resolution is not established for names of hosts located in the Internet. This is a hidden property that sometimes can be considered very useful also for end users, if they are located inside an intranet.

Be sure to edit the preferences file only when Netscape Communicator is not running, because Communicator, when it is shut down, overwrites that file and you would lose all the modifications you have done.

5.7 Applet/Server Communication Through a Firewall

In this section we will show you how an end user can apply some of the hidden security preferences that we have introduced in the last section.

We wrote a simple Java applet that, when it is downloaded in the Web browser, tries to establish two connections to its servicing Web server. The communication between the client and the server is controlled by a firewall. We experimented this communication when the firewall had been configured using the following firewall technologies:

1. IP filters for HTTP (see 6.1.1, “IP Filters” on page 270)
2. HTTP proxy server (see 6.1.3, “Proxy Servers” on page 280)
3. SOCKS server (see 6.1.4, “SOCKS Servers” on page 284)

Java applets within an HTML page are transferred using the HTTP protocol, the same used to transfer the HTML page itself. The HTML page, as we are going to see, must contain a tag `<APPLET>` to call an applet. This can specify the applet class name as the value of the Code attribute. It is also possible and advisable, especially when many Java classes are involved, to pack all the classes into a compressed Java Archive (JAR) file. This file is then specified as the value of the Archive attribute.

If the security policy for the firewall is to permit HTTP traffic to flow through the firewall, then Java applets are permitted to reach the client machine, provided it accepts Java. In fact, in this case, Java applets are treated as all the other components of the Web pages.

However, one of the major problems that people have encountered with applets and firewalls is trying to get applets to communicate back to the server through a firewall. In this section we want to show you the difficulties that the applet finds and what the possible solutions are.

The following figure shows the source code of the applet we wrote. We named it AppletConnection.

```
import java.awt.*;
import java.net.*;

public class AppletConnection extends java.applet.Applet
{
   int javaSize, classSize;

   public void init()
   {
      try
      {
         URL javaURL = new URL(getCodeBase(), getClass.getName() + ".java");
         System.out.println(javaURL);
         URL classURL = new URL(getCodeBase(), getClass.getName() + ".class");
         System.out.println(classURL);
         URLConnection javaURLConnection = javaURL.openConnection();
         URLConnection classURLConnection = classURL.openConnection();
         javaSize = javaURLConnection.getContentLength();
         classSize = classURLConnection.getContentLength();
      }

      catch(Exception ex)
      {
         ex.printStackTrace();
      }

   }

   public void paint(Graphics g)
   {
      Font myFont = new Font("SansSerif", 3, 15);
      g.setFont(myFont);
      g.setColor(Color.black);

      if (javaSize == -1)
         g.drawString("I am sorry, I could not find " + getClass().getName() + ".java", 30, 50);
      else
         g.drawString(getClass().getName() + ".java is " + javaSize + " bytes long.", 30, 50);

      if (classSize == -1)
         g.drawString("I am sorry, I could not find " + getClass().getName() + ".class", 30, 80);
      else
         g.drawString(getClass().getName() + ".class is " + classSize + " bytes long.", 30, 80);
   }
}
```

Figure 248. AppletConnection.java

We saved this file in a directory named TEST, which we created below the public directory of the Web server. In other words, we stored it in D:\WWW\HTML\TEST. In the same directory, we stored also the class file of this applet, that we obtained after compiling it entering the command:

```
javac AppletConnection.java
```

The purpose of this applet, after it has been downloaded onto the client's browser, is to establish a URL connection with the Java file containing its source code and another URL connection with the class file of the applet itself. Notice that both these files are stored in the Web server. After these two connections have been established, the applet displays on the client's browser the size of each file.

Now, let's read together the source code of this applet, so that we can understand what it does.

First of all, we notice that this applet imports two packages:

1. java.awt, necessary in order for the applet to use the Graphics class in the paint() method.
2. java.net, necessary in order for the applet to use the URL and URLConnection classes in the init() method and to establish its connections to the Web server that downloaded it.

The URL and URLConnection classes are used twice in the init() method. In fact this applet tries to establish two URL connections with its servicing Web server:

1. The first connection is with the file that stores the Java source code of the applet itself. We created a URL object, named javaURL, that should be initialized to http://*ourWebServer*/TEST/AppletConnection.java.
2. The second connection is with the file that stores the Java class of the applet. We created for this reason a second URL object, named classURL, that should be initialized to http://*ourWebServer*/TEST/AppletConnection.class.

Note that *ourWebServer* can be either the host name or the IP address of the Web server servicing the applet, depending on whether the client browser invokes the Web server through its host name or through its IP address.

We then create two URLConnections objects, named javaURLConnection and classURLConnection, and we invoke the getContentLength() method on these objects to retrieve the size of each file. The exact number of bytes is then displayed on the client's browser through the paint() method. Notice that the method getContentLength() returns the content length of the resource that this connection's URL references, or returns `-1` if the content length is not known. In our applet, the problem of an unknown content length could be generated by the absence of the files in the directory D:\WWW\HTML\TEST where the applet looks for the files through the two URL objects created. If the applet does not find the files, the applet will display an error message on the client's browser.

The following figure shows the code for the HTML page through which the applet is invoked:

```
<HTML>

<HEAD>
<TITLE>Applet Connection</TITLE>
</HEAD>

<BODY>

<H3>AppletConnection</H3>

<APPLET Code="AppletConnection" Width=500 Height=300>
<H4>This area contains a Java applet, but your browser is not Java-enabled.</H4>
</APPLET>

</BODY>
</HTML>
```

Figure 249. AppletConnection.html

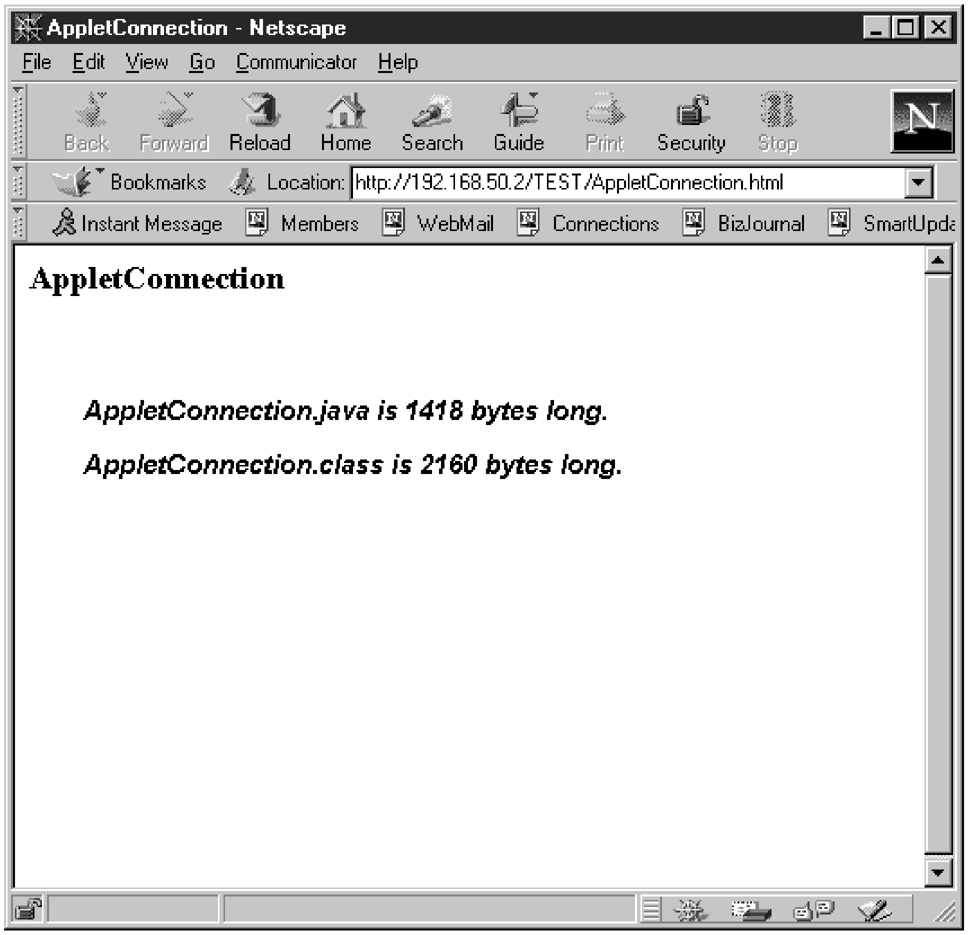
We named this HTML file AppletConnection.html and we stored it in the same directory D:\WWW\HTML\TEST where we had also saved AppletConnection.java and AppletConnection.class.

First of all we tested this applet in a normal client/server scenario, where we had also installed a Domain Name Service. The configuration of the two machines involved is shown in the following table:

Table 4. Client/Server Scenario Environment

	Client - Netscape Communicator 4.05	**Server - Domino Go Webserver 4.6.2.2**
IP Address	192.168.51.2	192.168.50.2
Host Name	wtr05366	wtr05218

To invoke the HTML page AppletConnectio.html, it was therefore possible to enter the URL specifying either the IP address 192.168.50.2 or the host name wtr05218 of the Web server machine. We started by pointing the browser to the URL http://192.168.50.2/TEST/AppletConnection.html and the result is shown in the following figure:


AppletConnection - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://192.168.50.2/TEST/AppletConnection.html
Instant Message Members WebMail Connections BizJournal SmartUpda

AppletConnection

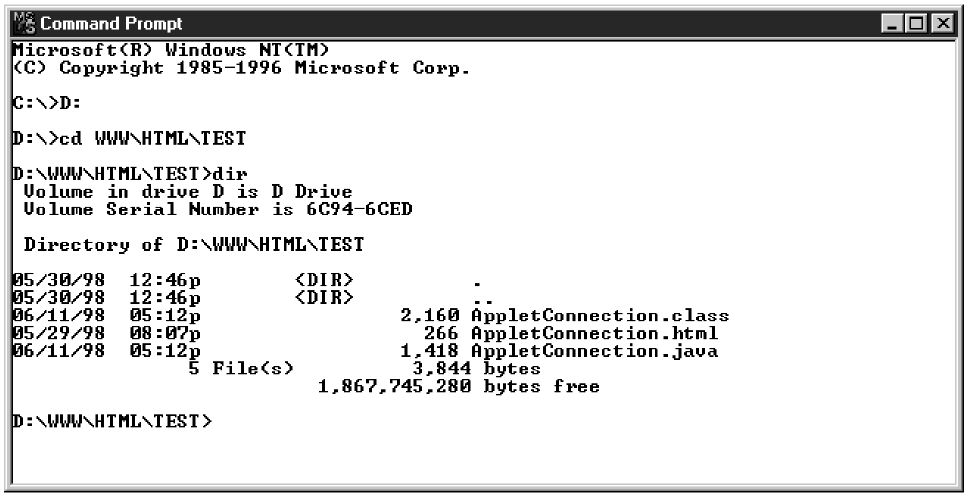
AppletConnection.java is 1418 bytes long.

AppletConnection.class is 2160 bytes long.

Figure 250. Applet Running on the Client Machine

The we tried again, but this time we specified the host name of the Web server machine. So we pointed our browser to the URL http://wtr05218/TEST/AppletConnection.html and the result was the same as shown in Figure 250.

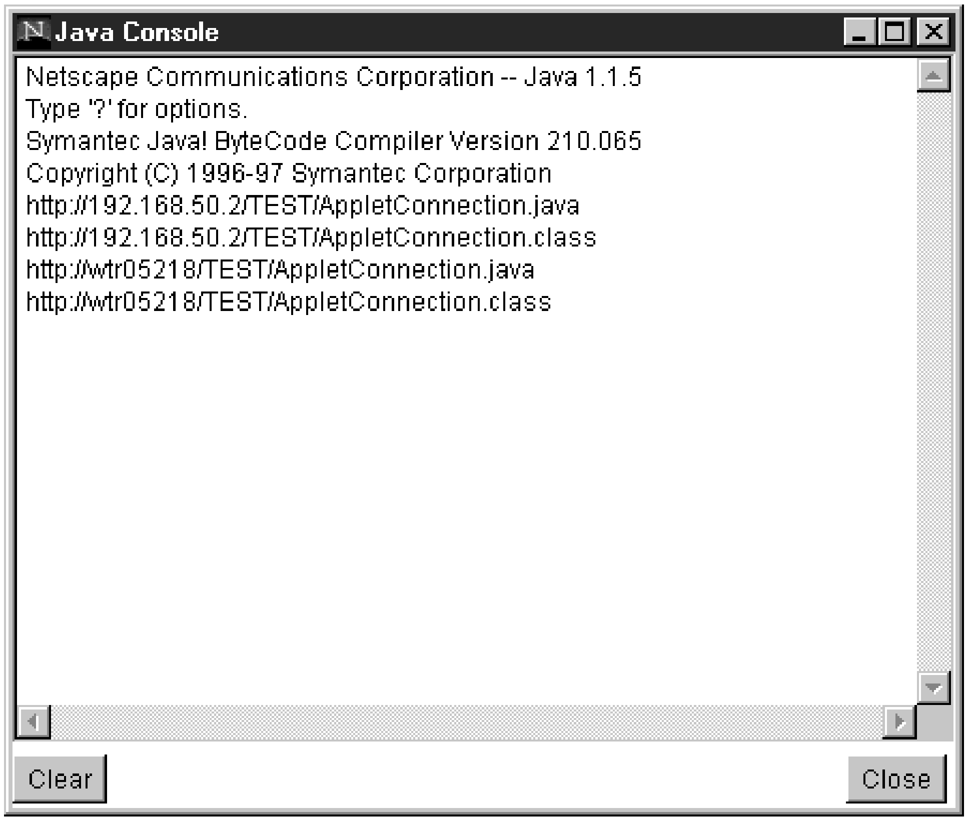
Notice that the size of the two files was displayed correctly, since AppletConnection.java was really 1418 bytes long and AppletConnection.class was really 2160 bytes long, as the following screen demonstrates:



```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>D:

D:\>cd WWW\HTML\TEST

D:\WWW\HTML\TEST>dir
 Volume in drive D is D Drive
 Volume Serial Number is 6C94-6CED

 Directory of D:\WWW\HTML\TEST

05/30/98  12:46p        <DIR>          .
05/30/98  12:46p        <DIR>          ..
06/11/98  05:12p                 2,160 AppletConnection.class
05/29/98  08:07p                   266 AppletConnection.html
06/11/98  05:12p                 1,418 AppletConnection.java
               5 File(s)          3,844 bytes
                          1,867,745,280 bytes free

D:\WWW\HTML\TEST>
```

Figure 251. Contents of the Directory D:\WWW\HTML\TEST of the Web Server

We wanted to open the Java Console of the Netscape Navigator browser. If you read again the source code of the applet AppletConnection, you will see that there are two calls to the System.out.println() method, and the two URL objects javaURL and classURL are passed as parameters. Notice that the println() method automatically converts an Object to a String, using the toString() method. This is what the Java Console registered:

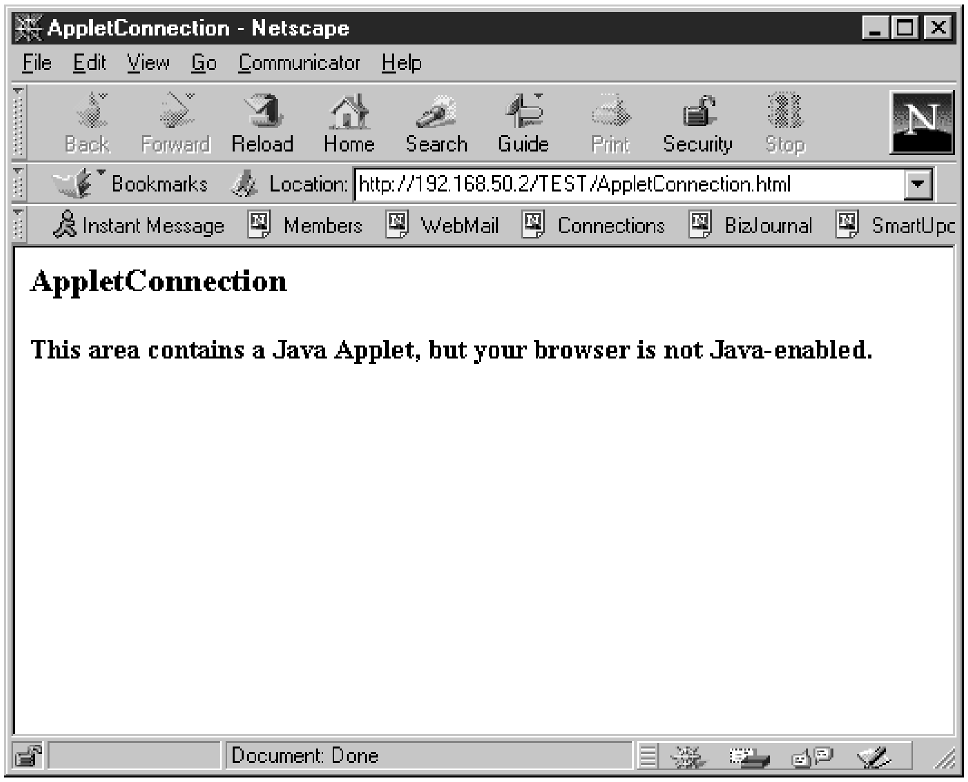


```
Netscape Communications Corporation -- Java 1.1.5
Type '?' for options.
Symantec Java! ByteCode Compiler Version 210.065
Copyright (C) 1996-97 Symantec Corporation
http://192.168.50.2/TEST/AppletConnection.java
http://192.168.50.2/TEST/AppletConnection.class
http://wtr05218/TEST/AppletConnection.java
http://wtr05218/TEST/AppletConnection.class
```

Figure 252. On Opening the Java Console

As you can see, the getCodeBase() method of the applet returns the IP address of the Web server if you invoked the HTML page through the Web server IP address. It returns the host name if you invoked the HTML page through the Web server host name.

Notice that the browser will display an error message if it was configured to refuse Java applets. In other words, if the **Enable Java** check box is not marked (see Figure 179 on page 182), the following error message appears in the client's browser when the HTML page is downloaded:

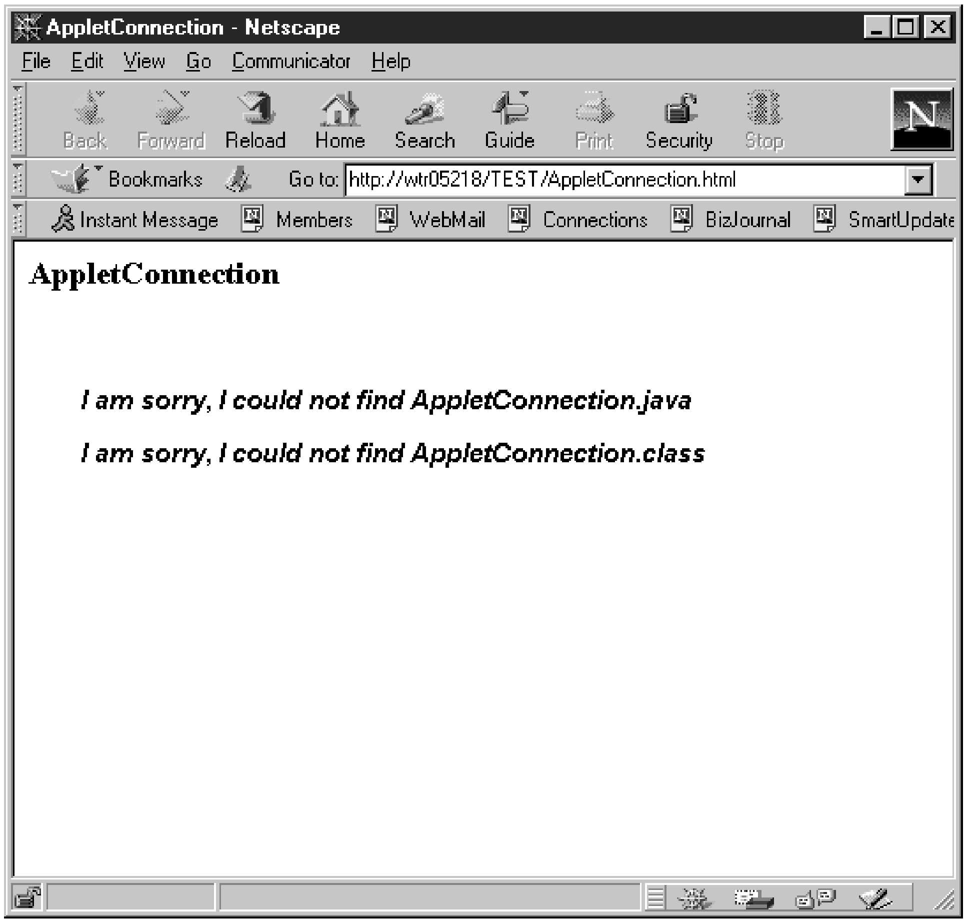

AppletConnection - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://192.168.50.2/TEST/AppletConnection.html
Instant Message Members WebMail Connections BizJournal SmartUpc
Document: Done

AppletConnection

This area contains a Java Applet, but your browser is not Java-enabled.

Figure 253. Error Message Displayed if the Browser Is Not Java-Enabled

We also verified that, if we removed the Java and the class files from the TEST directory, the applet displayed the error message that we had programmed.


AppletConnection - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Go to: http://wtr05218/TEST/AppletConnection.html
Instant Message Members WebMail Connections BizJournal SmartUpdate

AppletConnection

I am sorry, I could not find AppletConnection.java

I am sorry, I could not find AppletConnection.class

Figure 254. On Running the Applet When the Two Files Have Been Removed

Notice, however, that the class file must be in the TEST directory; otherwise, it is not loaded. So, in order to see that error message, we had to remove the class file from the TEST directory *after* downloading the applet and reloading the AppletConnection.html Web page.

So, everything worked as we had figured out. We got the same successful result using a firewall implementing IP filters for HTTP (see 8.5.3, “IP Filters Configuration for HTTP and SSL” on page 416) in the same client/server environment described in Figure 406 on page 435. The reason is that IP filters for HTTP simply permit the HTTP protocol to flow between the client and the server, allowing a TCP/IP connection between a TCP port greater than 1023 on the client and the HTTP port 80 on the Web server. The client and the server can speak to each other directly through the filter, since the firewall routes the traffic, but does not act on behalf of the client.

After this, we wanted to test our applet when the communication between the client and the server was controlled by a firewall acting as an HTTP proxy server.

We found problems only if in our network environment no Domain Name Service had been configured to translate host names to IP addresses for hosts located in the non-secure network (see 9.5.2, “Domain Name Service in a Firewall Protected Network Environment” on page 445), or if the firewall had been configured to disable DNS queries (see Figure 417 on page 450).

This is a common situation, because the DNS is often configured to only translate host names of hosts behind the firewall. What happens is that, if you invoke the HTML page using the host name of the Web server, the checkConnect() method in the class SecurityManager implemented by the Web browser tries to translate the host name to an IP address, but behind the firewall this operation fails if the DNS has not been configured. An unexpected SecurityException is thrown and your Web browser displays an error message.

This problem is not present if you point your browser to the HTML file specifying the IP address of the Web server, rather than its host name. However, a SecurityException is thrown also if you use the IP address in the URL, but the applet developer hardcoded the Web server host name in the Java applet file, something like:

```
URL javaURL = new URL("http://wtr05218/TEST/AppletConnection.java");
URL classURL = new URL("http://wtr05218/TEST/AppletConnection.class");
```

Notice, however, that this solution is certainly less portable than the one we adopted in our code using the getCodeBase() method.

Now, how can you solve this problem and have your applet running in your Web browser? It is very simple, even if this solution affects the security of your system. You should shut down all the copies of Communicator currently running on your client machine, edit the prefs.js file (see Figure 246 on page 255) and add the line:

```
user_pref("security.lower_java_network_security_by_trusting_proxies", true);
```

With this preference enabled (see 5 on page 257), if the DNS lookup performed by the Communicator's SecurityManager fails, then the host name of the Web server is relied upon, rather than having a stricter DNS/IP address equivalence.

Notice that this is one of the hidden security preferences which end users in some intranets may need to enable. However, it impacts the security of the system, since it disables the DNS lookup performed by the SecurityManager to prevent spoofing attacks.

To disable that security preference, you can, after shutting down Communicator, either set it to `false` or completely remove that entry from the prefs.js file.

The situation was very similar when the firewall implemented a SOCKS server and the applet was able to connect its servicing Web server if a DNS in our network environment had been configured to translate host names into IP addresses for hosts located outside the firewall.

We made an experiment also with an applet trying to implement a connection to its servicing Web server via the java.net.Socket class. What we saw was that the applet was able to establish a successful connection only without the firewall or if the firewall simply routed the IP traffic, through appropriate IP filters for HTTP. But the connection failed if the firewall was implementing an HTTP proxy server or a SOCKS server. The problem for HTTP proxy servers is that support for proxy is part of the protocol that you are using over HTTP (in this case, HTTP). It is therefore not possible to encapsulate the proxy-specific stuff at the socket layer. Instead, with the way SOCKS works, it should be possible to put SOCKS support in the java.net.Socket code, resulting in an encapsulating of the SOCKS protocol layer and allowing the enforcement of the security policy without undue negative impact on applications running behind the firewall. This has been done in JDK 1.0.2, but unfortunately not in Netscape. So, if your network environment has a SOCKS server, then everything works fine as long as you use the Applet Viewer or java.net.URLConnection, but using java.net.Socket under Netscape with a SOCKS server between the client and the server will give you a SecurityException.

Notice that on the OS/2 operating system (at least Warp 4) the basic TCP/IP stack already contains support for SOCKS. In other words, the TCP/IP stack is SOCKSified (see 6.1.4, "SOCKS Servers" on page 284). Therefore, on this operating system, you should be able to use java.net.Socket under Netscape even if you are behind a SOCKS server.

The following table summarizes our conclusions with the three types of firewall technologies we worked on (IP filters for HTTP, HTTP proxy server and SOCKS server) and the two Java classes we used to permit the network connection between the applet and the Web server (java.net.URLConnection and java.net.Socket):

Table 5. Firewall Techologies and Java Classes for Network Connections

	java.net.URLConnection	**java.net.Socket**
IP Filters for HTTP	Connection permitted	Connection permitted
HTTP Proxy Server	Connection permitted	Connection denied
SOCKS Server	Connection permitted	• Netscape - Connection denied (except under OS/2) • Applet Viewer - Connection permitted

Chapter 6. Firewall Security in an NCF Environment

IBM Network Computing Framework supports several security measures, like user identification and authentication, access controls to resources and services, confidentiality, integrity, security management, key recovery, Java security and firewalls. This chapter provides a general description of firewall technologies and architectures that are commonly used in an NCF environment. Examples of how these firewall technologies can be implemented are shown in Chapter 8, “Three-Tier Applications in Firewall-Protected Network Environments” on page 361, Chapter 9, “IIOP in Firewall-Protected Network Environments” on page 435 and Chapter 10, “SSL Tunneling and SOCKS Server for HTTPS Scenarios” on page 487. For further information, see the IBM redbooks *Protect and Survive Using IBM Firewall 3.1 for AIX*, SG24-2577, and *A Comprehensive Guide to Virtual Private Networks, Vol. I*, SG24-5201.

We would like to mention that some of the examples of the use of IBM eNetwork Firewall Version 3.2 that we will present in this chapter and in Chapter 8, “Three-Tier Applications in Firewall-Protected Network Environments” on page 361, Chapter 9, “IIOP in Firewall-Protected Network Environments” on page 435 and Chapter 10, “SSL Tunneling and SOCKS Server for HTTPS Scenarios” on page 487 are based upon the AIX version of this product. Things could be slightly different on a Windows NT platform.

For example, the NT version of IBM eNetwork Firewall 3.2 supports SOCKS V5, while the AIX version still supports only SOCKS V4. The HTTP proxy server implemented on the NT version is based upon HTTP 1.1 and has user authentication, while the HTTP proxy server implemented on the AIX version is based upon HTTP 1.0 and does not have user authentication. Also, the Windows NT version has Network Address Translation (NAT) only starting from the recently announced 3.2.1.1 version, while the AIX version has NAT already in the 3.2 version.

6.1 Understanding and Using Firewall Technologies

This section will describe the most important firewall technologies that are commonly used in an NCF environment, like IP filters, proxy servers, SOCKS servers and SSL tunneling.

We discuss in Chapter 8, “Three-Tier Applications in Firewall-Protected Network Environments” on page 361 how to use IP filters for HTTP and HTTPS between a client machine and a Web server and also for the DB2 proprietary protocol over TCP/IP between the Web server and a DB2 server. Examples of how to implement proxy servers and SOCKS servers for HTTP, HTTPS and also IIOP over HTTP are shown in Chapter 9, “IIOP in Firewall-Protected Network Environments” on page 435 and Chapter 10, “SSL Tunneling and SOCKS Server for HTTPS Scenarios” on page 487, where you can also find how to implement SSL tunneling.

6.1.1 IP Filters

IP filters are tools to filter packets at the session level.

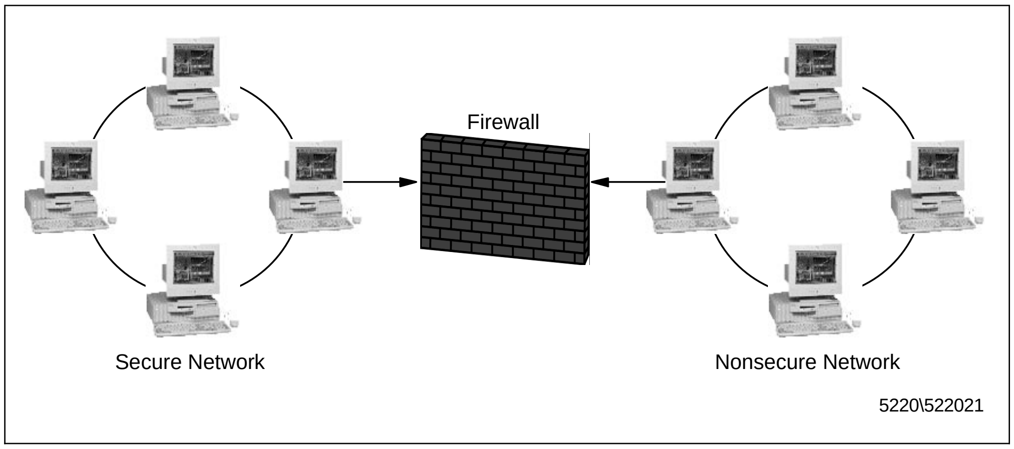


Figure 255. Firewall Implementing IP Filters

The machine where the firewall is installed requires two or more network adapters, each in a separate network or subnetwork. One set of interfaces must be declared non-secure, while the remaining set is declared secure.

Using IP filters, the firewall administrator can permit or deny IP traffic to flow in and out of each adapter, basing their decision upon the following criteria:

- Source and destination IP addresses and masks
- Type of IP protocol
- Source and destination port numbers
- Traffic direction
- Network interface
- Whether the traffic is routed through the firewall or not
- Whether the IP packet is fragmented or not

Notice that when IP filters are used along with other firewall technologies like HTTP proxy or SOCKS server, the firewall grants security to the protected network by performing the necessary work on behalf of the secure user, that in this way is hidden, in that the non-secure network cannot know that the secure user even exists.

If you set up your firewall in order to simply filter the traffic between the secure and the non-secure network, this means that the firewall is not acting on behalf of the secure user. The secure and the non-secure user can speak directly to each other and this can create a security risk. In fact, unless Network Address Translation (NAT) is used, the secure host's IP address will be exposed to the non-secure network. For this reason, firewall implementations completely based upon IP filters are indicated only in those cases where you want to separate, within the same intranet, a low-level security subnetwork from a high-level security subnetwork.

Even if the traffic between the intranet and the Internet is filtered by the firewall, IP filters maintain a direct connection between a secure user and non-secure hosts. A client Web browser in the intranet must be aware of this and for this reason it must be properly configured. Refer to 9.6.2, “Netscape Communicator Advanced Configuration for Using IP Filters” on page 463 to see how to implement this configuration with Netscape Communicator. If your client browser is Microsoft Internet Explorer, it is enough that the check box **Use Proxy Server** in the Internet Properties window (see Figure 264 on page 284) is not marked.

6.1.2 Expert IP Filters Using IBM eNetwork Firewall 3.2

IP filters are the basic way to implement a firewall. Proxy technology needs a proxy server for every application protocol. If you want your proprietary application protocol to pass through the firewall using the proxy technology, you must create a proxy server program for that specific protocol. SOCKS technology requires that the source code of a client program be re-compiled and re-linked. It may happen that, if you do not own the source code of that program, you cannot use the SOCKS technology (unless your client runs a SOCKSified TCP/IP stack, as we will see in 6.1.4, “SOCKS Servers” on page 284). Therefore, IP filter technology is important to implement a firewall.

Building IP filters can be difficult and needs a lot of network skills and experience. There are cases in which it can be very hard to build, maintain, manage and extend IP filters. IBM eNetwork Firewall 3.2 provides an easy way to perform these operations. IP filters in IBM eNetwork Firewall 3.2 consist of four main components:

1. Network objects
2. Rules
3. Services
4. Connections

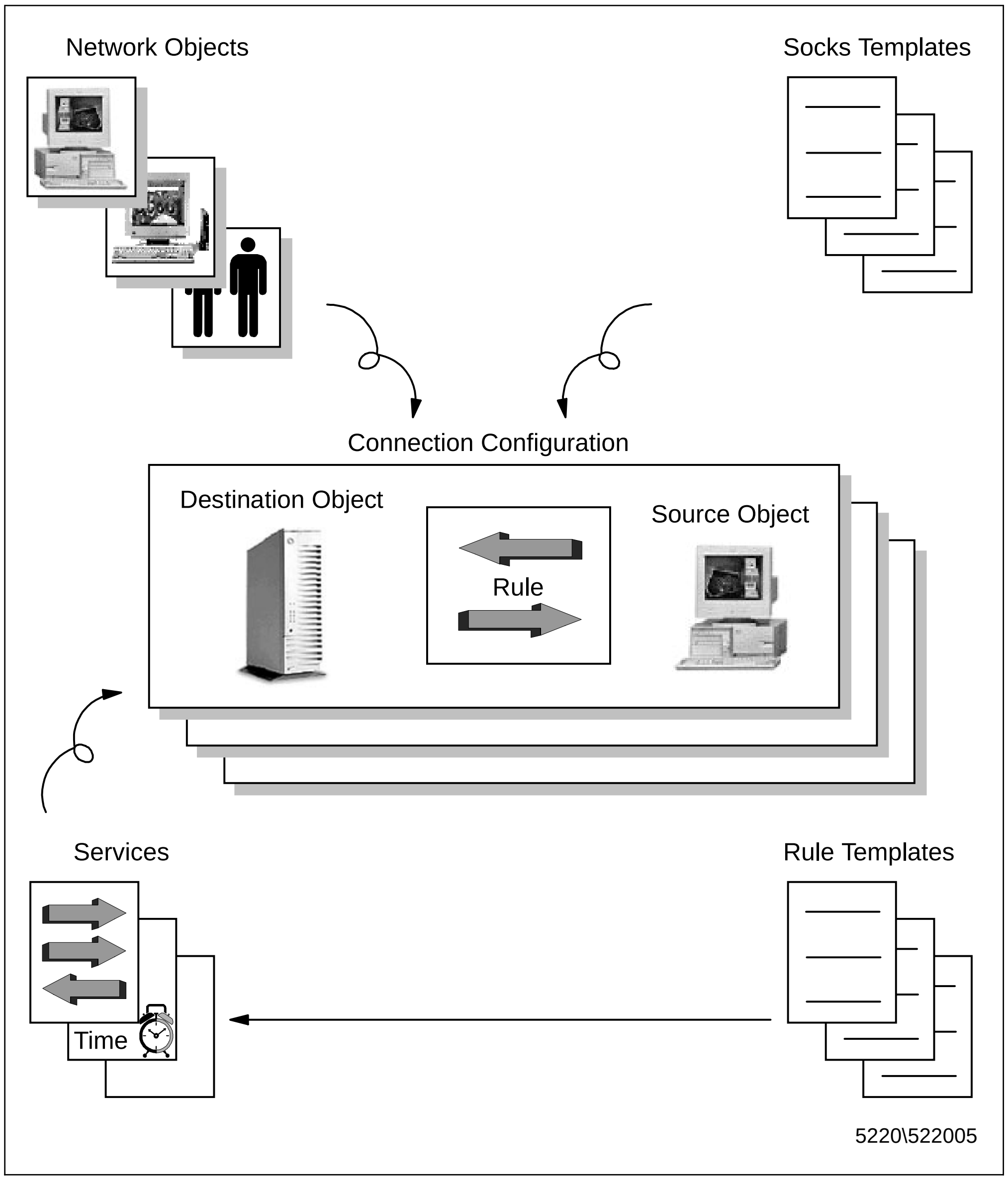


Figure 256. IP Filter Components in IBM eNetwork Firewall 3.2

According to IBM eNetwork Firewall 3.2 architecture design, each *connection* associates network objects and services and establishes if a specific type of IP traffic between a source network object and a destination network object is permitted or denied.

Network objects are representations of entities that already exist in the network, such as hosts, networks, routers, Virtual Private Networks (VPNs) or users. They are used to designate the source and the destination of a service when you create a connection.

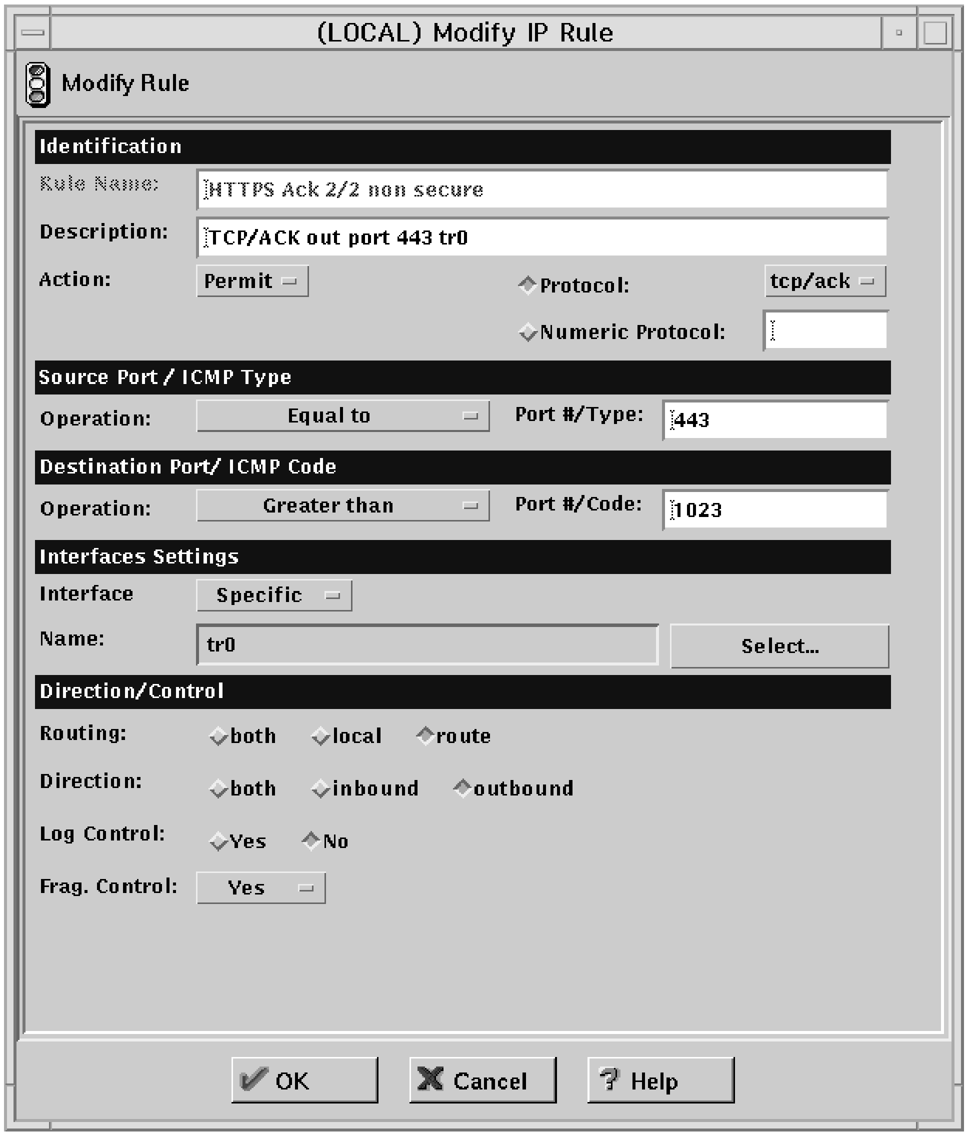
A *service* defines the type of IP traffic that is permitted or denied between two network objects. Notice that, unlike a connection, a service does not distinguish the source network object from the destination network object.

A service is composed of a set of rules. Each *rule* forces the firewall to permit or deny IP packets based upon their specific attributes.

We understand now how rules, services, objects and connections are used to define an IP filter using IBM eNetwork Firewall 3.2. Several examples of how to implement these technologies are shown in Chapter 8, “Three-Tier Applications in Firewall-Protected Network Environments” on page 361 and Chapter 9, “IIOP in Firewall-Protected Network Environments” on page 435. We will see in detail how these technologies can be applied through the IBM Firewall Configuration Client.

6.1.2.1 Rule Templates

The following figure shows the window that IBM eNetwork Firewall 3.2 provides to define new rules:



(LOCAL) Modify IP Rule

Modify Rule

Identification

Rule Name:	HTTPS Ack 2/2 non secure		
Description:	TCP/ACK out port 443 tr0		
Action:	Permit	[✓ Protocol:]	tcp/ack
		Numeric Protocol:	

Source Port / ICMP Type

Operation:	Equal to	Port #/Type:	443

Destination Port/ ICMP Code

Operation:	Greater than	Port #/Code:	1023

Interfaces Settings

Interface	Specific
Name:	tr0

Direction/Control

Routing:	both local [✓ route]
Direction:	both inbound [✓ outbound]
Log Control:	Yes [✓ No]
Frag. Control:	Yes

Figure 257. Rule Definition with IBM eNetwork Firewall 3.2

Rules are the basic units in IP filter definitions. A rule does not include the source or destination objects. This means that the rule does not depend on them and can be recycled for other services. Moreover this means that you do not need to consider the flow of the protocol when you define a new rule. However a rule depends on a protocol and when you define a new rule you can specify only one protocol. This is the list of the protocols you can select:

1. **all**

   Any protocol will match this rule.

2. **tcp**

   The packet protocol must be TCP to match this rule.

3. **tcp/ack**

   The packet protocol must be TCP with acknowledgment (ACK) to match this rule. Selecting this protocol is a security measure that is usually needed to prevent undesired connections with the secure network being established from the Internet, as shown several times in Chapter 8, “Three-Tier Applications in Firewall-Protected Network Environments” on page 361.

4. **udp**

   The packet protocol must be UDP to match this rule.

5. **icmp**

   The packet protocol must be Internet Control Message Protocol (ICMP) to match this rule.

6. **ospf**

   The packet protocol must be Open Shortest Path First (OSPF) protocol to match this rule.

7. **ipip**

   The packet protocol must be IP-in-IP (IPIP) protocol to match this rule.

8. **esp**

   The packet protocol must be Encapsulating Security Protocol (ESP) to match this rule.

9. **ah**

   Authentication Header (AH) is the packet protocol used by the Virtual Private Network for sending IP packets which have an associated authentication token.

As you have seen, application protocols such as HTTP, FTP and telnet are not present in the above list. Most applications run on top of those protocols. You may need to combine multiple protocols along with application source and destination ports, to define a filter service for a specific TCP/IP application, as shown in 6.1.2.2, “Services” on page 276. In cases like these, you will include a multiprotocol combination in the service definition.

The rule template contains other properties that you have to select in order to have your firewall properly configured:

- Action

  You can select whether you want to permit or deny access to the firewall.

- Source Port and Destination Port

  The source port is the port number (or range of port numbers) used on the machine from which the packets are flowing. The destination port is the port number (or range of port numbers) used on the machine where the packets arrive. Port numbers must always be in the range 1 through 65535. Remember that port numbers less than 1024 are considered privileged and are used by specific applications. When you are configuring your firewall in order to let a client machine in the secure network communicate with a server machine in the non-secure network, you usually select a port on the client **Greater than** `1023`, since the exact value on the client, even if greater than 1023, is dynamically determined when the communication takes place, while the port on the server will be **Equal to** the specific value used by the server application. For example 80 is the value reserved for a Web server communicating via HTTP and 443 is used if the communication is via SSL.

Assigned Port Numbers

A list of all the assigned port numbers is provided by the Networking Working Group in the Request for Comments RFC1700, at http://info.internet.isi.edu/in-notes/rfc/files/rfc1700.txt.

If you are defining a rule for which the source port is on the non-secure network, remember that in general you should also select the protocol as **tcp/ack**, meaning that only packets with the ACK flag set will be permitted to flow through the firewall. In this way, since the ACK flag is not set only in the first packet of a TCP communication, you can prevent unauthorized connections with the secure network being established from the Internet. We show several examples of how to use the tcp/ack protocol in Chapter 8, “Three-Tier Applications in Firewall-Protected Network Environments” on page 361 and Chapter 9, “IIOP in Firewall-Protected Network Environments” on page 435.

- Interface Settings

  When you define a new rule, the Interface Settings section allows you to select the type of interface through which the flow passes. You have four options:

  1. **both**

     For packets coming or going on either the secure or the non-secure network interface.

  2. **secure**

     For packets coming or going on the secure interface.

  3. **nonsecure**

     For packets coming or going on the non-secure interface.

  4. **specific**

     For packets coming or going on the network interface that you must specify in the Interface Name field.

- Routing

  This property specifies if and how the firewall routes the traffic. You have three different options:

  1. **both**

     This option applies to all traffic.

  2. **local**

     This option means that packets are local to the firewall host. This implies that:

  – Incoming packets are packets that are received by the interface but will remain in the firewall host. Their destination is local and they will not be routed to other machines.

  – Outgoing packets are transmitted through the interface to another machine, but their origin is local, meaning that they have been originated in the firewall machine itself.

3. **route**

   This option means that packets are not local to the firewall host, but are routed by it. This implies that:

   – Incoming packets are packets that are received by the interface and are destined to another host. Their destination is remote, not local. Those packets will not remain in the firewall.

   – Outgoing packets are transmitted through the interface to another machine but their origin is remote, meaning that they were originated by another machine and the firewall is routing them.

- Direction

  You can use this property to select the direction of the protocol through the network interface. You have again three different options:

  1. **both**

     Select this option for packets going out from or into the interface.

  2. **inbound**

     Select this option for packets coming into the selected interface from the network.

  3. **outbound**

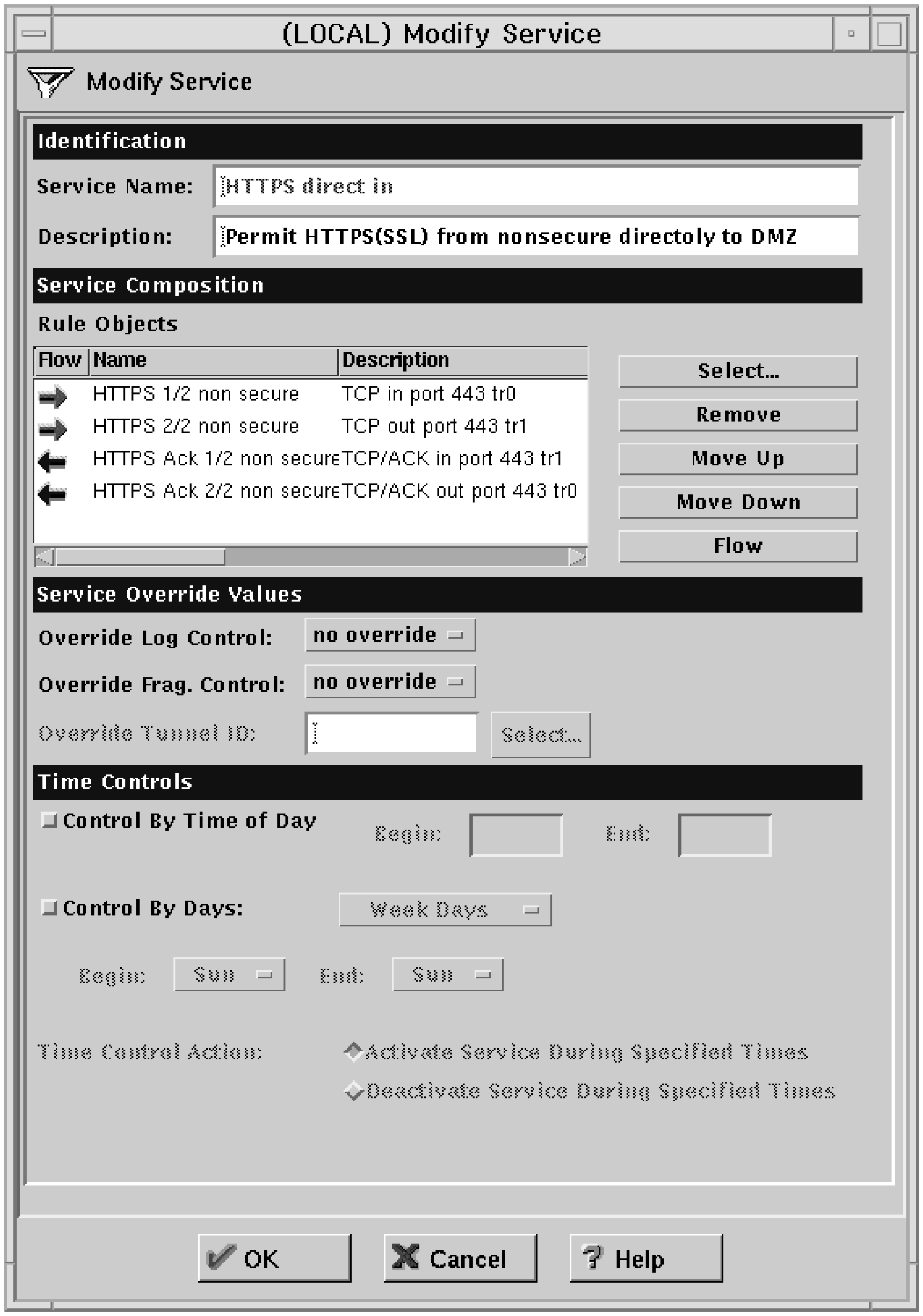
     Select this option for packets going out from the selected interface to the network.

6.1.2.2 Services

You can collect more than one defined rule into a service. When you add one or more rules to a service, you must specify the Flow property for each rule. In fact, even if a service does not include yet the concepts of source and destination objects, each service will be added to a connection, where a source and a destination object will be defined. So, each time you add a rule to a service, you should think if that rule will have to be applied from the source to the destination object or from the destination to the source object. Depending on this decision, you must change the direction of the arrow that appears near the rule name:

- An arrow going from left to right indicates that the rule in the service will be applied from the source to the destination object of the connection.
- An arrow going from right to left indicates that the rule in the service will be applied from the destination to the source object of the connection.

The following window is provided by IBM eNetwork Firewall 3.2 to define a new service:


(LOCAL) Modify Service

Modify Service

Identification

Service Name: HTTPS direct in

Description: Permit HTTPS(SSL) from nonsecure directoly to DMZ

Service Composition

Rule Objects

Flow	Name	Description
➡	HTTPS 1/2 non secure	TCP in port 443 tr0
➡	HTTPS 2/2 non secure	TCP out port 443 tr1
⬅	HTTPS Ack 1/2 non secure	TCP/ACK in port 443 tr1
⬅	HTTPS Ack 2/2 non secure	TCP/ACK out port 443 tr0

Service Override Values

Override Log Control: no override

Override Frag. Control: no override

Override Tunnel ID:

Time Controls

Control By Time of Day   Begin: End:

Control By Days: Week Days

Begin: Sun   End: Sun

Time Control Action: Activate Service During Specified Times / Deactivate Service During Specified Times

Figure 258. Service Definition Using IBM eNetwork Firewall 3.2

Notice that the Time Controls section allows you to select when you want that the service you are defining to be activated or deactivated at specified times.

6.1.2.3 Network Objects

Network objects represent components that exist in the network, like hosts, networks, firewalls, routers, interfaces, Virtual Private Networks and users. If you create new network objects, you can then use them to define a new connection, since the source and the destination object of a connection are network objects.

A network object is identified by a type, a name and a description. Possible types that you can select when defining a new network object are:

1. **Host**

   A particular node on your network with a mask of 255.255.255.255.

2. **Network**

   A collective range of network addresses that is characterized by an address range and a specific subnet mask.

3. **Firewall**

   A single machine with a firewall installed on it with a mask of 255.255.255.255.

4. **Router**

   A host that routes traffic between two or more networks with a mask of 255.255.255.255.

5. **Interface**

   A network adapter on a machine with a mask of 255.255.255.255. It does not have to be an adapter on the firewall.

6. **VPN**

   A Virtual Private Network on the other side of a tunnel.

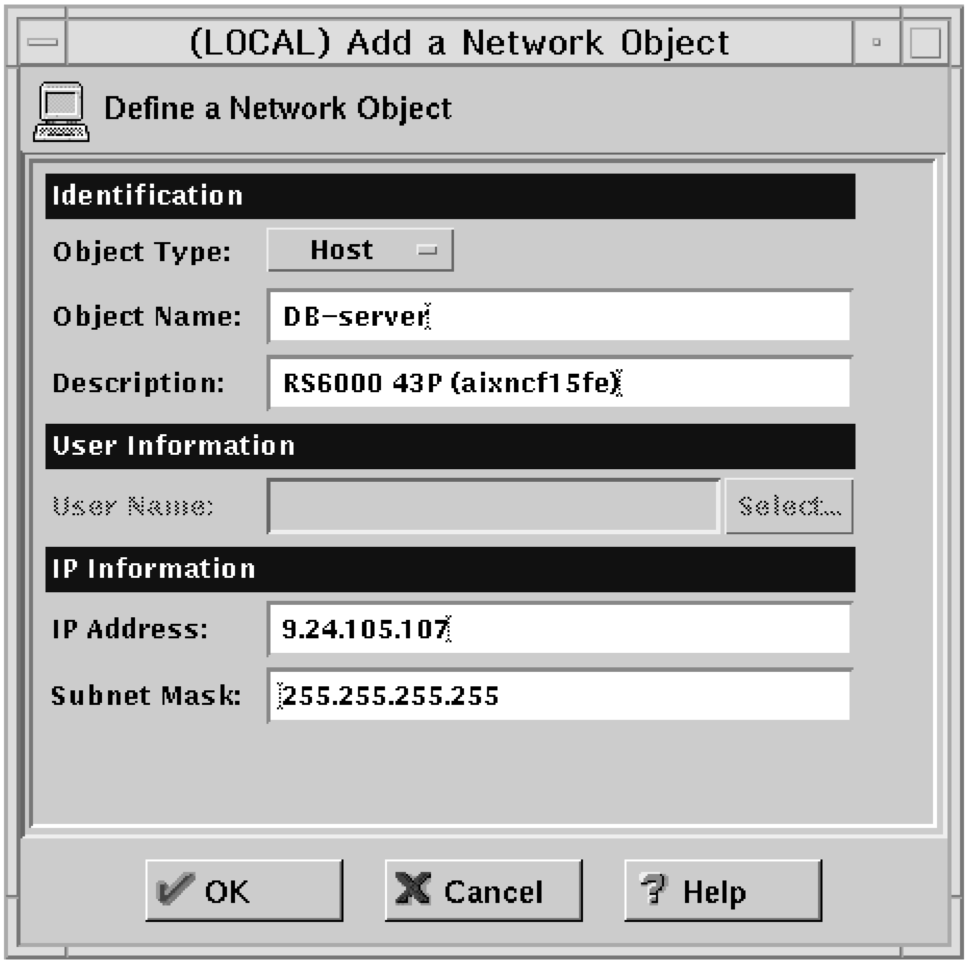
7. **User**

   A remote client without an IP address or subnet mask defined. The user definition is used for a remote client who will be dynamically assigned an IP address at connect time.

When you define a new network object, you must specify its IP address also. So, when you add this network object to a connection definition, as the source or the destination object, you will simply have to select the name of the network object, without the need to specify the IP address or the subnet mask.

In all the examples provided in this book, we always define network objects whose type is Host. Each time we enter a subnet mask of 255.255.255.255, because this is the correct subnet mask to enter when you want to define a single Host object. If you specify 255.255.255.0 as subnet mask in a network object, you are defining a network object corresponding to a maximum number of 254 Host objects.

The following window shows how to define a new network object with IBM eNetwork Firewall 3.2:


(LOCAL) Add a Network Object

Define a Network Object

Identification

Object Type: Host

Object Name: DB-server

Description: RS6000 43P (aixncf15fe)

User Information

User Name:

Select...

IP Information

IP Address: 9.24.105.107

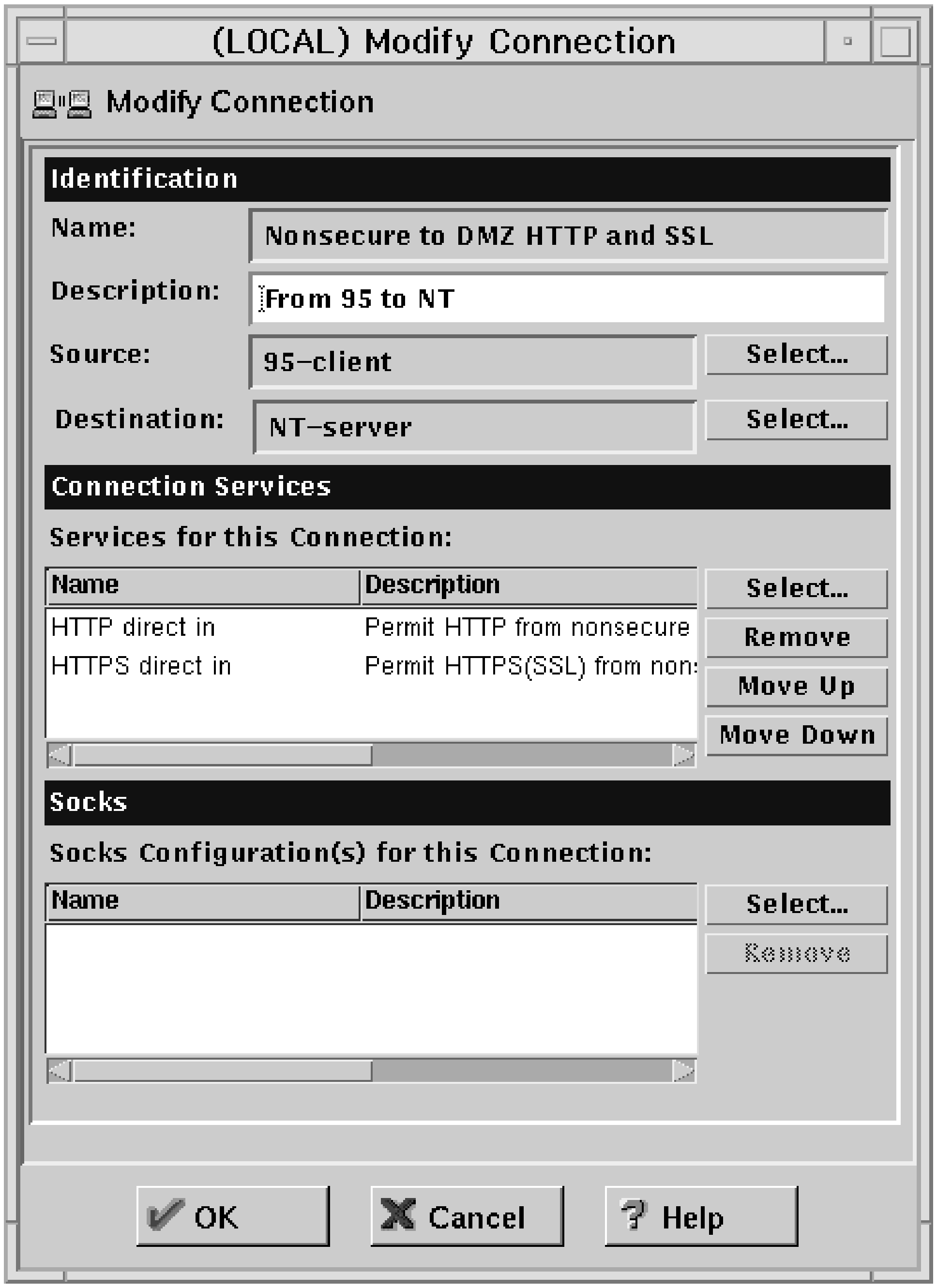
Subnet Mask: 255.255.255.255

OK Cancel Help

Figure 259. Network Object Definition with IBM eNetwork Firewall 3.2

6.1.2.4 Connections

You can collect more than one defined service to create a new connection, specifying also the source and the destination object to which the services apply. The following figure shows how a connection can be easily defined by using IBM eNetwork Firewall 3.2:



(LOCAL) Modify Connection

Modify Connection

Identification

Name:	Nonsecure to DMZ HTTP and SSL
Description:	From 95 to NT
Source:	95-client
Destination:	NT-server

Connection Services

Services for this Connection:

Name	Description
HTTP direct in	Permit HTTP from nonsecure
HTTPS direct in	Permit HTTPS(SSL) from non:

Socks

Socks Configuration(s) for this Connection:

Name	Description

Figure 260. Connection Definition

6.1.3 Proxy Servers

Proxy servers are tools that run at the application level of the ISO/OSI network model. For this reason they are also known as application level gateways.

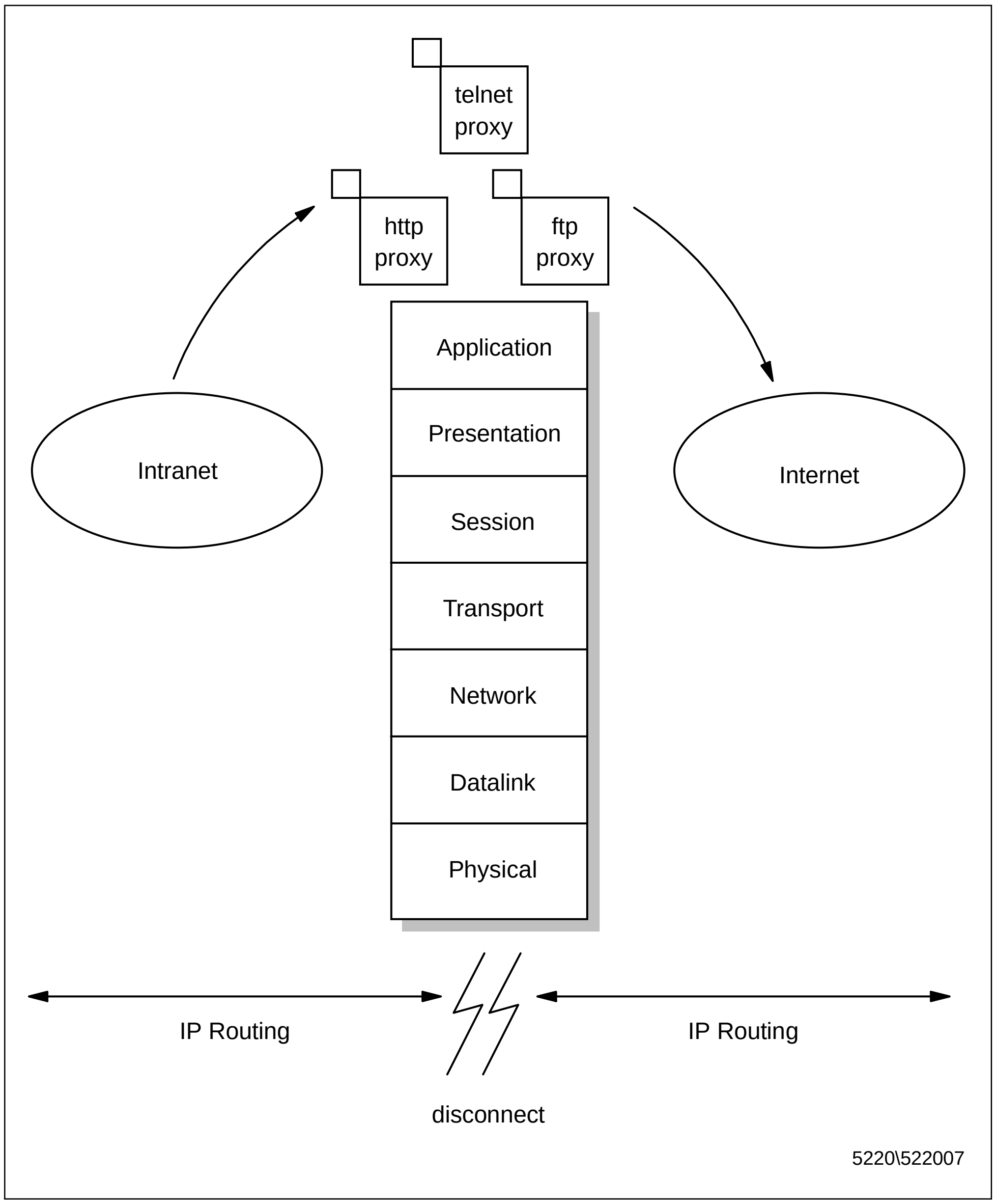


Figure 261. Proxy Servers

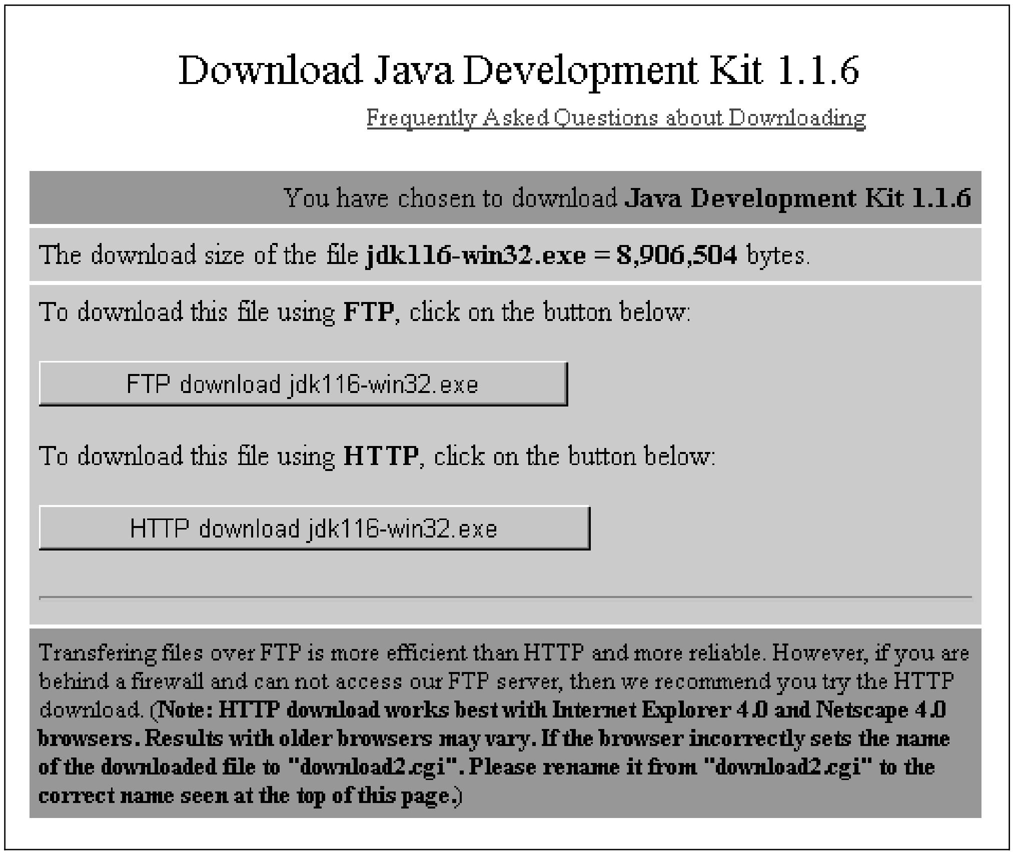
When a firewall acts as a proxy server, it performs the necessary work on behalf of the secure user. The client must send its request to the proxy server and not to the server directly. The connection is broken at the proxy server, which sends the client's request to the server without any computer on the non-secure network knowing that the secure client even exists. Using this firewall technology, the client and the server do not speak directly to each other, but through the proxy server, so that the client located in the intranet remains hidden to the rest of the world. Proxy servers do not involve packet routing at all. The IP address of the client machine on the intranet from which the IP session was established will never appear on the Internet, and attackers and intruders cannot use addresses of the protected network to gain information about the structure of the intranet.

Proxy servers also allow other security measures:

- Telnet and FTP proxy servers can be configured in order to perform authentication and authorization checks.

- Requests for Internet sites can be logged along with addresses of the requesting machines.
- Requests for certain Web sites can be banned.

HTTP proxy servers support several protocols, like FTP, HTTP, HTTPS (which is HTTP over SSL), Gopher and WAIS. The following screen shows, as an example, how you can use the same HTTP proxy server that you have configured to use with your browser, to download the Java Development Kit using either the HTTP protocol or the FTP protocol:


Download Java Development Kit 1.1.6

Frequently Asked Questions about Downloading

You have chosen to download **Java Development Kit 1.1.6**

The download size of the file **jdk116-win32.exe = 8,906,504** bytes.

To download this file using **FTP**, click on the button below:

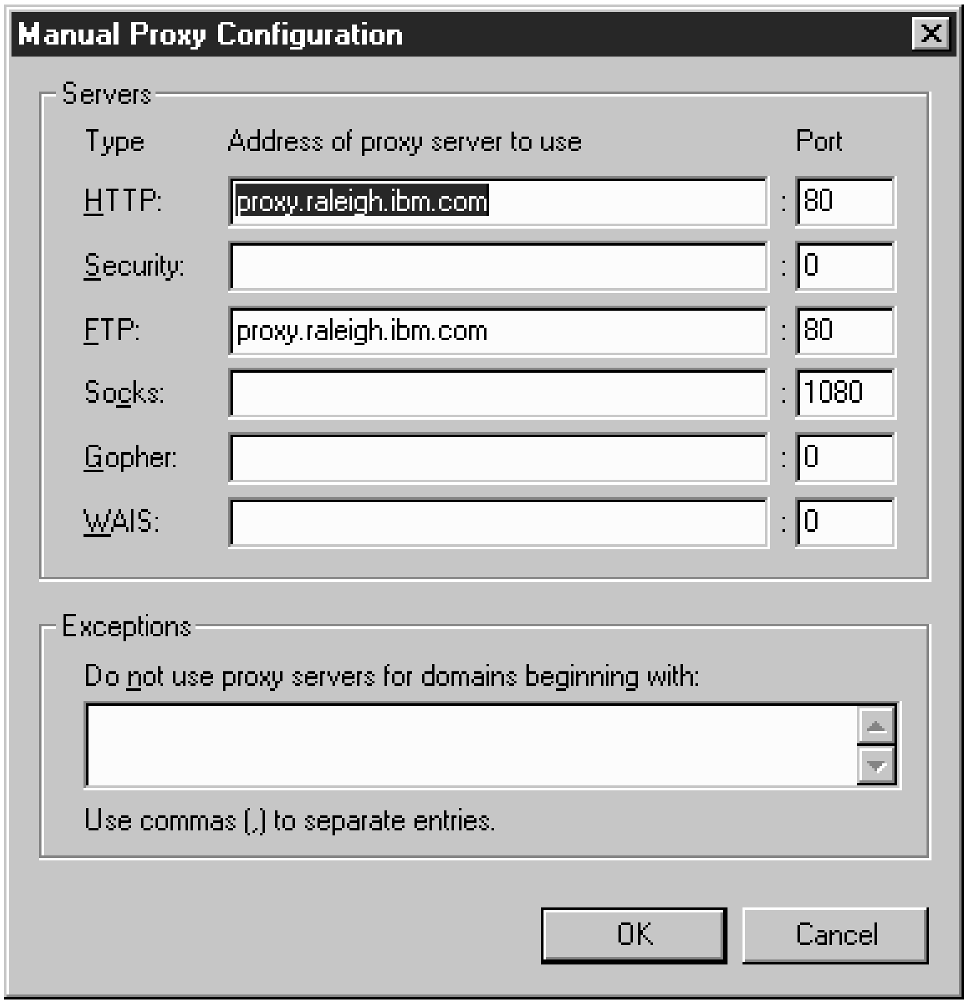
To download this file using **HTTP**, click on the button below:

Transfering files over FTP is more efficient than HTTP and more reliable. However, if you are behind a firewall and can not access our FTP server, then we recommend you try the HTTP download. (**Note: HTTP download works best with Internet Explorer 4.0 and Netscape 4.0 browsers. Results with older browsers may vary. If the browser incorrectly sets the name of the downloaded file to "download2.cgi". Please rename it from "download2.cgi" to the correct name seen at the top of this page.**)

Figure 262. HTTP Proxy Server Used for HTTP or FTP

When we worked with the IBM eNetwork Firewall 3.2 for AIX, we noticed that an HTTP proxy server on this platform could not be configured to perform user authentication, so it is advisable to use IP filters together with an HTTP proxy server on the firewall, in order to allow access from the secure network only. Notice that on Windows NT, the HTTP proxy does have user authentication.

The client browser must be configured to forward each request to the HTTP proxy server, or it will try to establish a direct connection to the Web server machine located in the Internet. Netscape Communicator 4.05 can be configured through the Preferences window. Opening the Edit menu, select **Preferences...**. The Preferences window will be brought up. You should select **Advanced** from the Category tree and then **Proxy**. In the Proxies section, mark the radio box **Manual proxy configuration** and then click **View**. You will then be able to fill the HTTP field with the IP address or the host name of the system running the proxy software and the port number for HTTP protocol access (usually 80). When these parameters are set correctly, you will be able to access Web pages and sites through the HTTP proxy server. If you want, you can use the same window to enter the IP addresses or the host names, and the TCP port, of the systems running the FTP proxy software. When these values are set correctly, you will be able to access anonymous FTP sites.


Manual Proxy Configuration

Servers

Type	Address of proxy server to use	Port
HTTP:	proxy.raleigh.ibm.com	80
Security:		0
FTP:	proxy.raleigh.ibm.com	80
Socks:		1080
Gopher:		0
WAIS:		0

Exceptions

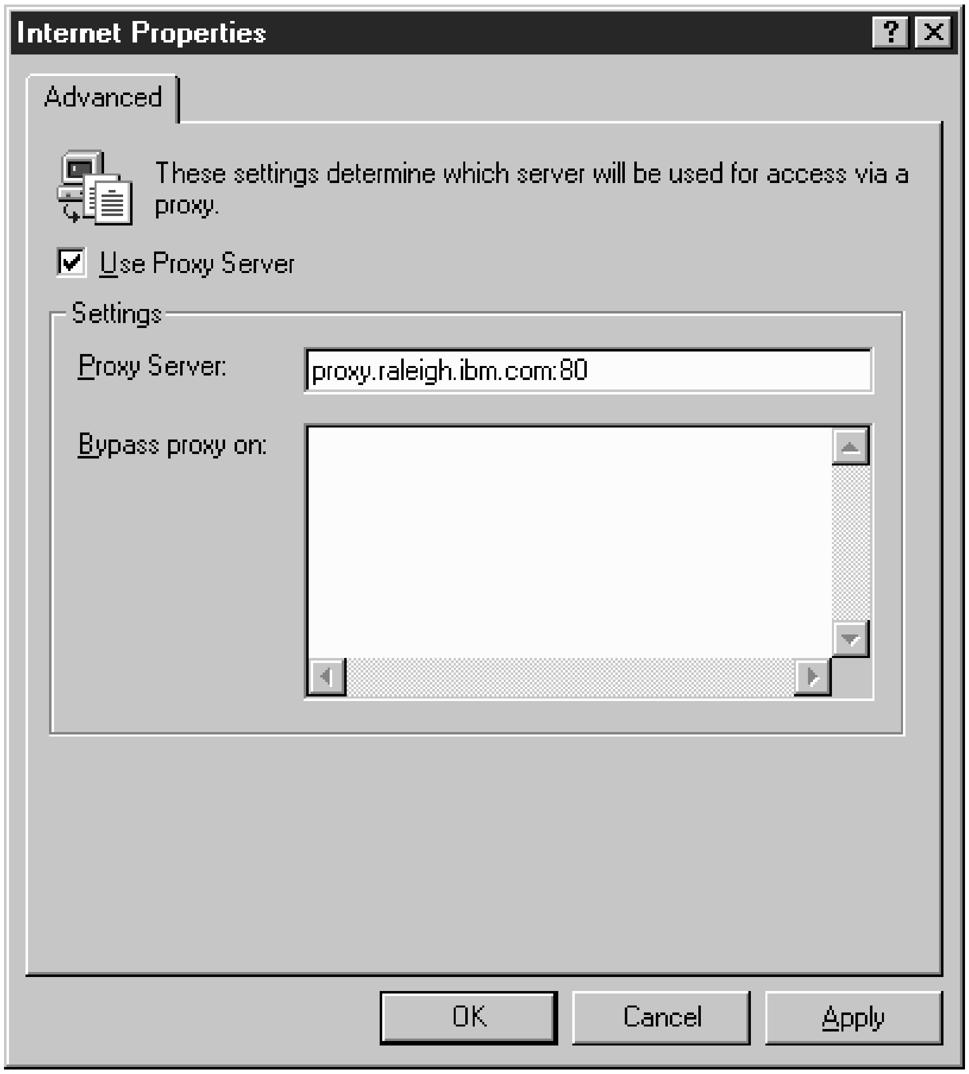
Do not use proxy servers for domains beginning with:

Use commas (,) to separate entries.

Figure 263. Netscape Communicator Advanced Configuration for Using HTTP Proxy Server

You can put entries for multiple hosts in each field, separated by commas. Wildcards are not allowed for multiple addresses. Since HTTP proxy servers support several protocols like HTTP, FTP, HTTPS, Gopher and WAIS, it is possible to enter the IP address or the host name of the same machine in those fields in the above figure. If all the parameters are set correctly, the client browser will be able to access Web pages and sites, anonymous FTP sites, Gopher menus, WAIS databases through the same proxy server.

Microsoft Internet Explorer 4.0 allows a similar configuration, by double clicking the **Internet** icon from the Control Panel and entering the IP address or host name of the HTTP proxy server in the Internet Properties window:


Internet Properties

Advanced

These settings determine which server will be used for access via a proxy.

[✓] Use Proxy Server

Settings

Proxy Server: proxy.raleigh.ibm.com:80

Bypass proxy on:

Figure 264. Internet Explorer Configuration for Using HTTP Proxy Server

Notice that with both Communicator and Internet Explorer you can select a set of IP addresses or host names located in your intranet for which you want to bypass the HTTP proxy server. This is very useful to reduce useless network traffic.

6.1.4 SOCKS Servers

SOCKS technology involves a SOCKS server running on the firewall machine. The client machine uses a TCP protocol named SOCKS to communicate with the SOCKS server.

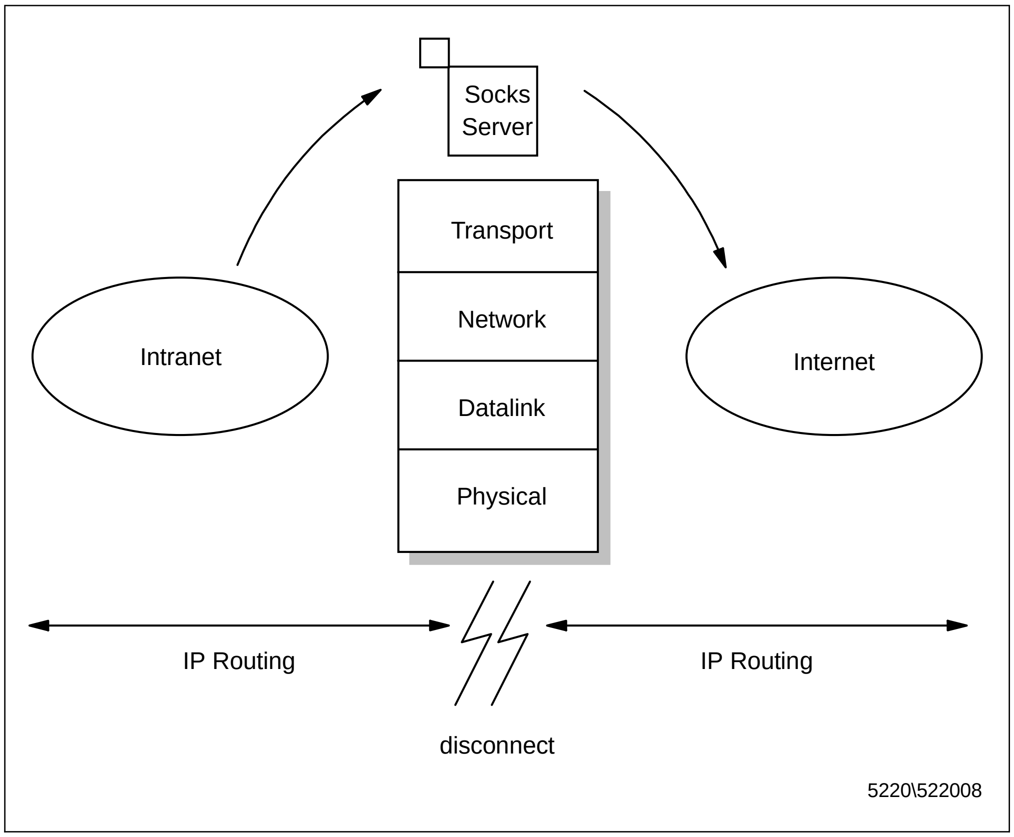


Figure 265. SOCKS Servers

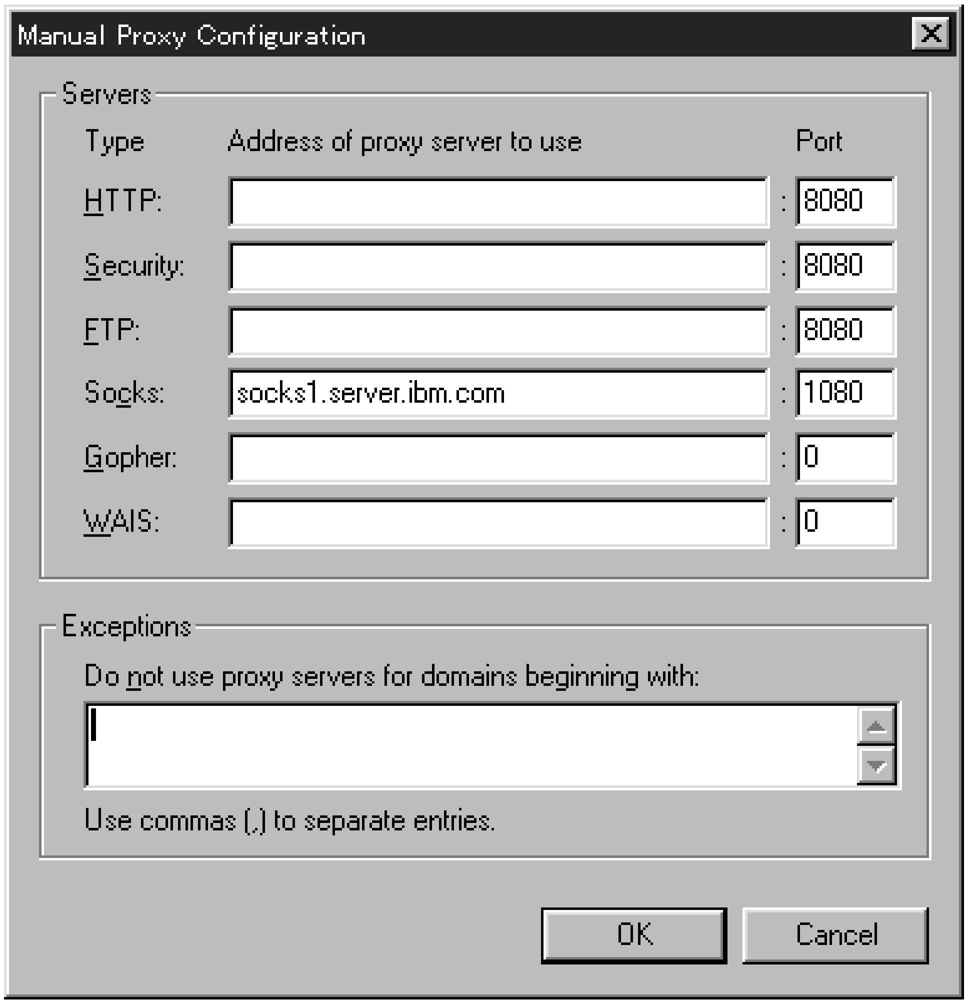
SOCKS technology provides security by encapsulating any TCP protocol in the SOCKS protocol. This operation is provided on the client machine, so that each packet is encapsulated within a SOCKS packet and then it is transmitted to the SOCKS server on the firewall. Then the SOCKS server extracts the proper information from each packet, like destination address and port number, and sends the data. Once again, as it happened with proxy server technologies, the session is broken by the firewall and once again the secure user is hidden to the Internet, so non-secure hosts cannot know that a secure user even exists and the source IP address that will get exposed to the Internet will be that of the SOCKS server itself. The IP address or host name of the machine from which the request actually started is not exposed to the non-secure network. Intruders and hackers cannot access internal information from your intranet.

When a non-secure host must send a response back to a secure client, actually it sends the response back to the SOCKS server, which will encapsulate the packets in the SOCKS protocol and then will send them back to the client machine in the secure network.

SOCKS technology has become very popular since it offers a big advantage. In fact the firewall administrator can simply allow any TCP/IP connection (any TCP protocol and any port number) between the firewall itself and the non-secure network, protecting the secure network by denying all the connection to the secure network that have been initiated from the Internet.

From the above description, it looks clear that modifications are required on the client side in order for the client machine to be able to use the SOCKS protocol and communicate with the SOCKS server on the firewall machine. There are two possibilities to accomplish this function:

The first possibility is to re-compile the client software in order to link the network client code with the SOCKS libraries. If you follow this option, you will obtain a *SOCKSified* client code. Of course you need to have the source code of the network client application in order to re-compile and re-link it. Fortunately a lot of client applications, like Netscape Communicator and Microsoft Internet Explorer, already offer built-in support for communicating with a SOCKS server using the SOCKS protocol. It is enough that you configure them specifying the SOCKS server's IP address or host name, then all the client requests will be passed to that server using the SOCKS protocol. Of course the TCP port used on the SOCKS server must be specified too. Its value is usually 1080. The following figure shows the configuration page for Netscape Communicator in order to use the SOCKS protocol to reach a SOCKS server :


Manual Proxy Configuration

Servers

Type	Address of proxy server to use	Port
HTTP:		8080
Security:		8080
FTP:		8080
Socks:	socks1.server.ibm.com	1080
Gopher:		0
WAIS:		0

Exceptions

Do not use proxy servers for domains beginning with:

Use commas (,) to separate entries.

Figure 266. SOCKS Server Configuration

You can access this page by opening the Edit menu and selecting **Preferences...**. When the Preferences window is brought up, you should select **Advanced** from the Category tree, and then **Proxy**. Remember that you do not have to fill both the HTTP and Socks fields in Figure 266, specifying simultaneously an HTTP proxy server and a SOCKS server, or each request starting from your browser will be sent to the SOCKS server first and from there to the proxy server, causing useless network traffic.

Unfortunately a SOCKSified version of a client program is not always available and you not always own the source code of that application, so you cannot re-compile or re-link it. A recent approach, and this is the second possibility, is to replace the dynamically linked libraries, that implement the TCP calls, with a SOCKSified version, named *SOCKSified TCP stack*. The SOCKSified TCP stack can be used with any client application, without the need to modify the code. In this case also it is required that the client program is aware of the SOCKS server's IP address or host name.

6.1.5 Understanding SSL Tunneling

SSL tunneling is a new firewall technology that is used to pass SSL through a firewall. We use SSL tunneling in Chapter 10, “SSL Tunneling and SOCKS Server for HTTPS Scenarios” on page 487. You can refer to 10.1, “SSL Tunneling Scenario” on page 488 to see all the implementation details.

The Secure Socket Layer (SSL) protocol was designed by Netscape Communications Corporation to provide security in a client/server environment and to avoid any type of Man-In-the-Middle (MIM) attack. For this reason, the SSL protocol cannot pass through a traditional proxy server, because SSL would consider a proxy server like a MIM.

A first solution to this problem is to configure the firewall through appropriate IP filters. We have already explained how IP filter technology provides security by controlling source and destination IP addresses and masks, type of IP protocol, source and destination port numbers, traffic direction, network interfaces, etc. The advantage of IP filters when using the SSL protocol is that when the firewall implements IP filters without acting as a proxy, the secure host can speak directly with the non-secure host in the Internet, since the communication is not broken by the firewall. You could open the reserved port 443 on the firewall to pass SSL and the client and the server could establish a direct connection. The risk of this solution is that an internal attacker could try to use the same 443 TCP port without using SSL and there would be no possibility to block a similar attack.

A more secure alternative is to use a firewall that supports the SSL tunneling CONNECT extension method, as described at http://www.netscape.com/newsref/std/tunneling_ssl.html.

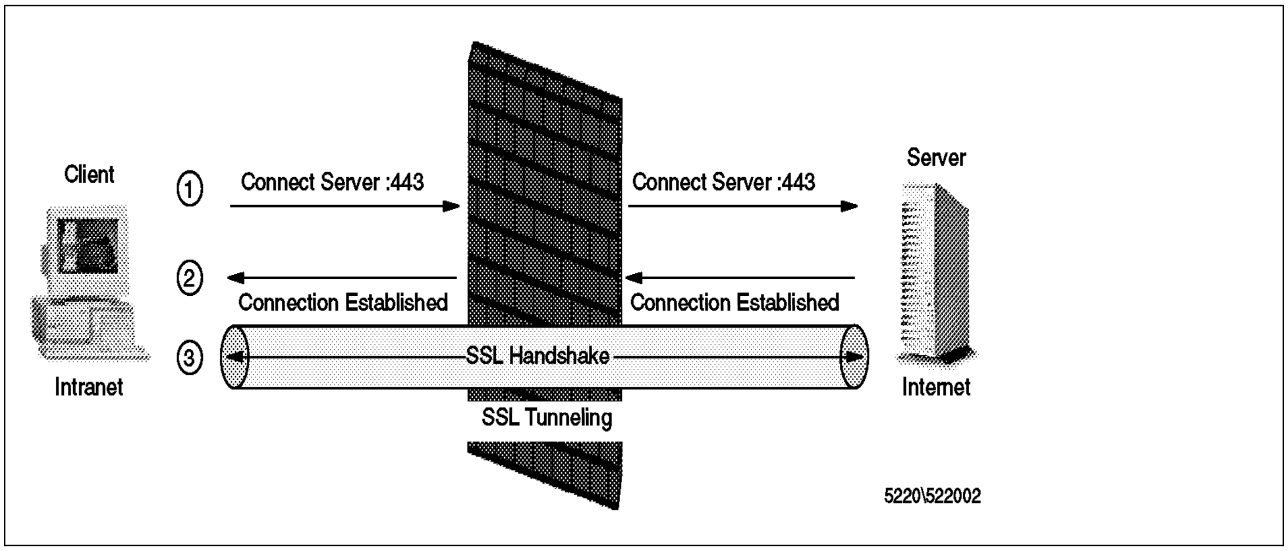


Figure 267. SSL Tunneling

The client connects to the firewall and uses the CONNECT method to specify the host name and the port number to connect to. This information is passed within the CONNECT header and the host name and the port number must be separated by a colon, something like `Server:port`. The port number that Web servers usually set for SSL communication is 443, since this is the reserved number that the Networking Working Group has assigned to SSL in the Request for Comments RCF1700.

Using SSL tunneling, the client initiates an SSL secure connection via HTTP, then handshakes and creates a secure connection to the server. The firewall has access to the client proxy request headers, but the session is encrypted. As soon as the handshake has taken place, the firewall simply copies bytes back and forth to each side of the transaction. In this way the firewall can monitor the request but not the traffic.

So, in order to support SSL tunneling, an HTTP proxy server does not need to implement its own SSL, but only needs the support for the SSL tunneling CONNECT extension method. Anyway the HTTP proxy server cannot access any private data, like for example client and server's certificates, the shared ciphering key and the application data exchanged during the SSL connection. Moreover the HTTP proxy server does not cache the SSL traffic.

Notice that when a firewall is configured in order to permit the SSL protocol, it is usually configured to permit the HTTP protocol. It is in fact very common to access an `https://` URL from a Web page that had been accessed through an `http://` URL, while it is rare that a user on a client browser accesses a secure Web site directly pointing to an `https://` URL (see 8.5.3, “IP Filters Configuration for HTTP and SSL” on page 416, 10.1.1, “How to Configure SSL Tunneling” on page 488 and 10.2.1, “How to Configure SOCKS Server for HTTPS” on page 494).

The Web browser application needs to be properly configured in order to use SSL tunneling. The HTTP and Security fields in the Manual Proxy Configuration window provided by Netscape Communicator must be filled with the name of the HTTP proxy server implementing the SSL tunneling CONNECT extension method, as shown in Figure 452 on page 493.

6.2 Understanding Firewall Architectures

In this section we will provide a description of the most common firewall architectures.

6.2.1 Screening Router

This firewall architecture is the first level of defense that is usually implemented when protecting a network against invasions from the Internet. The intranet is separated from the non-secure network by using a router.

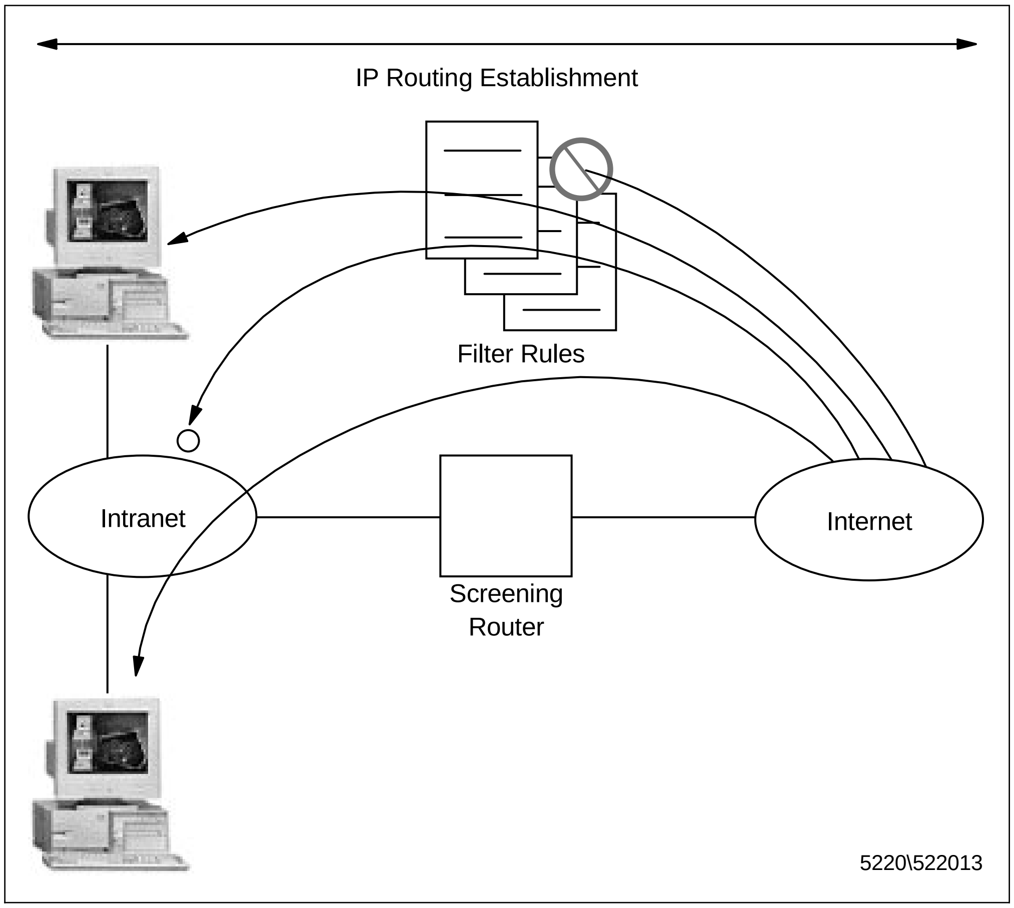


Figure 268. Screening Router

A screening router filters all IP packets that pass through. It can prevent access to machines or ports in the secure network, but it can also do the opposite operation, in the sense that it can prevent a machine in the secure network from accessing the Internet.

A screening router performs its packet filtering at the network and data-link layers, so it does not consider the application level of the ISO/OSI network model. For this reason, a screening router does not offer complete protection. Network traffic is controlled by screening routers without changing any client or host application. Since the action performed by a screening router is restricted to the network and data-link layers of the TCP/IP protocol, a screening router can have access to information like IP addresses, port numbers and TCP flags, but nothing more than this, so there is no possibility to understand what happens at the application layer.

Anyway screening routers are often used in conjunction with other tools as security building blocks.

6.2.2 Bastion Host

A bastion is simply a machine having two or more network adapters, placed between an intranet and the Internet and not allowing any IP forwarding. In other words, a direct IP connection is denied through this machine and no IP packet can flow directly between the secure and the non-secure network.

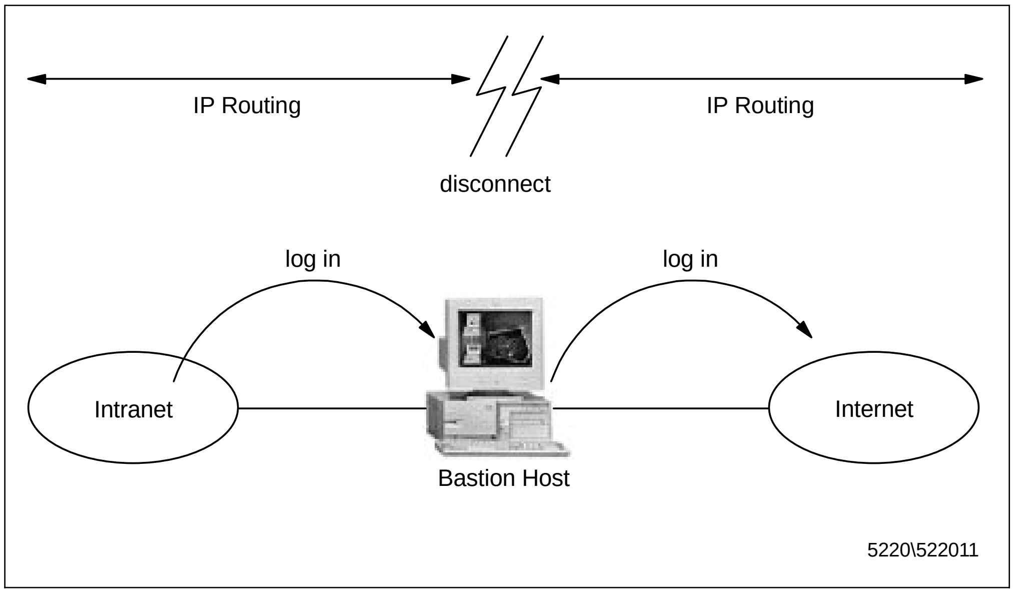


Figure 269. Bastion Host

The only machine from which users can have access to both the intranet and the Internet is the bastion itself. No computer in the secure network can access the Internet without passing through the bastion and no computer in the Internet can access the secure network without passing through the bastion.

Since IP forwarding is denied, only users having an account on the bastion, with two identifications (one for the bastion and one for the remote host), can have access to both networks.

This configuration presents some advantages, because all network access is centralized in only one machine and for this reason it becomes easier to manage network security. But, because the only way to access both networks is to be a user on the bastion, this machine must be able to support many users. Moreover a security risk is that a cracker who is able to impersonate an user would also be able to have full access to the secure network. For this reason password controls must be enforced.

One solution to this problem is offered by SOCKS servers, which implement more general purpose bastion applications. They do not require that users log in and this way the load on the machine is reduced.

6.2.3 Screened Bastion Host

It often happens that firewall administrators combine a screening router with a bastion on the same machine and obtain what is called a *dual-homed gateway*. The advantage is that you can protect a dual-homed gateway from external attacks with filtering. The disadvantage is that managing this machine becomes very difficult and in case an attacker penetrates this machine it could take a lot of time to discover it.

A better solution is to use two separate machines: one is a bastion and the other is a screening router. The screening router is for protecting the bastion from external attacks. The firewall architecture you obtain in this case is called a screened bastion host.

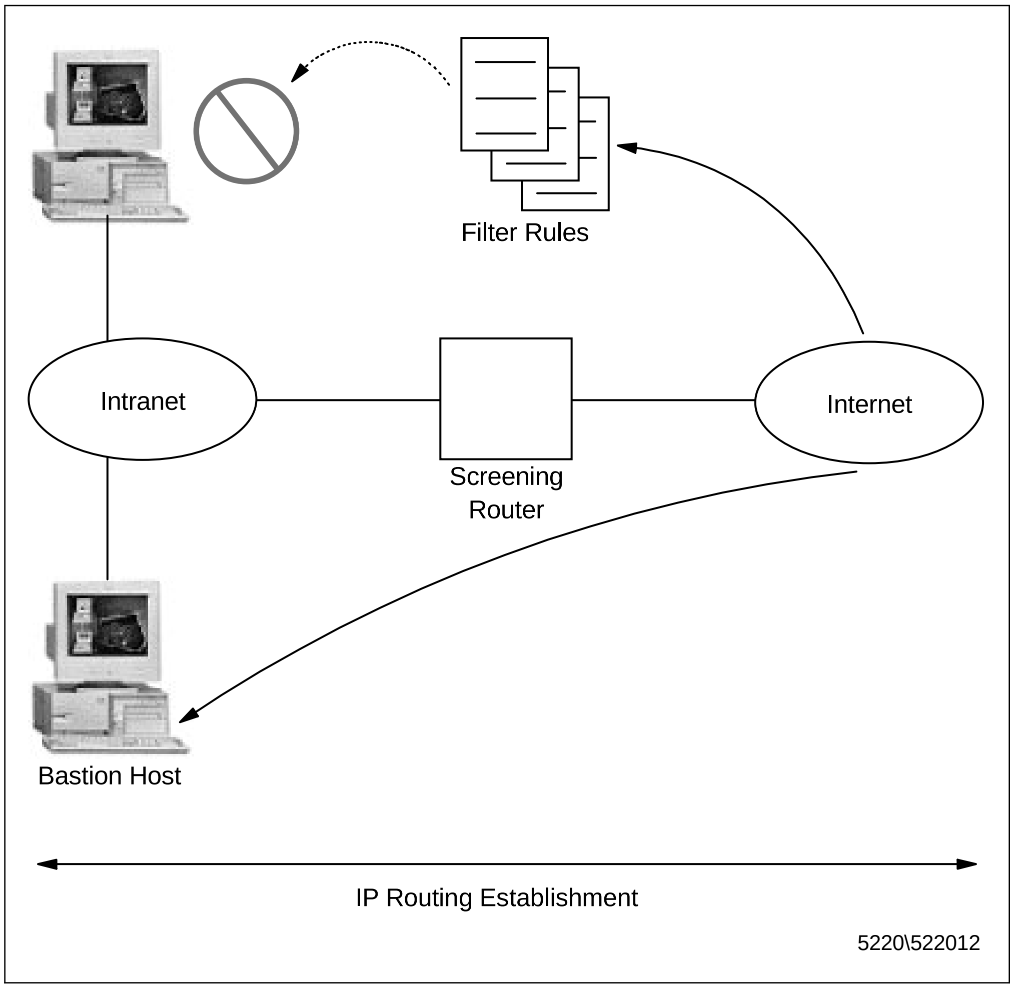


Figure 270. Screened Bastion Host

For a screened bastion host the same considerations that we made for a general bastion can be applied, so it still holds true that this machine denies IP forwarding between the intranet and the Internet and that each user who wants to have access to both networks must have an account on the bastion with two identifications. The presence of the router protecting the bastion host, however, offers more security, because it becomes very difficult for a cracker to penetrate the secure network.

In this case also it is possible to configure the bastion as a SOCKS server, in order to avoid that the bastion supports many users.

6.2.4 Screened Subnet

Another common firewall architecture that combines bastion hosts and screening routers is called screened subnet. In this case the subnetwork between the bastion and the screening router, which is also known as Demilitarized Zone (DMZ), is used as a site for application services, like for example a Web server. The reason why these machines are placed in this subnetwork rather than in the intranet itself is that in this way, while they are widely available, they also get protection from the external network. Another advantage from a security point of view is that this architecture can hide the intranet from the non-secure network, since each communication between the two networks needs to pass through the DMZ.

Usually each machine in the subnetwork performs only simple tasks, and the number of these machines depends on the number of bastions you have.

The following figure shows an example of screened subnet, obtained with two routers and two bastion hosts:

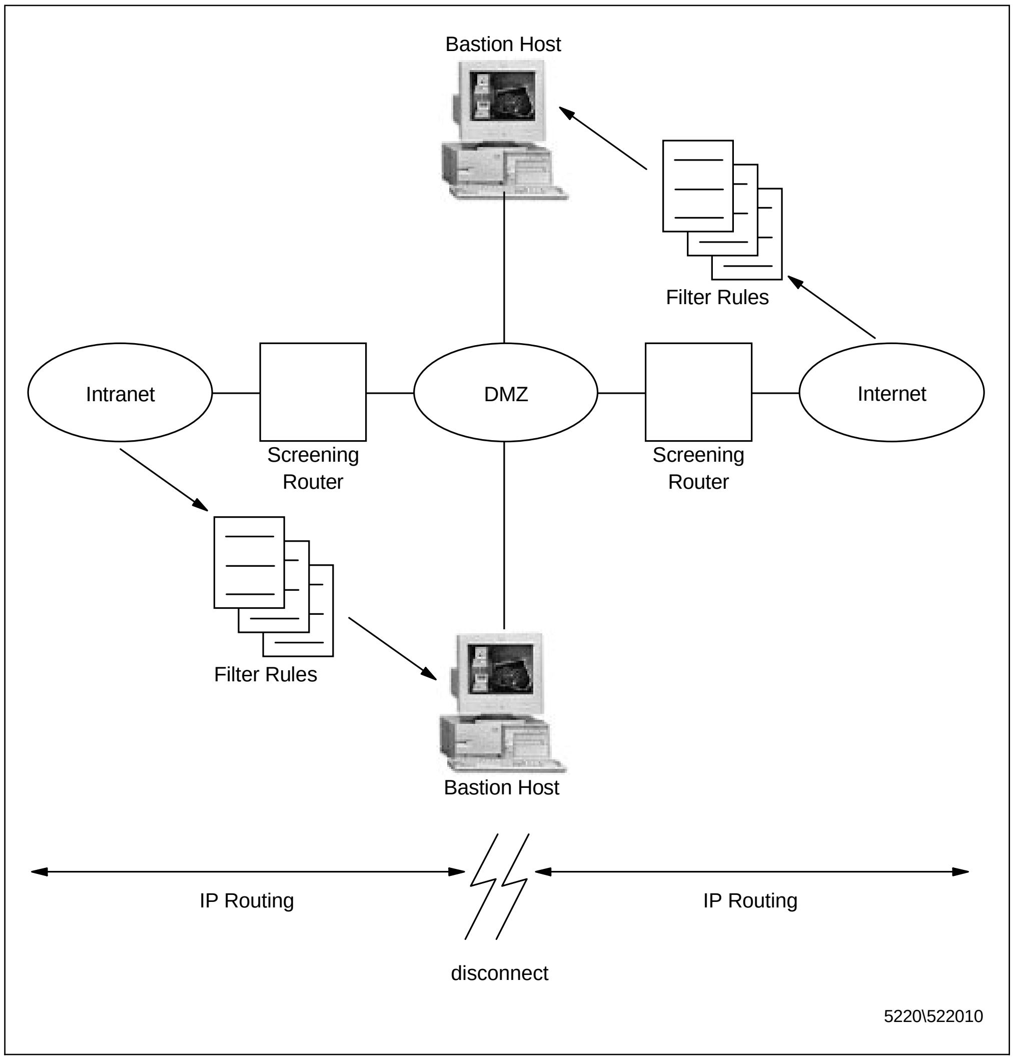


Figure 271. Screened Subnet between Two Screened Routers

A disadvantage of this architecture is its cost. In effect, dedicating a single machine to each component of the firewall design can be really expensive. A common solution, that we also apply in Chapter 8, "Three-Tier Applications in

Firewall-Protected Network Environments" on page 361, is to provide the firewall machine with three network adapters. The architecture that you obtain is logically the same, but physically different, since you need only one firewall machine. The following figure shows just how a screened subnet can be obtained using a single router with three network adapters:

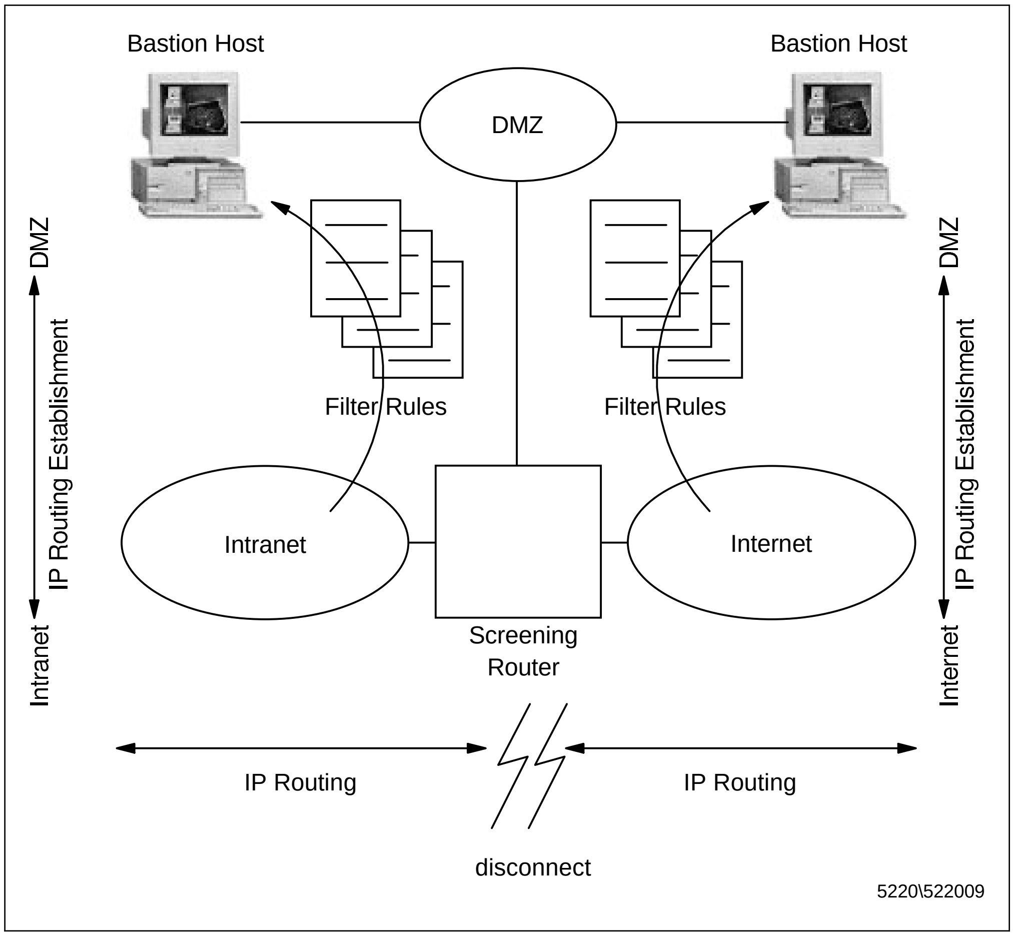


Figure 272. Screened Subnet Using Three Network Adapters

In this case the DMZ is still in the area where network traffic never flows between the intranet and the Internet. Refer to Chapter 8, "Three-Tier Applications in Firewall-Protected Network Environments" on page 361 to see all the implementation details.

Part 2. NCF Security Scenarios

Chapter 7. How to Install and Configure a NCF Secure Environment

This chapter shows how to install and set up the key products that are commonly used in an NCF secure environment on the Windows NT and AIX platforms. We will show you all the steps that we followed to install the components for the NCF secure environment that we used to build the NCF scenarios described in this redbook. An NCF environment can be based on a two-tier model, if it involves only a client/server interaction (see Chapter 9, “IIOP in Firewall-Protected Network Environments” on page 435 and Chapter 10, “SSL Tunneling and SOCKS Server for HTTPS Scenarios” on page 487), or it can be based on a three-tier model if it involves an application server too (see Chapter 8, “Three-Tier Applications in Firewall-Protected Network Environments” on page 361). In order to grant security in our NCF environment, we also used a firewall machine.

Before going on with this chapter, it would be good to have a general idea of how we installed and configured our NCF environment. The following list shows the operating system and software we used.

- Client (IBM ThinkPad 560E - Operating System Microsoft Windows 95)
  - Netscape Communicator 4.05
- Server (IBM PC 365 - Operating System Microsoft Windows NT 4.0)
  - Java Development Kit 1.1.5 and 1.1.6
  - Lotus Domino Go Webserver 4.6.2.2
  - IBM WebSphere Application Server 1.0 beta 3 (still named ServletExpress)
  - IBM DB2 Client Application Enabler 5.0
- Application Server (IBM RS/6000 43P - Operating System IBM AIX 4.3)
  - IBM DB2 Universal Database Workgroup Edition 5.0
- Firewall (IBM RS/6000 370 - Operating System AIX 4.2.0 and 4.2.1)
  - IBM Firewall 3.1.1 and IBM eNetwork Firewall 3.2

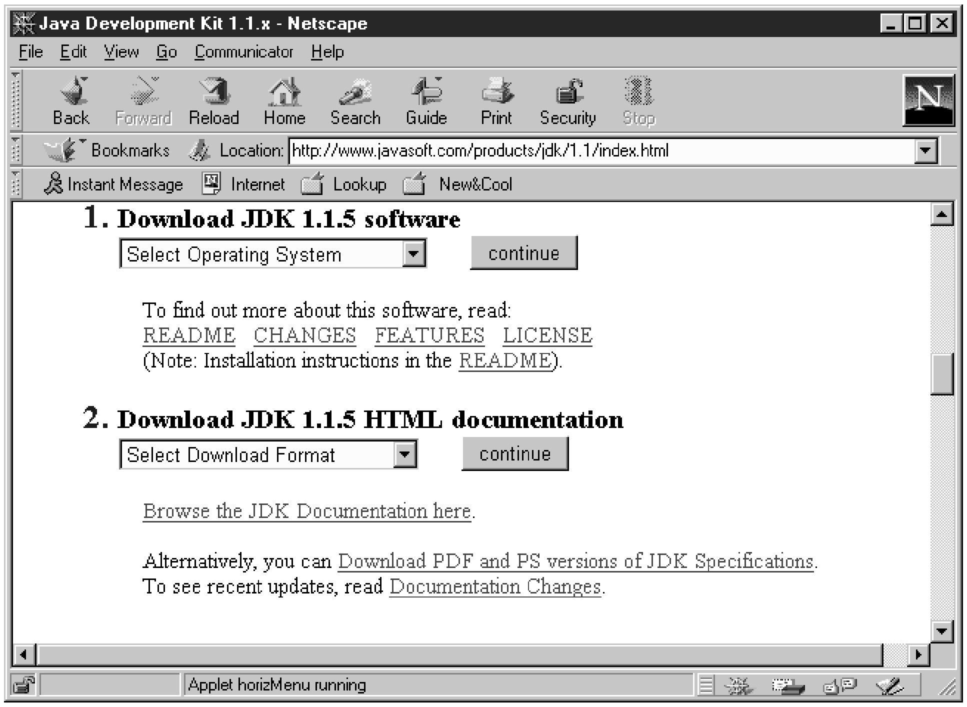
As we said in Chapter 1, “An Overview of NCF Security” on page 3, the client tier in an NCF environment only has the role of requesting the applications and presenting the information, through a GUI represented by a Java-enabled Web browser. In the NCF environment that we used for our scenarios, the client machine had been installed with Netscape Communicator 4.05 as Web browser. We will not show here how to install Communicator, since its installation is very simple. The client platform in our scenarios was Windows 95. Notice however that the operating system where a Java-enabled Web browser runs is not relevant in an NCF environment and does not impact the configuration of the other tiers, since HTML and Java are platform-independent languages.

As we will explain in 7.6, “IBM Firewall 3.1.1 and IBM eNetwork Firewall 3.2 for AIX” on page 350, we had the opportunity to work with IBM Firewall 3.1.1 (requiring AIX 4.2.0) and IBM Firewall 3.2 (requiring AIX 4.2.1). This is the reason why in the above list the firewall entry shows two different versions for the operating system and for IBM firewall.

7.1 Java Development Kit 1.1.5 and 1.1.6

This section describes how to configure the Java Development Kit (JDK) 1.1.5 properly. During the project JDK 1.1.6 became available, we installed it and made several tests using this new release. All the considerations that you read in this section about Version 1.1.5 can be applied to Version 1.1.6. Notice that we couldn't use Version 1.2, since it was still in beta at the time of the project and it was not supported by any Web browser or Web server yet (see Chapter 2, “The New Java 1.2 Security Model” on page 9).

The JDK can be downloaded for free from the JavaSoft Web site http://www.javasoft.com, as shown in the following window:


Java Development Kit 1.1.x - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://www.javasoft.com/products/jdk/1.1/index.html
Instant Message Internet Lookup New&Cool

1. Download JDK 1.1.5 software

Select Operating System [continue]

To find out more about this software, read:
README CHANGES FEATURES LICENSE
(Note: Installation instructions in the README).

2. Download JDK 1.1.5 HTML documentation

Select Download Format [continue]

Browse the JDK Documentation here.

Alternatively, you can Download PDF and PS versions of JDK Specifications.
To see recent updates, read Documentation Changes.

Applet horizMenu running

Figure 273. Downloading the JDK 1.1.5 from the JavaSoft Web Site

The installation of JDK was very easy. This tool is available for several platforms, but we installed it on a test machine running Windows NT Version 4.0 with Service Pack 3 as our base, and that was the same machine where later we installed also the Web server.

The JDK Version 1.1 contains four main components:

1. The Java Core Classes. They are stored in a zip file named classes.zip, that must not be unzipped. It must remain zipped for the compiler and interpreter to access the class files within it properly. This file contains all of the compiled class files for the JDK.

2. Java tools needed by developers (compiler, interpreter, Applet Viewer, debugger, documentation generator, Java Archive Tool, Digital Signing Tool, etc.). For our purposes, we used the Java compiler `javac`, that compiles Java programs into bytecodes, and the Java interpreter `java`, that executes Java bytecodes.

3. Java documentation and demos.

4. Java source files, provided to developers for information purposes only, to help them learn, use and understand Java. These files come inside the zip file src.zip and they should not be modified.

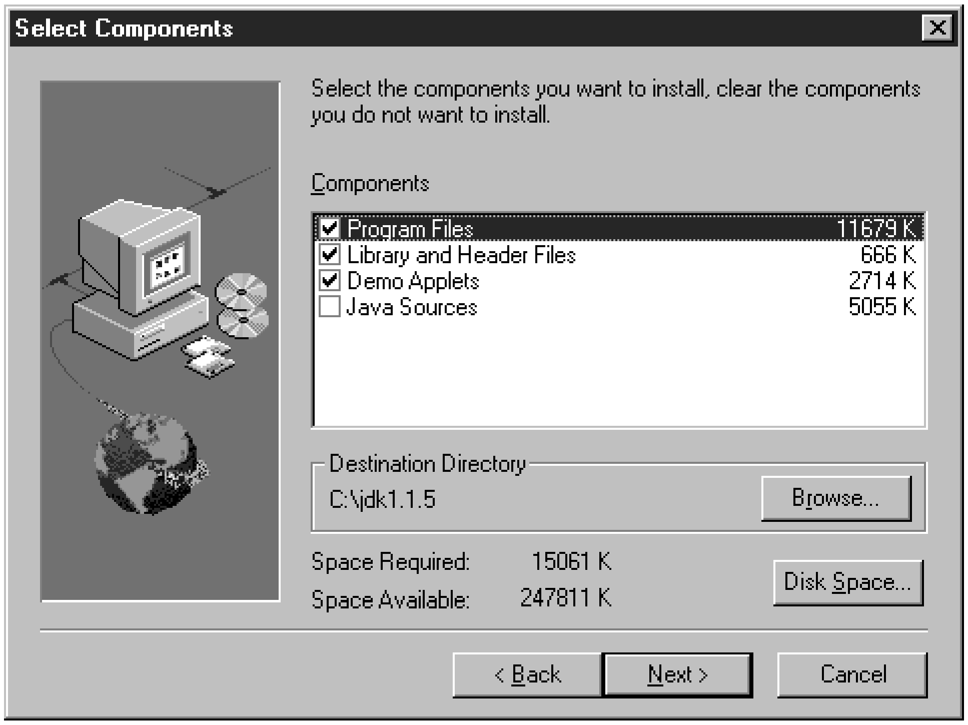
   We installed the following Java components:

   a. Program Files

   b. Library and Header Files

   c. Demo Applets

as shown in the following picture:



Select Components

Select the components you want to install, clear the components you do not want to install.

Components

✓	Program Files	11679 K
✓	Library and Header Files	666 K
✓	Demo Applets	2714 K
	Java Sources	5055 K

Destination Directory
C:\jdk1.1.5

Space Required: 15061 K
Space Available: 247811 K

Figure 274. JDK Installation Component List

As you can see, the component named Java Sources was not installed in our environment, since we did not consider it necessary to install the Java source files in our test environment.

7.1.1 Path System Environment Variable

We installed JDK 1.1.5 on Windows NT Server 4.0, so we changed the value for the Path system environment variable. In fact all the main Java tools run from the MS-DOS command line and you can specify the path to a tool either by typing the path in front of the tool each time you launch it or by modifying the value of the Path system environment variable. For example, if the JDK is installed in C:\jdk1.1.5, to run the compiler on a file myfile.java, go to a DOS shell and execute:

```
C:\jdk1.1.5\bin\javac myfile.java
```

or add `C:\jdk1.1.5\bin` to your Path statement and then simply type:

```
javac myfile.java
```

whatever the current directory is.

The second choice is more convenient and can be accomplished in the following way:

1. Open the Control Panel by selecting **Settings** from the Start menu.
2. Double click on the **System** icon.
3. Click the **Environment** tag at the top of the System window.
4. Select the `Path` environment variable and append `C:\jdk1.1.5\bin` to the current value into the Value box.
5. Click the **Set** button.
6. Click either **Apply** or **OK**.

Note

In Windows NT, all the changes made in the System Properties window to the system environment variables do not apply to applications currently running. So, for example, your changes to the Path variable will not impact an MS-DOS Command Prompt window that was already open and entering the command

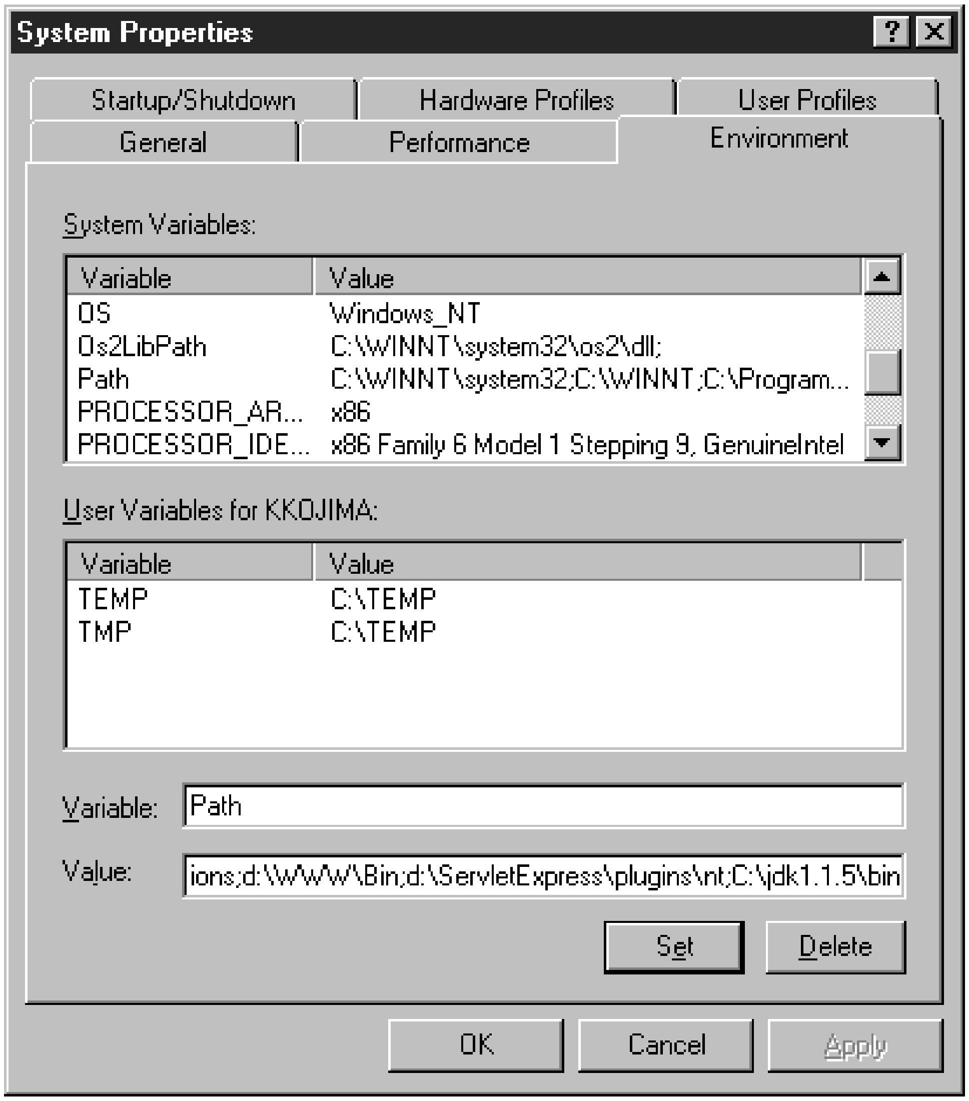
```
javac myfile.java
```

in a directory different from C:\jdk1.1.5\bin will fail. However opening a new MS-DOS Command Prompt window will force the system to read the new configuration and the command

```
javac myfile.java
```

will work, whatever the current directory is.

The following picture shows how you can have access to the Path statement through the System Properties window:


System Properties

System Variables:

Variable	Value
OS	Windows_NT
Os2LibPath	C:\WINNT\system32\os2\dll;
Path	C:\WINNT\system32;C:\WINNT;C:\Program...
PROCESSOR_AR...	x86
PROCESSOR_IDE...	x86 Family 6 Model 1 Stepping 9, GenuineIntel

User Variables for KKOJIMA:

Variable	Value
TEMP	C:\TEMP
TMP	C:\TEMP

Variable: Path

Value: ions;d:\WWW\Bin;d:\ServletExpress\plugins\nt;C:\jdk1.1.5\bin

Figure 275. Modifying the Path System Environment Variable

7.1.2 How the CLASSPATH System Environment Variable Works

The CLASSPATH tells the JVM (Java Virtual Machine) and other Java applications, which are located in the jdk1.1.5\bin directory, where to find the class libraries such as the file classes.zip, which is in the lib directory. By default, the java tools temporarily append the following to whatever CLASSPATH you have explicitly set in your startup file:

```
.;bin\..\classes;bin\..\lib\classes.zip
```

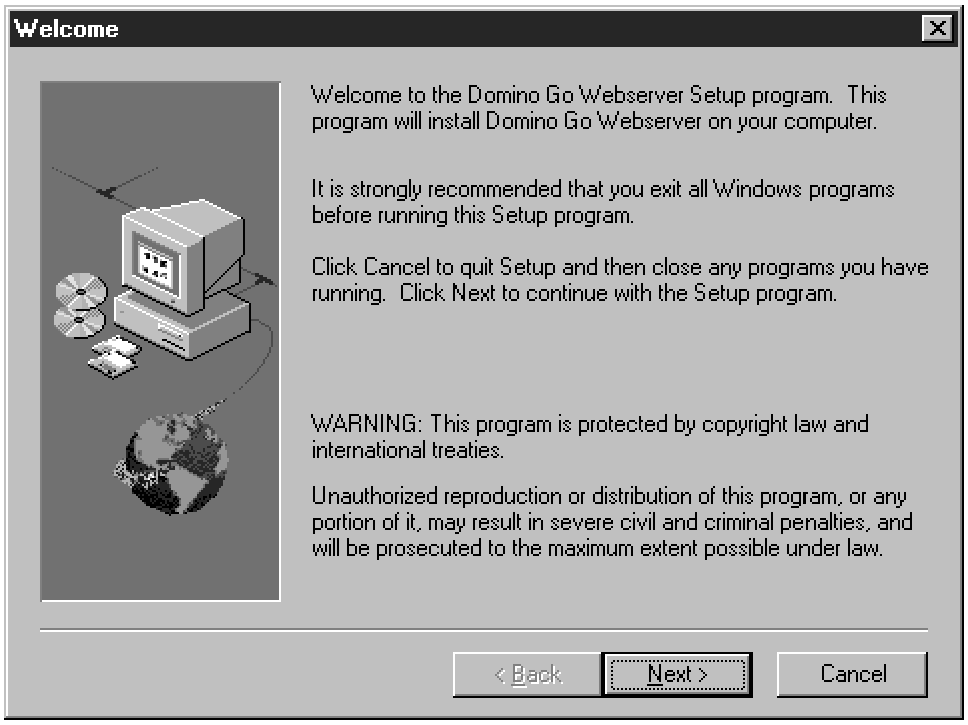
where *bin* substitutes the absolute path to the jdk1.1.5\bin directory. Therefore, if you keep the bin and lib directories at the same directory level (that is, if they have a common parent directory), the Java executable files will find the classes. You need to set the CLASSPATH variable only if you move classes.zip or want to load a different library such as one you developed. For example we needed to set the CLASSPATH when we installed ServletExpress, because this application comes with its own classes.

The CLASSPATH system environment variable can be created (if it does not already exist) or modified through the System Properties window, shown in Figure 275.

7.2 Lotus Domino Go Webserver 4.6.2.2

Installing Lotus Domino Go Webserver is very simple. You should only remember that you do not need to enable the Java Servlet support if you want to use the servlet engine provided by ServletExpress, as shown in Figure 277 on page 303.

We installed Lotus Domino Go Webserver on a Windows NT Server 4.0 machine where Service Pack Version 3 had been applied. After launching the installer program, we got the following window:



Welcome to the Domino Go Webserver Setup program. This program will install Domino Go Webserver on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup program.

Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

WARNING: This program is protected by copyright law and international treaties.

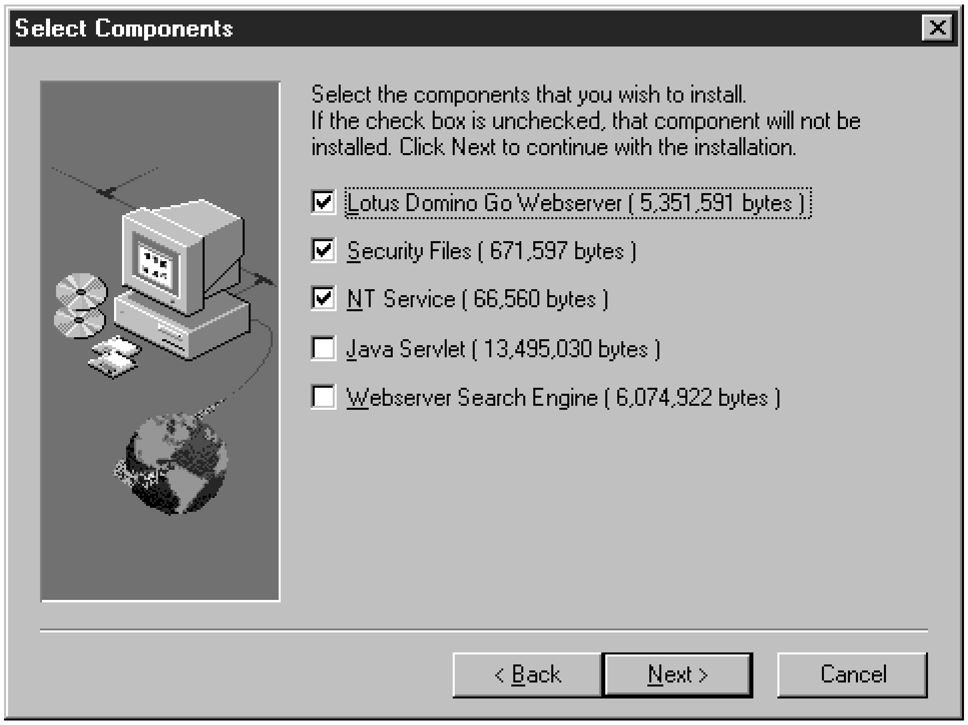
Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

Figure 276. Lotus Domino Go Webserver Installation Welcome Window

Beside the Web server itself, we installed the following components for Lotus Domino Go Webserver:

- Security Files
- NT Service

as shown in the following figure:



Select the components that you wish to install.
If the check box is unchecked, that component will not be installed. Click Next to continue with the installation.

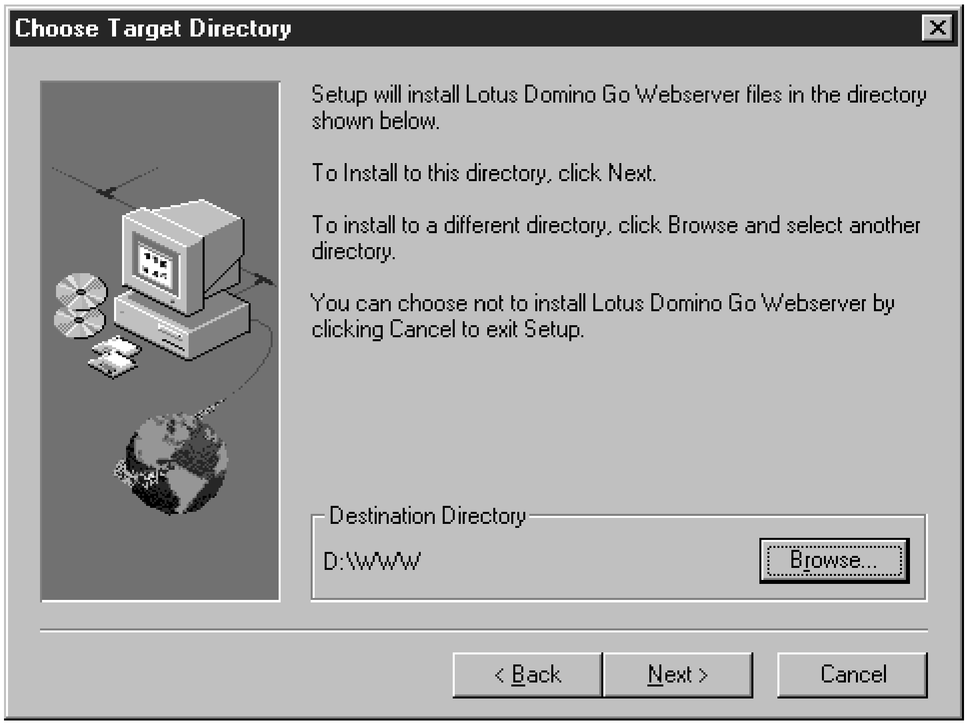
- [✓] Lotus Domino Go Webserver (5,351,591 bytes)
- [✓] Security Files (671,597 bytes)
- [✓] NT Service (66,560 bytes)
- [ ] Java Servlet (13,495,030 bytes)
- [ ] Webserver Search Engine (6,074,922 bytes)

Figure 277. Lotus Domino Go Webserver Installed Component List

The Security Files component was very important for our scenarios since it included the support for enabling the Web server to use the Secure Socket Layer (SSL) protocol. The NT Service component is also very important since it allows Lotus Domino Go Webserver to run as a Windows NT service. We will see that a consequence of this choice is, for example, that Lotus Domino Go Webserver can be started and stopped by interacting with the Services dialog for Windows NT (see Figure 287 on page 309).

We did not consider it necessary to install the Webserver Search Engine and also we did not need the Java Servlet component because we installed the more powerful servlet engine provided by IBM ServletExpress.

The Lotus Domino Go Webserver installation directory is usually named WWW and we had Setup create it on the D drive, as shown in the next dialog:


Choose Target Directory

Setup will install Lotus Domino Go Webserver files in the directory shown below.

To Install to this directory, click Next.

To install to a different directory, click Browse and select another directory.

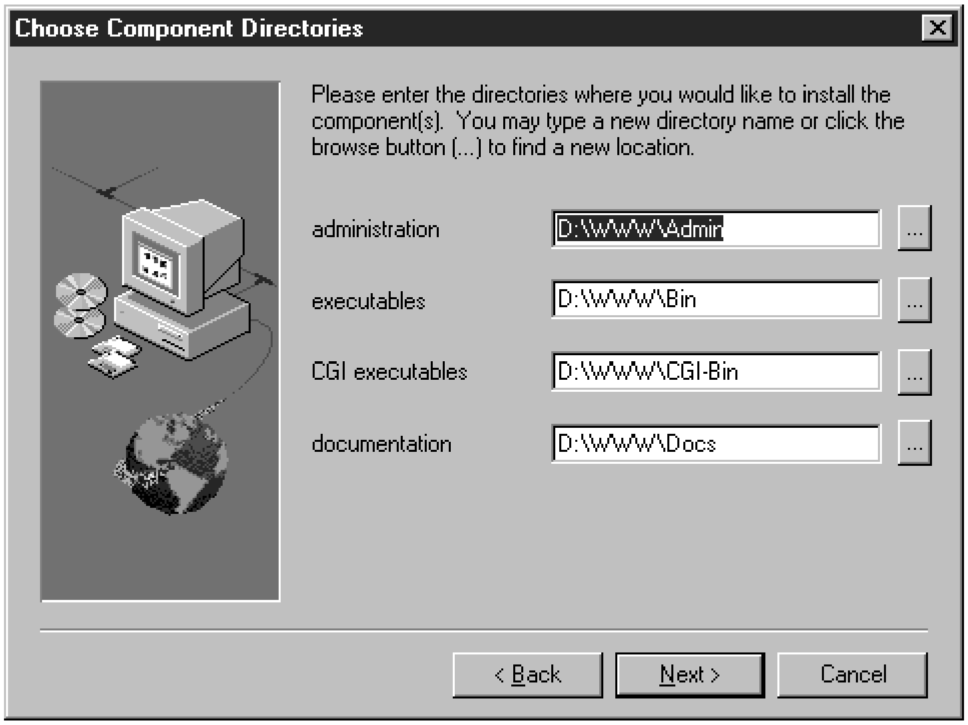
You can choose not to install Lotus Domino Go Webserver by clicking Cancel to exit Setup.

Destination Directory

D:\WWW

Figure 278. Choose Target Directory

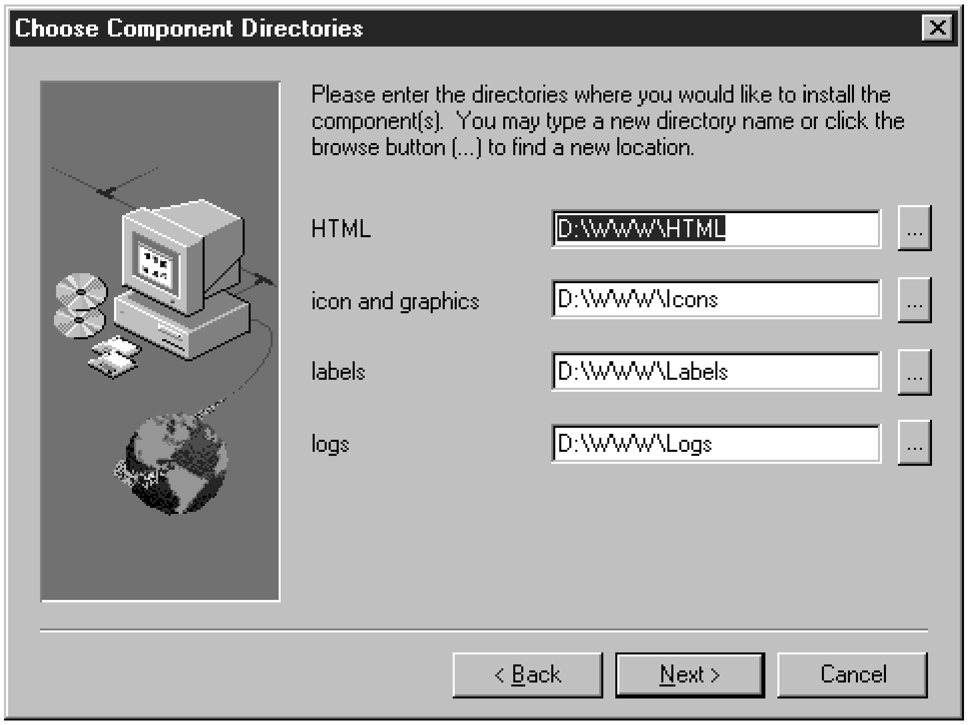
We could also confirm the directories where the specific components of Lotus Domino Go Webserver were installed, as shown in Figure 279 and Figure 280 on page 305.


Choose Component Directories

Please enter the directories where you would like to install the component(s). You may type a new directory name or click the browse button (...) to find a new location.

administration	D:\WWW\Admin
executables	D:\WWW\Bin
CGI executables	D:\WWW\CGI-Bin
documentation	D:\WWW\Docs

Figure 279. First Screen for Choosing Component Directories

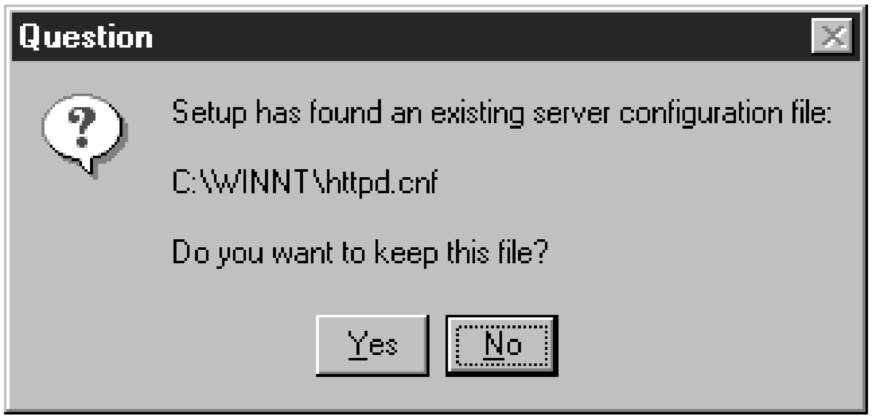

Choose Component Directories

Please enter the directories where you would like to install the component(s). You may type a new directory name or click the browse button (...) to find a new location.

HTML	D:\WWW\HTML
icon and graphics	D:\WWW\Icons
labels	D:\WWW\Labels
logs	D:\WWW\Logs

Figure 280. Second Screen for Choosing Component Directories

Since we had already installed Lotus Domino Go Webserver Version 4.6.1, we were informed by the installation program that existing configuration files httpd.cnf and ics_pics.cnf were found in the directory C:\WINNT, as Figure 281 and Figure 282 on page 306 show:

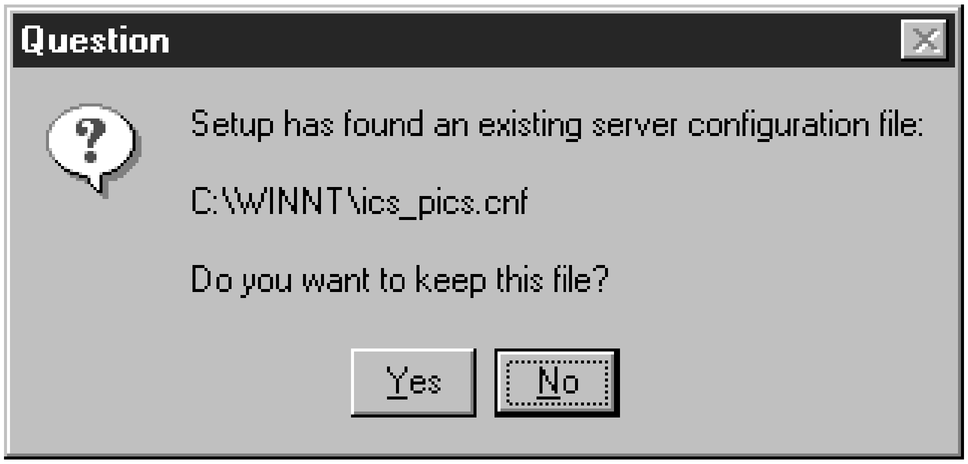

Question

Setup has found an existing server configuration file:

C:\WINNT\httpd.cnf

Do you want to keep this file?

Figure 281. Selecting the httpd.cnf Configuration File


Question

Setup has found an existing server configuration file:

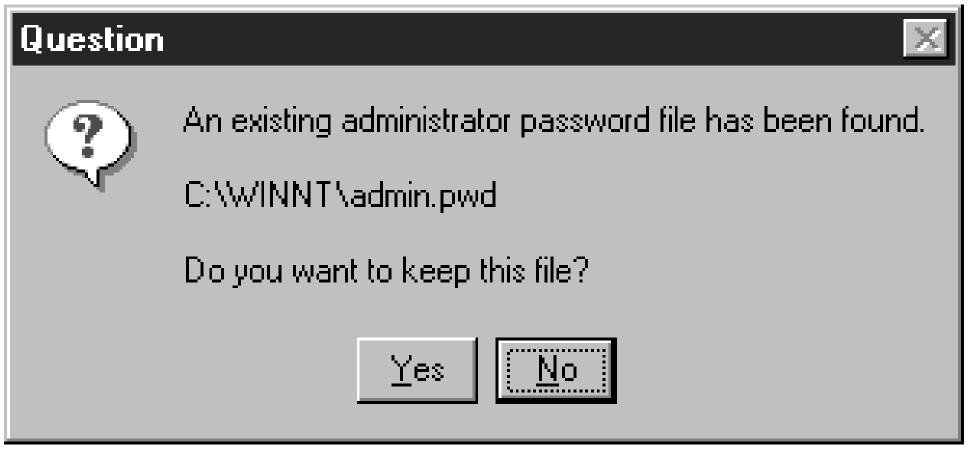
C:\WINNT\ics_pics.cnf

Do you want to keep this file?

Figure 282. Selecting the ics_pics.cnf Configuration File

We pressed the **No** button in both of the above dialogs, since we were installing the new Version 4.6.2.2 for Lotus Domino Go Webserver and we wanted to get its new configuration files. However you won't get the above screens if not upgrading a previous version of Lotus Domino Go Webserver.

Similarly, we were informed that an existing administrator password file, named admin.pwd, was found in the directory C:\WINNT, as you can see in the following dialog:


Question

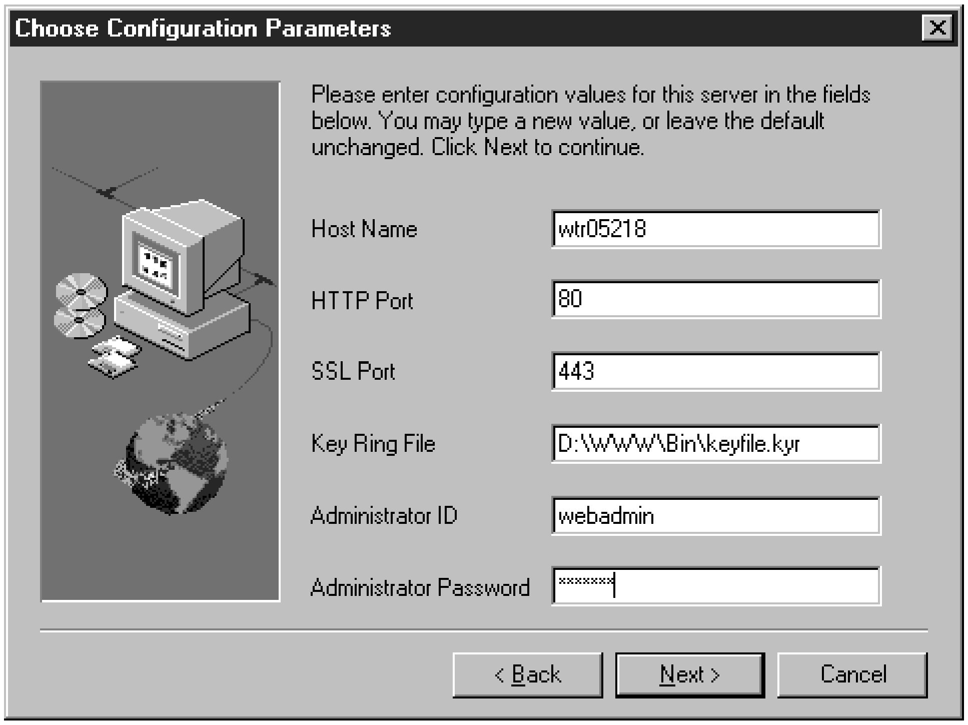
An existing administrator password file has been found.

C:\WINNT\admin.pwd

Do you want to keep this file?

Figure 283. Selecting the admin.pwd Administrator Password File

We pushed **No** in this case also and then we had to enter a user ID and a password for the Administrator of the Web server, as shown in the following window:


Choose Configuration Parameters

Please enter configuration values for this server in the fields below. You may type a new value, or leave the default unchanged. Click Next to continue.

Host Name	wtr05218
HTTP Port	80
SSL Port	443
Key Ring File	D:\WWW\Bin\keyfile.kyr
Administrator ID	webadmin
Administrator Password	*******

Figure 284. Choosing Configuration Parameters for Domino Go Webserver

Your decision about this password is very significant from a security point of view, because this password is an important string of characters that you use to protect the access to your Web server. You should select these characters carefully and never apply a simple string as your password. If you need to use the password in order to manage your Web server through the Internet, we recommend that you make an SSL connection using an `https://` URL between a control workstation and a Web server machine. The reason for our recommendation is that the password is transmitted encoded to the Web server, but not encrypted, and could be easily decoded by a Man-In-the-Middle. If you have already configured SSL, you can connect to your Web server securely by simply pointing your control workstation's Web browser to the secure URL https://*yourservername*, where *yourservername* is the fully qualified name of your Web server machine. The following figure shows that, in case you are communicating across a secure connection, a closed lock appears in the bottom-left corner of the Netscape browser window:

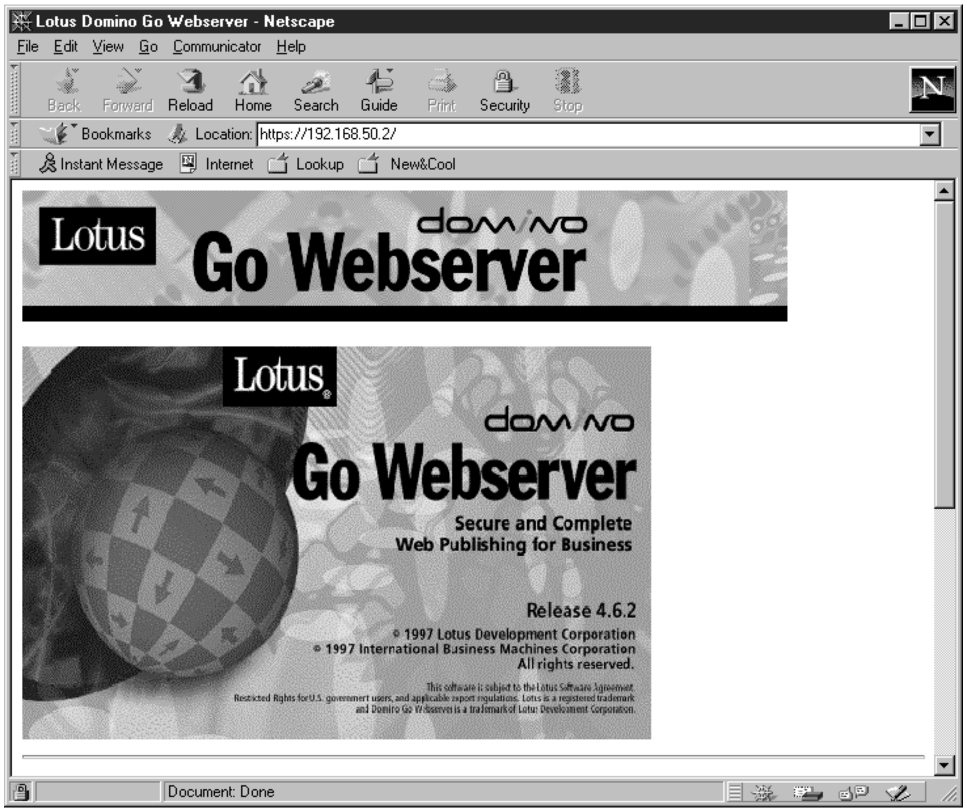
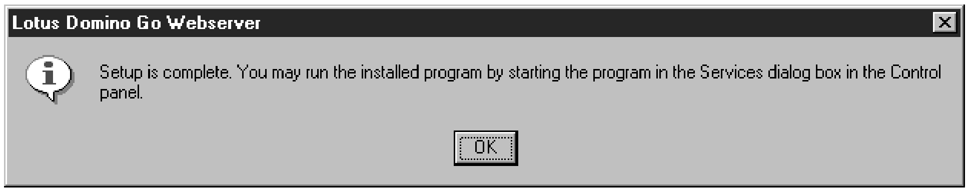

Lotus Domino Go Webserver - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: https://192.168.50.2/
Instant Message Internet Lookup New&Cool
Lotus
domino
Go Webserver
Secure and Complete
Web Publishing for Business
Release 4.6.2
© 1997 Lotus Development Corporation
© 1997 International Business Machines Corporation
All rights reserved.
Document: Done

Figure 285. SSL Icon

That lock is the security icon used by Netscape to inform the user that a secure SSL connection is in place. Otherwise that lock appears open. You can enforce further security by setting your Web server to request client authentication, as shown in 3.1.4, "SSL Client Authentication" on page 84.

At the end of the installation process, we were informed by the InstallShield application that Lotus Domino Go Webserver in the NT environment can be started and stopped using the Services dialog in the Control Panel, since it is a Windows NT service.



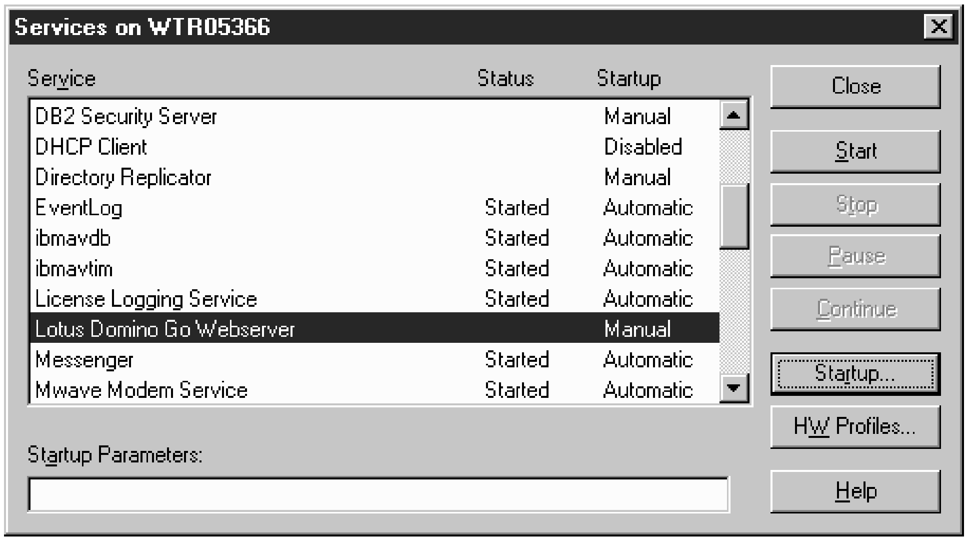
Setup is complete. You may run the installed program by starting the program in the Services dialog box in the Control panel.

Figure 286. Setup Indicates How to Start and Stop Lotus Domino Go Webserver

Another possibility would be to enter the following commands at an MS-DOS command prompt to start and stop Lotus Domino Go Webserver:

1. `net start "Lotus Domino Go Webserver"`
2. `net stop "Lotus Domino Go Webserver"`

However it is convenient to use the Services dialog at least once. In fact, after you open the Services dialog, you can select **Lotus Domino Go Webserver**, as shown in the following screen:

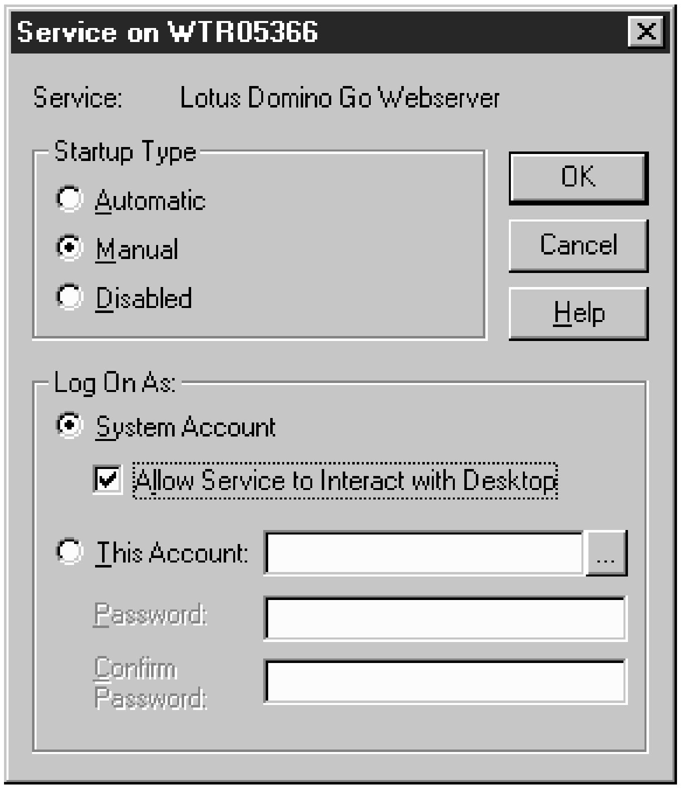

Services on WTR05366

Service	Status	Startup
DB2 Security Server		Manual
DHCP Client		Disabled
Directory Replicator		Manual
EventLog	Started	Automatic
ibmavdb	Started	Automatic
ibmavtim	Started	Automatic
License Logging Service	Started	Automatic
Lotus Domino Go Webserver		Manual
Messenger	Started	Automatic
Mwave Modem Service	Started	Automatic

Startup Parameters:

Figure 287. The Services Dialog on Windows NT Server

Then you click on the **Startup...** button and you get the configuration dialog for the Lotus Domino Go Webserver service. Check the box named **Allow Service to Interact with Desktop**, as this figure indicates:


Service on WTR05366

Service: Lotus Domino Go Webserver

Startup Type: Automatic [✓ Manual] Disabled

Log On As: [✓ System Account]

[✓] Allow Service to Interact with Desktop

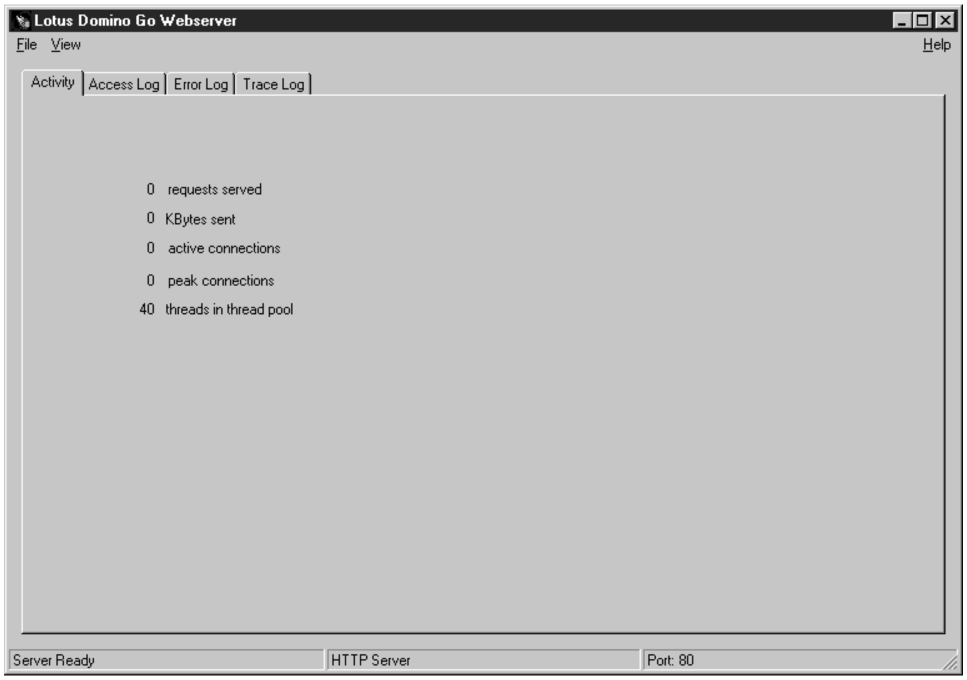
This Account:

Password:

Confirm Password:

Figure 288. Configuration Dialog for the Lotus Domino Go Webserver Service

By default, that box is not marked. If you select it, from now on, every time you start Lotus Domino Go Webserver, you will also get the following window:



0 requests served
0 KBytes sent
0 active connections
0 peak connections
40 threads in thread pool

Server Ready | HTTP Server | Port: 80

Figure 289. Lotus Domino Go Webserver Service Window

This window allows you to interact with the service. You can monitor activities, accesses, errors and traces by clicking on the corresponding tabs. You can also restart the Web server by clicking on **Restart** from the File menu. This will force Lotus Domino Go Webserver to read new configurations without completely shutting down the server.

Notice that selecting **Automatic** in the configuration dialog shown in Figure 288 on page 309 will cause Lotus Domino Go Webserver to start automatically every time the system starts.

At the end of the Lotus Domino Go Webserver installation, no modification needs to be made to the CLASSPATH system environment variable. That would be necessary only if we had installed the Java Servlet support provided by Lotus Domino Go Webserver. The file icsclass.zip contains all the Java class files necessary to compile basic Java servlets in the Domino Go Webserver environment. Notice that this file will come in the directory CGI-Bin, below the root directory WWW, even if you have not selected to install the Java Servlet support.

Lotus Domino Go Webserver can be configured to use the SSL protocol. Moreover it is also possible to enable SSL client authentication. We used both these security features for our scenarios. If you want to repeat our steps, you can refer to 3.1.2, “Lotus Domino Go Webserver SSL Setup” on page 71, 3.1.3, “Lotus Domino Go Webserver SSL Server Authentication” on page 79 and 3.1.4, “SSL Client Authentication” on page 84.

7.3 IBM ServletExpress 1.0

ServletExpress consists of a very powerful Java-based servlet engine that is independent of your Web server. It also provides a component named CORBA Support.

Note: As we specified in Chapter 4, “IBM WebSphere Application Server Security” on page 121, ServletExpress, at the time of this project, was still in its beta version. It has since been incorporated in the recently announced IBM WebSphere Application Server. However, no major modifications have been applied and you can safely refer to this section if you want to see how to install and configure this product. Only notice that the directory structure has been modified.

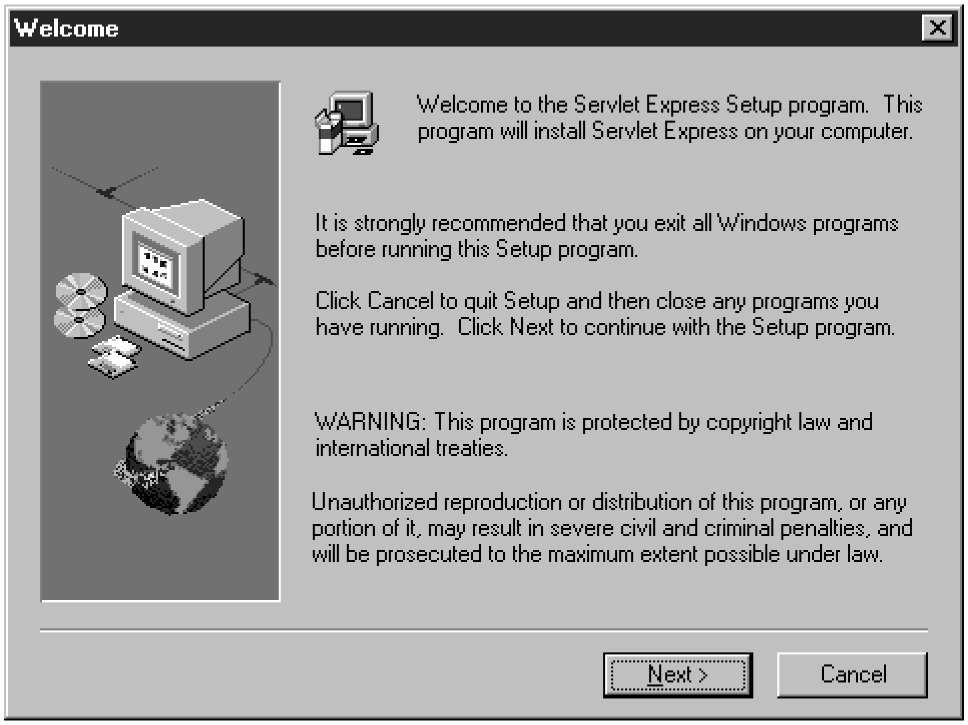
In our scenarios we did not use the basic servlet support that comes with Lotus Domino Go Webserver 4.6.2.2. We considered it better to use the full servlet support provided by ServletExpress 1.0. At the time of this project, the beta Release 2.1 for ServletExpress 1.0 was available. We also had the opportunity to use the beta 3 release, that was made available during the project.

7.3.1 Installation

We installed ServletExpress on the Windows NT 4.0 machine where we had already installed Lotus Domino Go Webserver. The installation of ServletExpress is not difficult. You should only remember two points:

1. Log on as Administrator or make sure that the account you are installing from has administrative privileges.
2. Shut down the Web server, if it is running.

As soon as you run the installer program for ServletExpress, you will get a Welcome screen.



Welcome to the Servlet Express Setup program. This program will install Servlet Express on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup program.

Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

Figure 290. ServletExpress Installation Welcome Window

Click on **Next** and you will get another dialog where you must select the plugin module which corresponds to your platform. We selected **Domino Go Webserver Version 4.6.1 or greater**, since we wanted to integrate the servlet engine provided by ServletExpress with Domino Go Webserver Version 4.6.2.2.

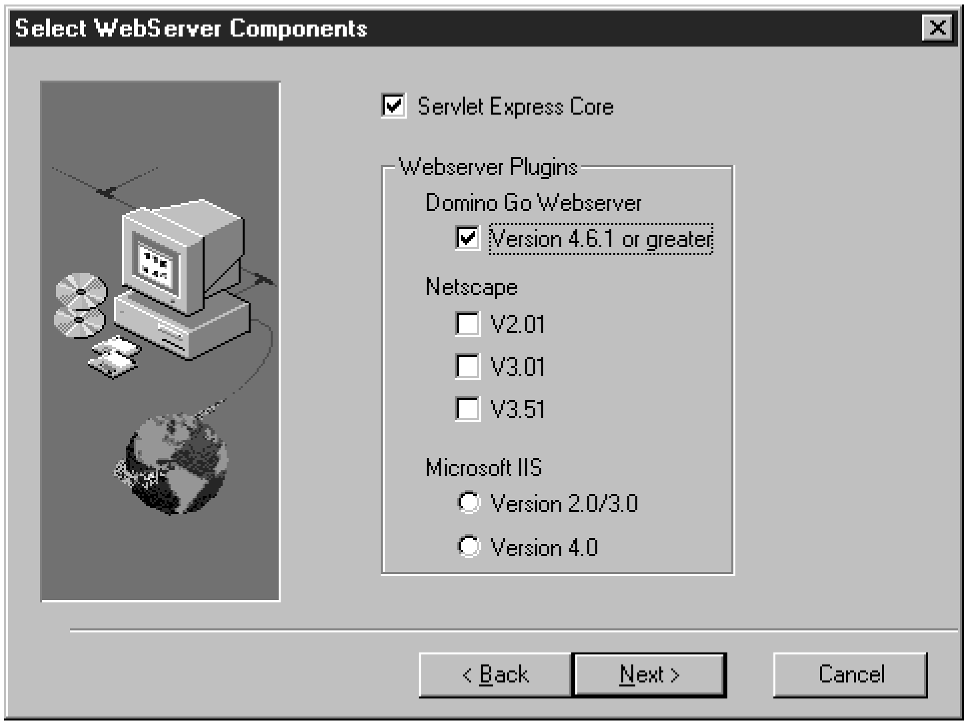
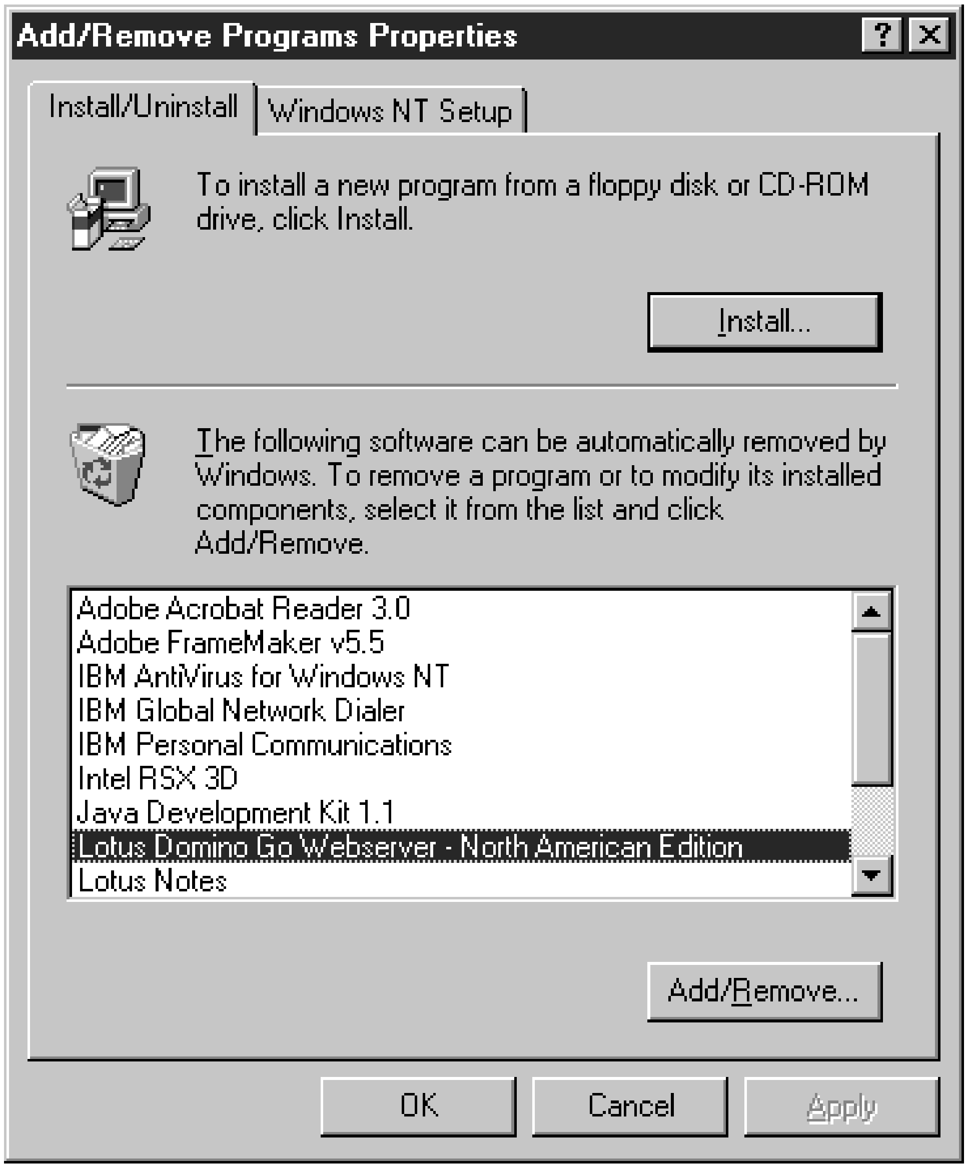


Figure 291. Webserver Plugin Selection

It is very important that no other servlet support is active when you install ServletExpress. If you had previously installed the Java Servlet component for Lotus Domino Go Webserver (see Figure 277 on page 303), you have to uninstall it before running the setup program for ServletExpress. You have two choices:

1. Open the Add/Remove Program Properties window from the Control Panel, select **Lotus Domino Go Webserver** and click on the **Add/Remove** button, as shown in the following screen:


Add/Remove Programs Properties

Install/Uninstall | Windows NT Setup

To install a new program from a floppy disk or CD-ROM drive, click Install.

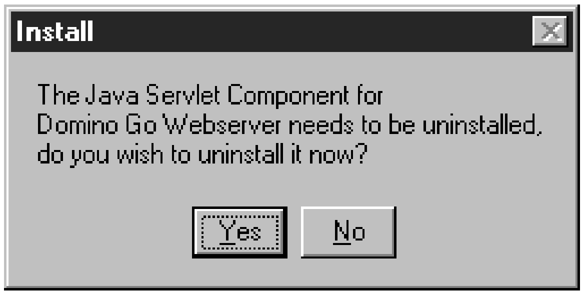
The following software can be automatically removed by Windows. To remove a program or to modify its installed components, select it from the list and click Add/Remove.

Adobe Acrobat Reader 3.0
Adobe FrameMaker v5.5
IBM AntiVirus for Windows NT
IBM Global Network Dialer
IBM Personal Communications
Intel RSX 3D
Java Development Kit 1.1
Lotus Domino Go Webserver - North American Edition
Lotus Notes

Figure 292. Remove the Domino Go Webserver Java Servlet Component

Then you should remove only the Java Servlet component for Lotus Domino Go Webserver. This operation can be accomplished successfully only if Lotus Domino Go Webserver is currently not running.

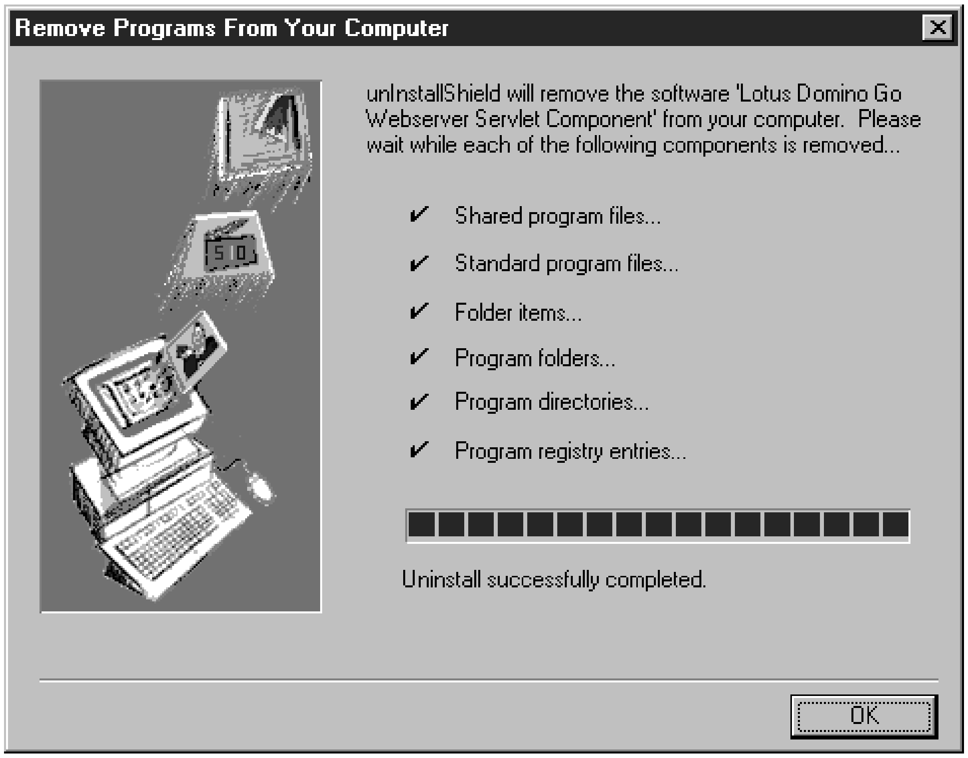
2. Launch the ServletExpress installer program and go on until it detects by itself the Java Servlet component for Domino Go Webserver. The InstallShield will then prompt you with the uninstallation process for the Java Servlet component for Domino Go Webserver:


Install

The Java Servlet Component for
Domino Go Webserver needs to be uninstalled,
do you wish to uninstall it now?

Figure 293. Uninstalling Domino Go Webserver Java Servlet Component

Both these processes should let you successfully uninstall the Java Servlet Component for Lotus Domino Go Webserver:


Remove Programs From Your Computer

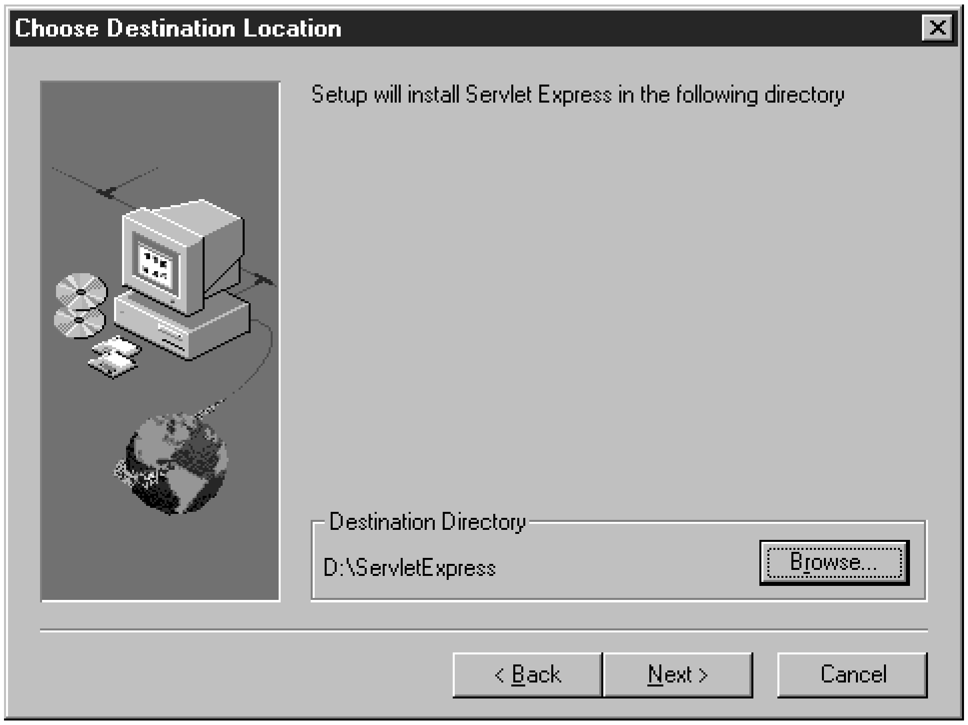
unInstallShield will remove the software 'Lotus Domino Go Webserver Servlet Component' from your computer. Please wait while each of the following components is removed...

- ✓ Shared program files...
- ✓ Standard program files...
- ✓ Folder items...
- ✓ Program folders...
- ✓ Program directories...
- ✓ Program registry entries...

Uninstall successfully completed.

Figure 294. Completing Domino Go Webserver Servlet Component Uninstallation

We were allowed to confirm the directories where the ServletExpress files had to be installed and the program folder where the InstallShield had to add the ServletExpress icon.

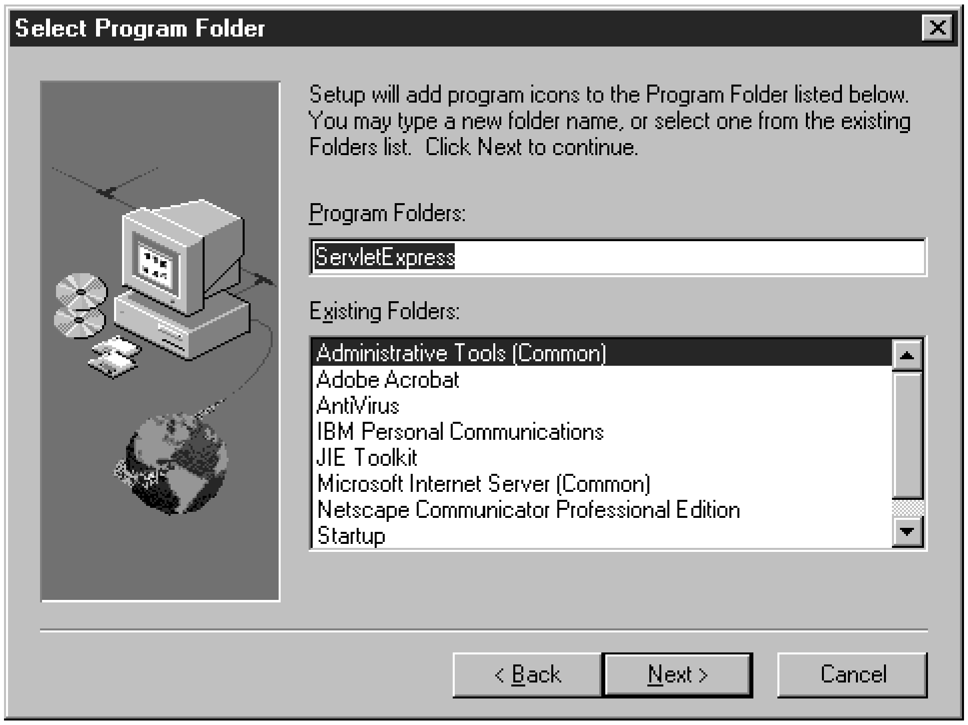

Choose Destination Location

Setup will install Servlet Express in the following directory

Destination Directory

D:\ServletExpress

Figure 295. Choose Destination Directory for ServletExpress Installation


Select Program Folder

Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing Folders list. Click Next to continue.

Program Folders:

ServletExpress

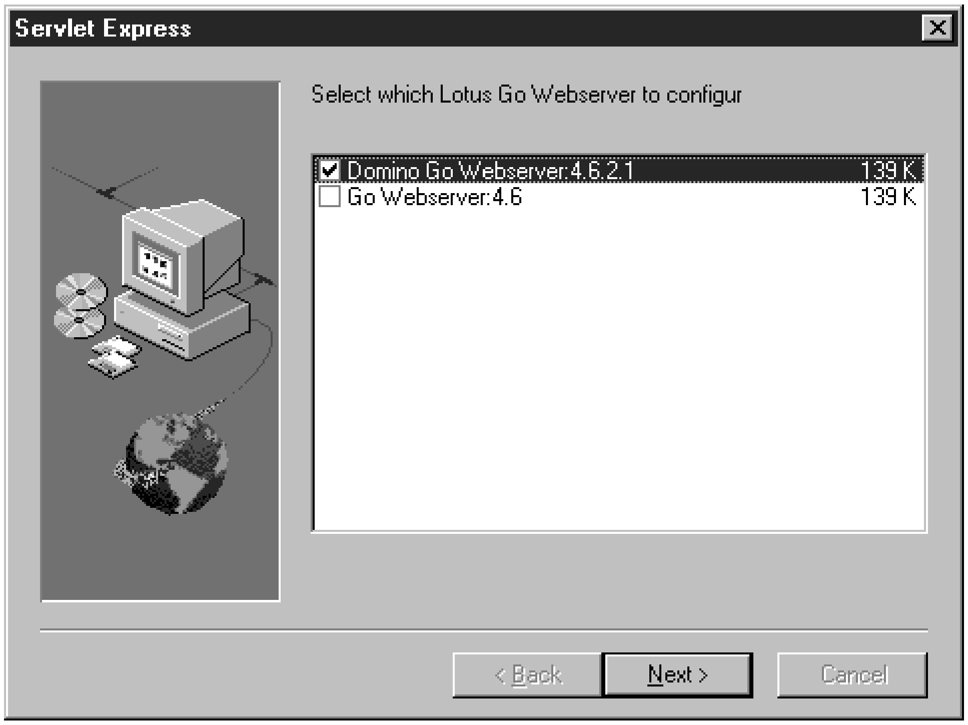
Existing Folders:

- Administrative Tools (Common)
- Adobe Acrobat
- AntiVirus
- IBM Personal Communications
- JIE Toolkit
- Microsoft Internet Server (Common)
- Netscape Communicator Professional Edition
- Startup

Figure 296. Select ServletExpress Program Folder

We installed the ServletExpress files in the directory D:\ServletExpress. Then we clicked on **Next** to go ahead with the installation accepting the default ServletExpress program folder.

You should also select the version of Lotus Domino Go Webserver currently installed on your machine. We selected **Domino Go Webserver 4.6.1**.



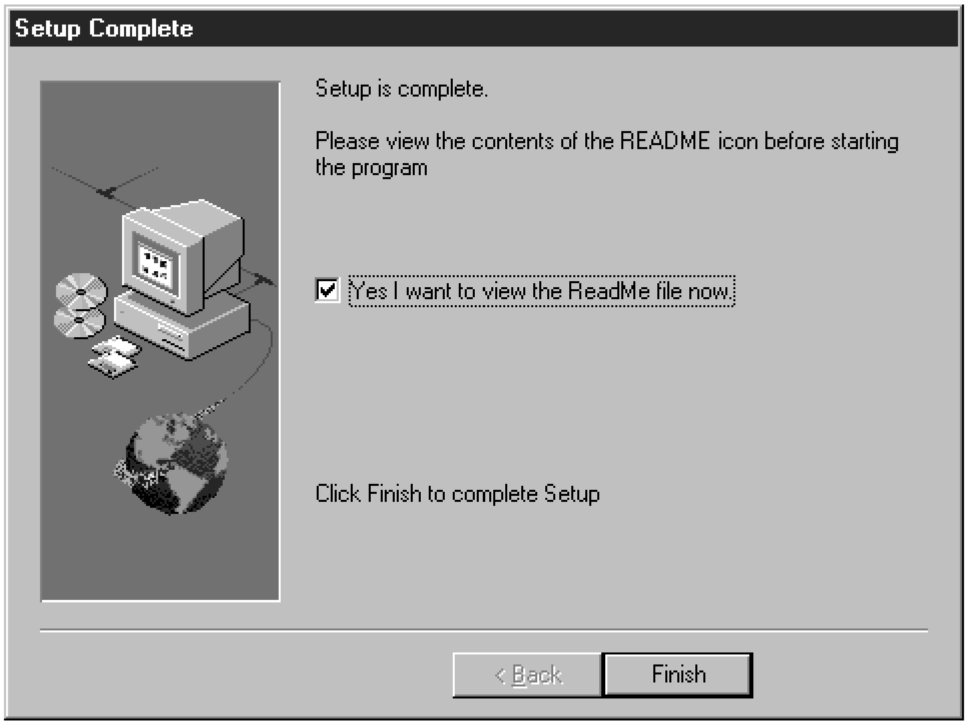
Servlet Express

Select which Lotus Go Webserver to configur

✓ Domino Go Webserver:4.6.2.1	139 K
☐ Go Webserver:4.6	139 K

Figure 297. Select the Exact Version of Lotus Domino Go Webserver

Be sure to read the README file before running the program. The InstallShield application will prompt you with this possibility at the end of the installation.



Setup Complete

Setup is complete.

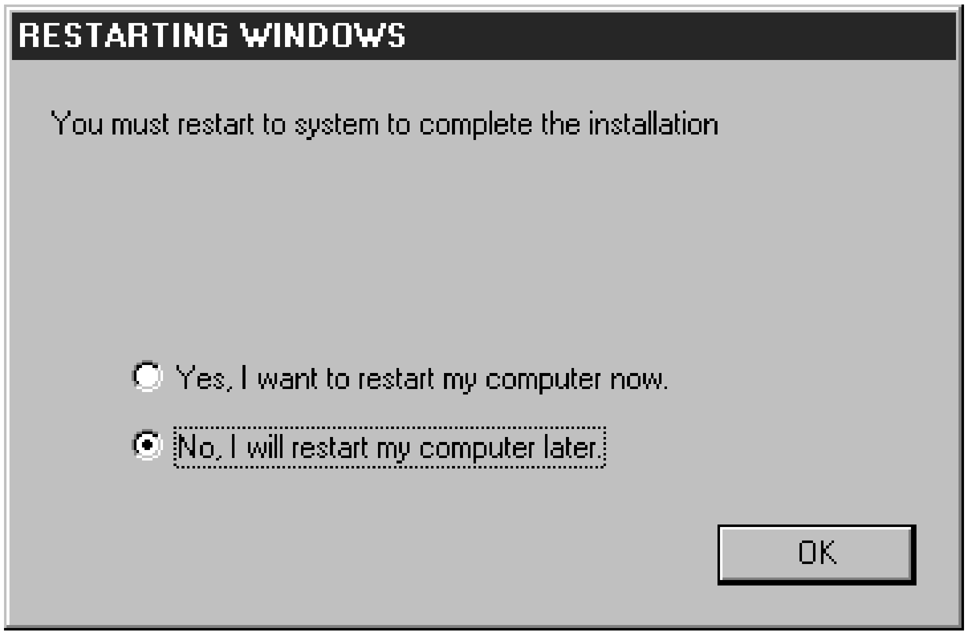
Please view the contents of the README icon before starting the program

✓ Yes I want to view the ReadMe file now.

Click Finish to complete Setup

Figure 298. ServletExpress Setup Complete Window - Read the README file!

Then restart your computer before running ServletExpress.


RESTARTING WINDOWS

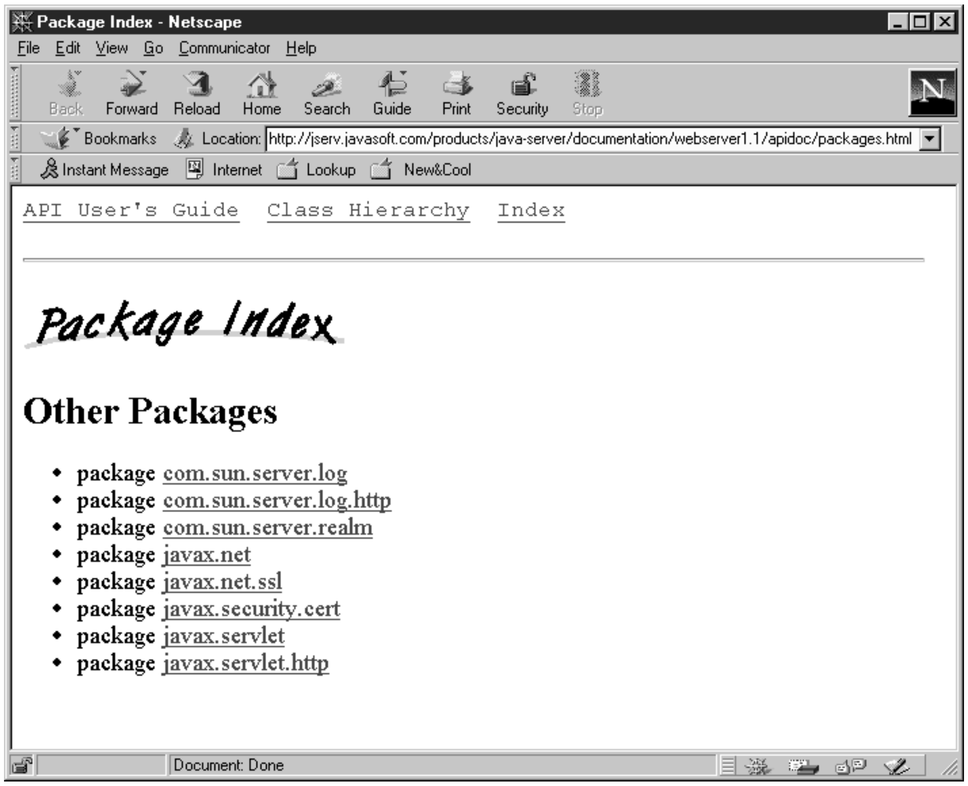
You must restart to system to complete the installation

Yes, I want to restart my computer now.
[✓ No, I will restart my computer later.]
OK

Figure 299. Restarting Windows NT After the Installation

The README file will provide you with very important information related to known defects and configuration tips. At the time of this project, ServletExpress Version 1.0 was available only in beta release. We discovered the following problems in the beta 2.1 release. However these problems will not be present in future releases of the product.

1. ServletExpress Manager is the user interface for managing servlets. In Windows NT, if you use a text editor instead of the ServletExpress Manager to edit the properties files, you should use WordPad or the `edit` command line utility instead of Notepad. If you use Notepad and save the files, you might not be able to later use ServletExpress Manager to edit the same files.
2. If the ServletExpress Manager is running when you try to run a servlet, the Domino Go Webserver will core dump. To avoid this problem, do not run the ServletExpress Manager and servlets at the same time. After you finish using the ServletExpress Manager, be sure to log off before you run any servlets.
3. If you want to develop servlets, you will need to use the Java Servlet Development Kit (JSDK) Version 1.1 class files. ServletExpress includes those class files in its libraries, but you should refer to the JavaSoft Web site if you want to obtain the related documentation. The following figure shows how you can retrieve it:


Package Index - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://jserv.javasoft.com/products/java-server/documentation/webserver1.1/apidoc/packages.html
Instant Message Internet Lookup New&Cool
Document: Done

API User's Guide Class Hierarchy Index

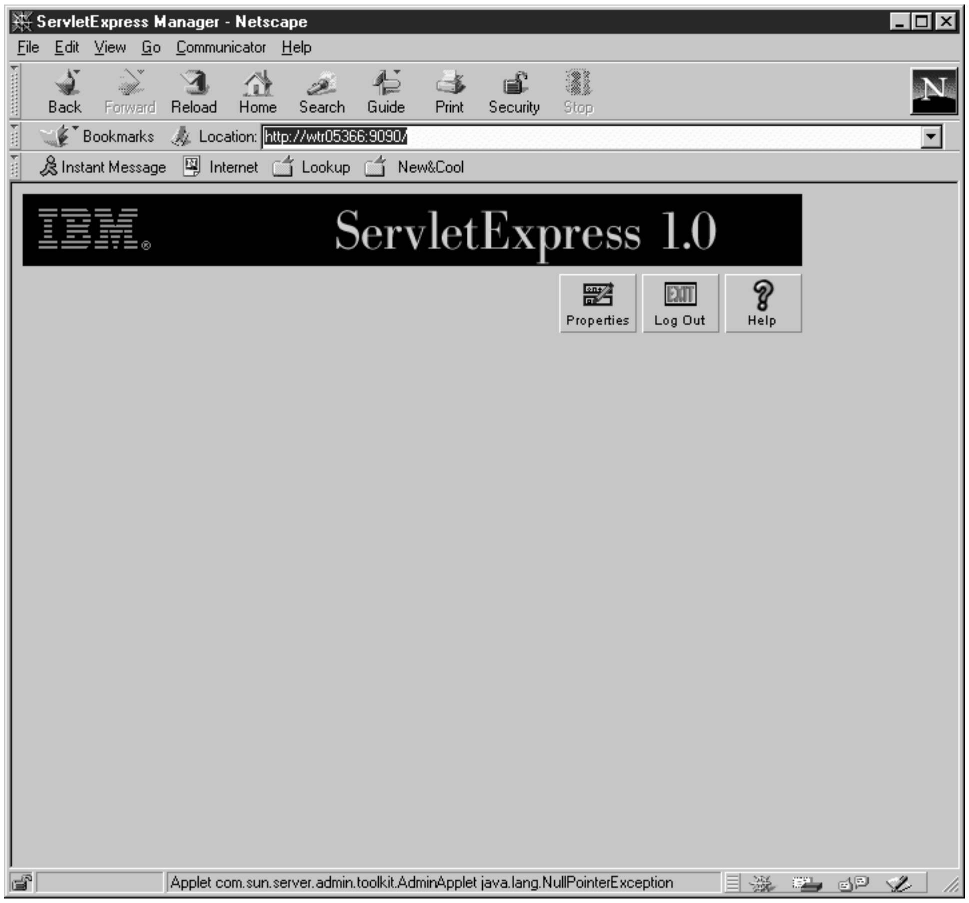
Package Index

Other Packages

- **package** com.sun.server.log
- **package** com.sun.server.log.http
- **package** com.sun.server.realm
- **package** javax.net
- **package** javax.net.ssl
- **package** javax.security.cert
- **package** javax.servlet
- **package** javax.servlet.http

Figure 300. JSDK Class Files Documentation

4. To run the ServletExpress Manager applet, you simply point your Java-enabled Web browser to the URL http://*yourWebserver*:9090/, where *yourWebserver* is the fully qualified name of your Web server machine and 9090 is the default port for the ServletExpress Manager. You will need an Applet Viewer or a Web browser that fully supports JDK 1.1. Netscape Communicator 4.04 does not have a full JDK 1.1 support and if you try to run the ServletExpress Manager, after entering the user ID and password for the administrator, a java.lang.NullPointerException will be thrown.


ServletExpress Manager - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05366:9090/
Instant Message Internet Lookup New&Cool
IBM
ServletExpress 1.0
Properties
Log Out
Help

Applet com.sun.server.admin.toolkit.AdminApplet java.lang.NullPointerException

Figure 301. NullPointerException Thrown by the ServletExpress Manager

If you want to run the ServletExpress Manager with Netscape Communicator 4.04, you should upgrade the JVM level in the browser, as indicated in 5.3, “Interacting with Signed Java Applets” on page 224. In order to upgrade the JVM level in Netscape Communicator 4.04, you could also download the 404awt.zip file from Netscape's Web site http://developer.netscape.com/software/jdk/download.html and unzip it as indicated in the same Web page. However you will not need to install any upgrade if you use a later version of Netscape Communicator (we used Version 4.05) as well as Microsoft Internet Explorer Version 4.0 or Sun HotJava Version 1.1.

7.3.2 The CLASSPATH System Environment Variable

As soon as you install ServletExpress, you should update the value for the CLASSPATH system environment variable, so that it includes all the class files necessary to develop advanced servlets using the powerful servlet engine provided by ServletExpress. The following window shows what value the CLASSPATH environment variable had in our environment:

```
.;
D:\jdk1.1.5\lib\classes.zip;
D:\ServletExpress\lib\servexp.jar;
D:\ServletExpress\lib\jst.jar;
D:\ServletExpress\lib\jsdk.jar;
D:\ServletExpress\lib\x509v1.jar;
D:\ServletExpress\lib;
D:\ServletExpress\web\classes\ibmjbrt.jar;
D:\WWW\CGI-Bin\icsclass.zip;
D:\ServletExpress\servlets;
```

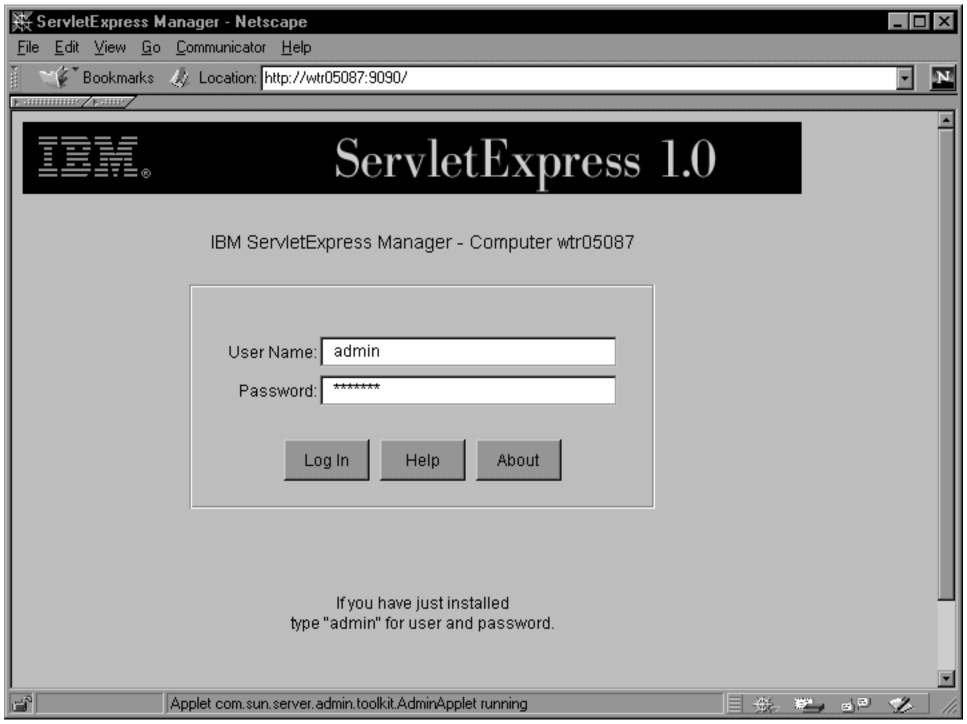
Figure 302. Current Settings for the CLASSPATH Environment Variable

The zip file icsclass.zip could also be removed from the CLASSPATH. It was used to develop servlets using the servlet engine provided by Lotus Domino Go Webserver. If you want to keep it in the CLASSPATH variable, be sure that it is placed after the ServletExpress JAR files.

When we upgraded JDK 1.1.5 to the next release 1.1.6, we edited the CLASSPATH variable again, changing the second entry to `D:\jdk1.1.6\lib\classes.zip`.

7.3.3 How to Change Administrator Password

The first panel you receive after loading the ServletExpress Manager is the login. The default Administrator ID is admin and the default password is admin. You can see this information noted on the following screen:


ServletExpress Manager - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://wtr05087:9090/
IBM
ServletExpress 1.0
Log In
Help
About

IBM ServletExpress Manager - Computer wtr05087

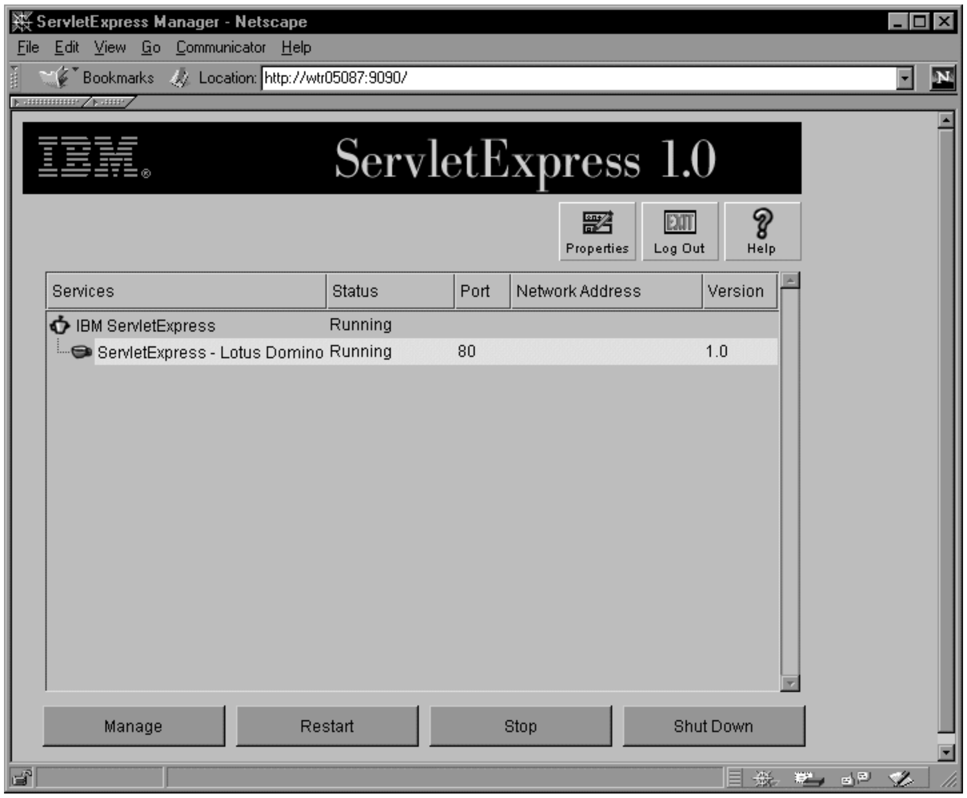
User Name: admin
Password: *******

If you have just installed
type "admin" for user and password.

Applet com.sun.server.admin.toolkit.AdminApplet running

Figure 303. ServletExpress Manager Login

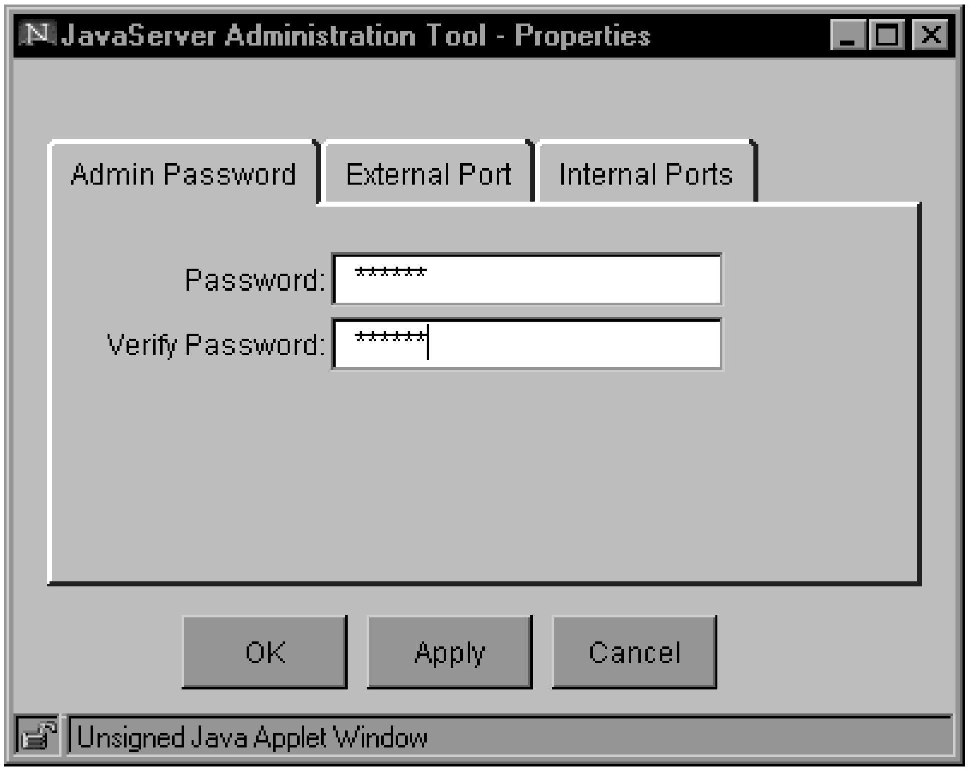
After the login is complete, the main panel for the ServletExpress Manager is shown. This panel shows ServletExpress running on Lotus Domino Go Webserver.


ServletExpress Manager - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://wtr05087:9090/
IBM
ServletExpress 1.0
Properties
Log Out
Help
Manage
Restart
Stop
Shut Down

Services	Status	Port	Network Address	Version
IBM ServletExpress	Running			
ServletExpress - Lotus Domino	Running	80		1.0

Figure 304. ServletExpress Manager Main Panel

The first thing to do after accessing the ServletExpress Manager should be to change the default administrator password from admin to something private. This is not required by the application, but it is good security practice. It is very dangerous to continue to use the default value especially if your environment is connected to the Internet. Whoever has access as the ServletExpress administrator can manage your servlets. You get a dialog to change the password by pressing the **Properties** button at the top of the screen shown in Figure 304.


JavaServer Administration Tool - Properties

Admin Password | External Port | Internal Ports

Password: ******

Verify Password: ******

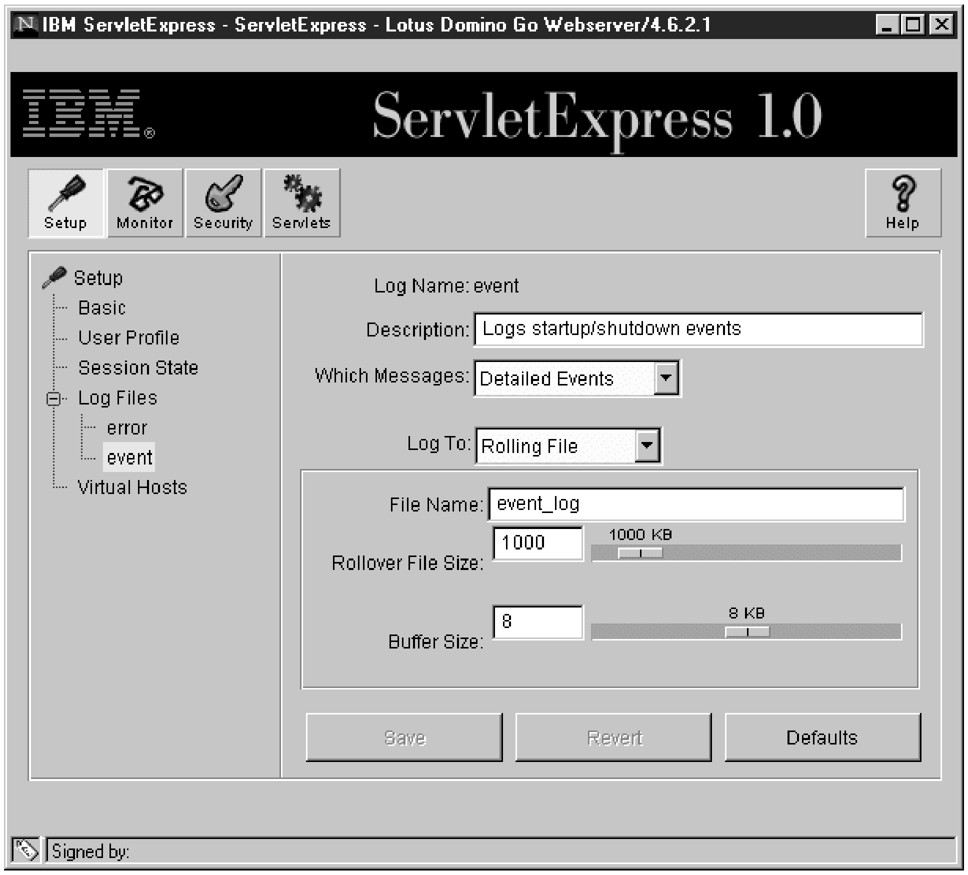
Figure 305. ServletExpress Manager Properties Panel

7.3.4 How to Configure ServletExpress Log Files

ServletExpress gives you the ability to view all servlet events and errors through two types of log files:

- The event log lists server tracing events (for example, a startup or shutdown) and servlet log information.
- The error log lists internal server errors and is useful for troubleshooting.

Event and error log files parameters can be configured through the Setup page of the ServletExpress Manager. The following two screens show how you can configure the kind of messages you want to log, how you can store and display the log files and how you can set the file size limit for each log file:


IBM ServletExpress - ServletExpress - Lotus Domino Go Webserver/4.6.2.1

ServletExpress 1.0

Log Name: event

Description: Logs startup/shutdown events

Which Messages: Detailed Events

Log To: Rolling File

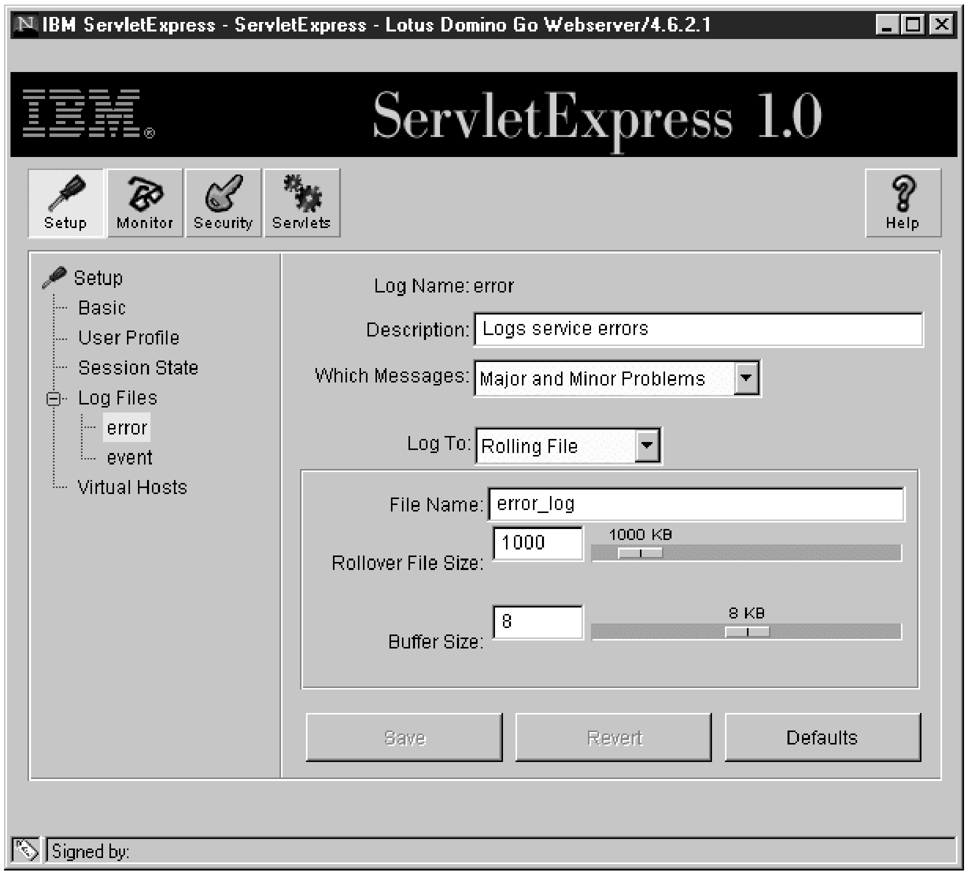
File Name: event_log

Rollover File Size: 1000 — 1000 KB

Buffer Size: 8 — 8 KB

Signed by:

Figure 306. How to Configure the Event Log File Parameters



Log Name: error
Description: Logs service errors
Which Messages: Major and Minor Problems
Log To: Rolling File
File Name: error_log
Rollover File Size: 1000 (1000 KB)
Buffer Size: 8 (8 KB)

Figure 307. How to Configure the Error Log File Parameters

The service log directories are all stored in the directory *servletexpress_root*\logs, *servletexpress_root* being the root directory where you installed ServletExpress (in our case it was D:\ServletExpress).

The ServletExpress engine supports two other types of logging:

1. Native DLL logging: log messages produced by the Web server C code before entering Java.
2. Java standard out logging: any System.out and System.err prints go to this log.

These log files are very helpful for troubleshooting, but logging is not enabled by default. For this release of ServletExpress, you must manually edit the jvm.properties file, stored in the *servletexpress_root*\properties\server\ServletExpress\servletservice directory, to control these two types of logging.

How to Edit the jvm.properties File

As we have already said, if the platform where you installed ServletExpress is Windows NT, you must be sure to use WordPad or the command line utility `edit` instead of Notepad to edit the jvm.properties files.

To control native DLL logging:

1. Open the jvm.properties file.

2. To enable logging:
   - Set the property `ncf.native.logison=true` (notice that by default this property is set to `false`).
   - Set the property `ncf.native.logfile=`*servletexpress_root*`\logs\native.log`.
   - Restart your Web server.
3. To disable logging:
   - Set the property `ncf.native.logison=false` (default value).
   - Restart your Web server.

To control Java standard out logging:

1. Open the jvm.properties file.
2. To enable logging to a file:
   - Set the property `ncf.jvm.stdoutlog.enabled=true` (by default it would be set to `false`).
   - Set the property `ncf.jvm.stdoutlog.file=true` (by default this property also is set to `false`).
   - Set the property `ncf.jvm.stdoutlog.filename=`*servletexpress_root*`\logs\ncf.log`.
3. To enable logging to a Java console window:
   - Set the property `ncf.jvm.stdoutlog.enabled=true` (default would be `false`).
   - Set the property `ncf.jvm.stdoutlog.file=false` (default value).
4. To disable logging:
   - Set the property `ncf.jvm.stdoutlog.enabled=false` (default value).
   - Set the property `ncf.jvm.stdoutlog.file=false` (default value).

The following figure shows the jvm.properties file after we have modified it to enable native DLL logging to the log file D:-ServletExpress\logs\native.log and Java standard out logging to the log file D:\ServletExpress\logs\ncf.log. We have also modified the ncf.jvm.classpath variable to the value that we will need to run the applications of the scenarios that we describe in Chapter 8, “Three-Tier Applications in Firewall-Protected Network Environments” on page 361 and Chapter 9, “IIOP in Firewall-Protected Network Environments” on page 435. The items that we changed are highlighted with numbers from **1** to **6**.

```
# System Properties
ServletExpressVersion=1.0.0
server.root=D:/ServletExpress/
server.name=ServletExpress
java.compiler=symcjit

# NCF Properties
ncf.service.name=servletservice
ncf.service.class=com.ibm.ServletExpress.service.SEServlet
ncf.plugin.classname=com.ibm.ServletExpress.ServletSystem

# Logging is off by default
 1  ncf.native.logison=true

#
# Turn logging off by default
#
 2  ncf.jvm.stdoutlog.enabled=true
 3  ncf.jvm.stdoutlog.file=true

# NCF - Admin Service Properties for BasicNCFConfig Applet
 4  ncf.jvm.classpath=D:\jdk1.1.5\lib\classes.zip;
                             D:\ServletExpress\lib\servexp.jar;
                             D:\ServletExpress\lib\jst.jar;
                             D:\ServletExpress\lib\jsdk.jar;
                             D:\ServletExpress\lib\x509v1.jar;
                             D:\ServletExpress\web\classes\ibmjbrt.jar;
                             D:\ServletExpress\web\classes\ibmjbde.jar;
                             D:\WWW\CGI-Bin\icsclass.zip;
                             D:\ServletExpress\servlets;

ncf.jvm.libpath=D:\jdk1.1.5\bin
ncf.jvm.path=D:\jdk1.1.5\bin
ncf.jvm.use.system.classpath=false

# Max Java Heap Size
ncf.jvm.mx=67108864

#
# Properties for Netscape webserver V2.01 on AIX or SOLARIS
#
#ncf.native.outofproc.runscript=/usr/bin/servlet_eng_runner.sh
#ncf.native.outofproc.port=8090
#ncf.native.outofproc.idstring="servexp"
#ncf.native.outofproc.netscapemime=<netscape_root>/config/mime.ty        pes

#
# Properties for Apache webserver on AIX or SOLARIS
#
#ncf.native.apache.outofproc.runscript=/usr/bin/apache_servlet_en        g_runner
#ncf.native.apache.outofproc.port=8082
#ncf.native.apache.outofproc.idstring="apache-servlet-engine"
```

Figure 308 (Part 1 of 2). jvm.properties File

```
# Properties for IIS
ncf.native.iis.extensionloc=/sePlugins/iis20.dll

# Properties for Domino Go
ncf.native.httpd.cnf.path=c:\rsant\etc\httpd.cnf
5 ncf.native.logfile=D:\ServletExpress\logs\native.log
6 ncf.jvm.stdoutlog.filename=D:\ServletExpress\logs\ncf.log
```

Figure 308 (Part 2 of 2). jvm.properties File

When we upgraded JDK 1.1.5 to the next release 1.1.6, we edited the jvm.properties file again, changing all the occurrences of `jdk1.1.5` to `jdk1.1.6`.

7.3.5 How to Add Servlets into ServletExpress

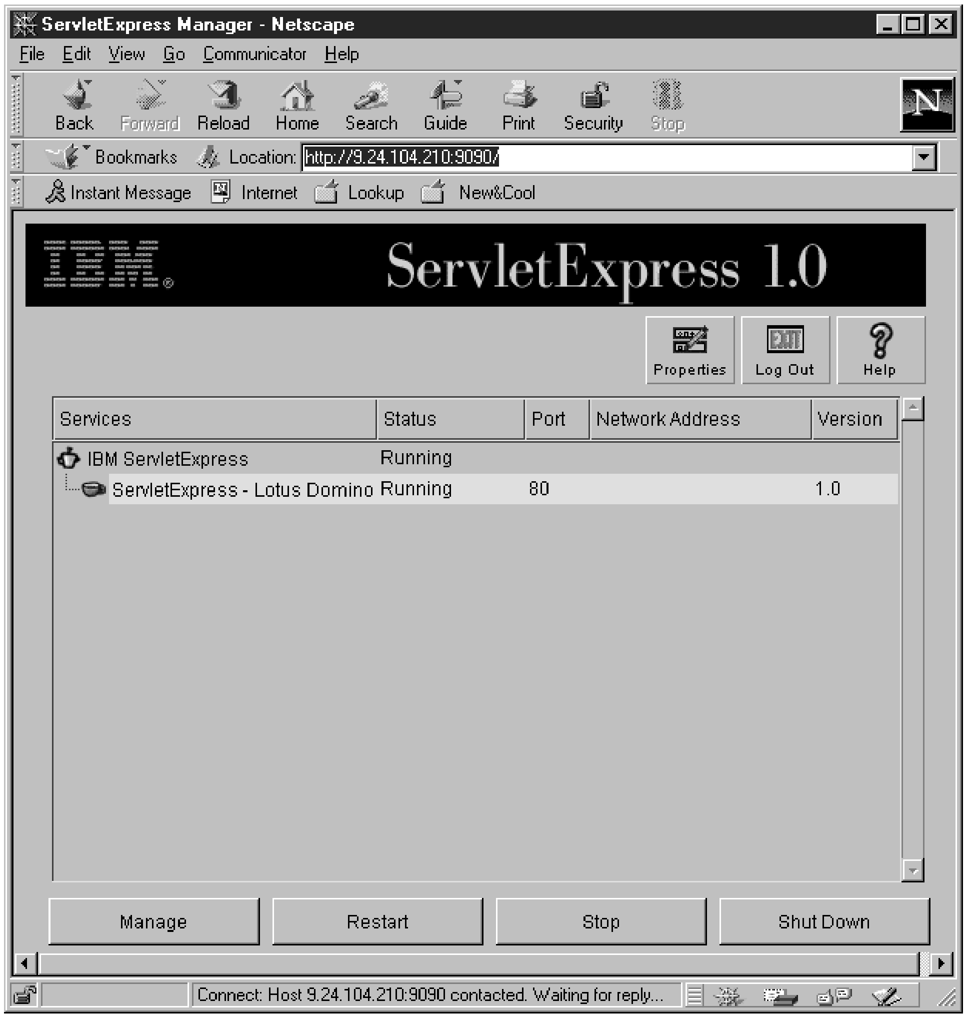
Before you can use a servlet, you must copy or move its class file in the servlets directory, that is *servletexpress_root*\servlets.

Then, using your Web browser, you can access the management page of ServletExpress by pointing your browser to the following URL: http://*yourservername*:9090/, *yourservername* being the fully qualified name of your Web server machine.

The ServletExpress Manager applet starts and displays the login page (see Figure 303 on page 320).

If you have just installed ServletExpress you should simply type `admin` in the User Name and Password fields and then click the **Log in** button. We have already mentioned, however, that you should change the administrator password the first time you log in, even if this operation is not mandatory. One very common attack is to access using default installation user IDs and passwords.

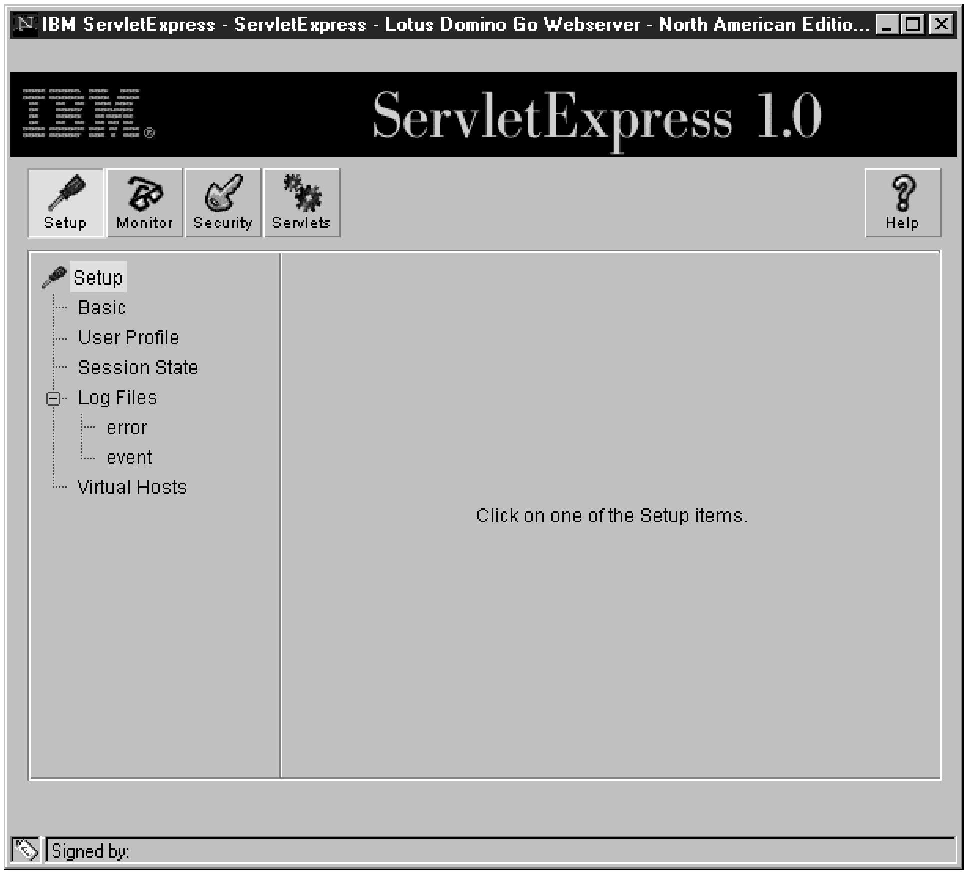
After you click the **Log in** button, you will get the page below:


ServletExpress Manager - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://9.24.104.210:9090/
Instant Message Internet Lookup New&Cool
ServletExpress 1.0
Properties Log Out Help
Manage Restart Stop Shut Down
Connect: Host 9.24.104.210:9090 contacted. Waiting for reply...

Services	Status	Port	Network Address	Version
IBM ServletExpress	Running			
ServletExpress - Lotus Domino	Running	80		1.0

Figure 309. ServletExpress Manager Main Page

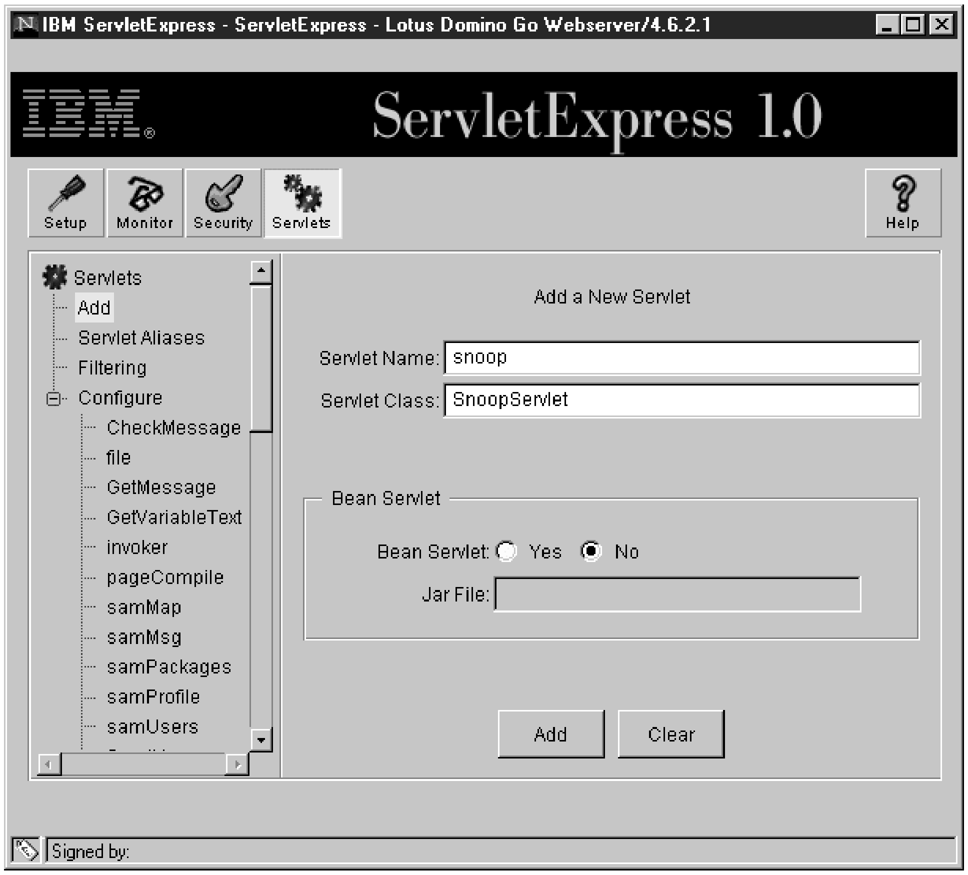
Select **ServletExpress - Lotus Domino Go Webserver/4.6.2.2** (this string is entirely visible only if you widen the Services column) and then click on the **Manage** button. The following window will be displayed:



Click on one of the Setup items.

Figure 310. ServletExpress - Lotus Domino Go Webserver Main Menu

Click on the **Servlets** icon and the Servlets page will be brought up:


IBM ServletExpress - ServletExpress - Lotus Domino Go Webserver/4.6.2.1

ServletExpress 1.0

Add a New Servlet

Servlet Name: snoop

Servlet Class: SnoopServlet

Bean Servlet

Bean Servlet: Yes [✓ No]

Jar File:

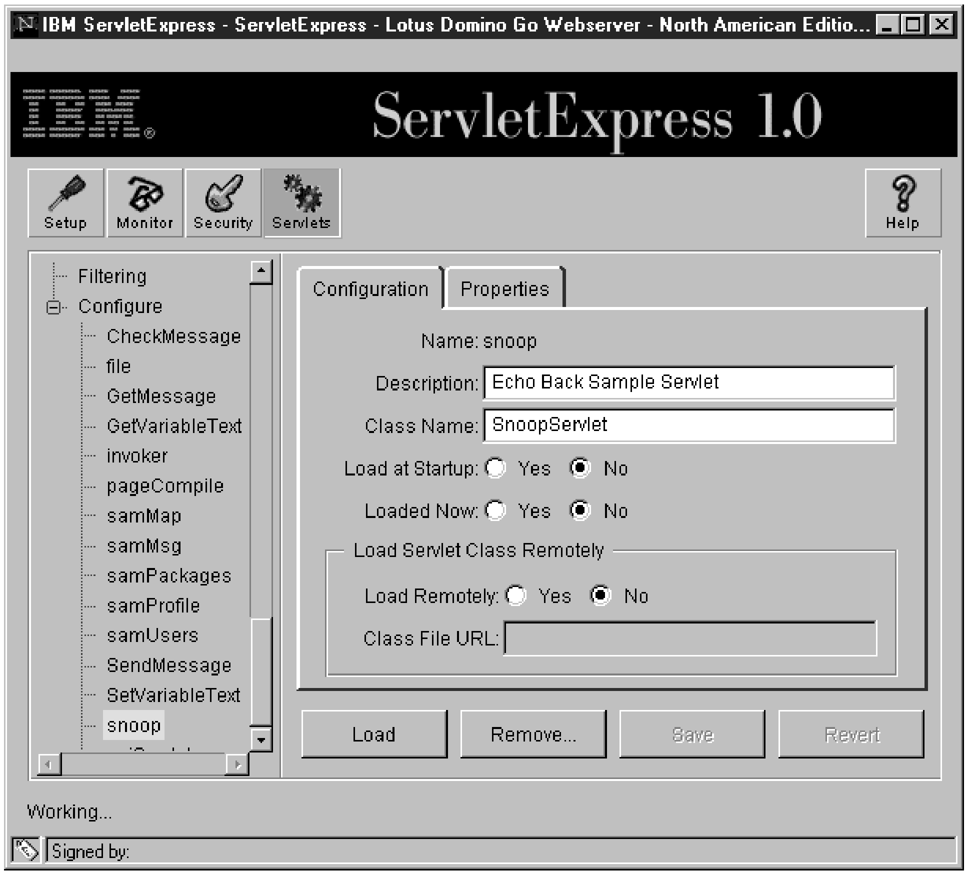
Figure 311. ServletExpress Manager Servlets Page

Select **Add** and enter the Servlet Name and Servlet Class for your servlet into this window:

1. The Servlet Name is the unique name of the servlet you are adding.
2. The Servlet Class is the name of the associated class file for the servlet you are adding, but you don't have to specify the .class extension.

After you have entered these values, press the **Add** button. As an example, we show how to add a servlet named snoop, whose Java class file name is SnoopServlet.class (but the extension .class is not specified in the Servlet Class field). This is a sample servlet that comes with the installation of ServletExpress. You will find both the java and class files for this example already stored in the servlet directory, so you won't need to move any file. If you follow these steps, snoop will be the name used by ServletExpress to identify the SnoopServlet servlet. Moreover you will be able to invoke this servlet by calling it SnoopServlet or simply snoop.

The ServletExpress Manager allows you to specify configuration and properties parameters for this servlet. In the Configuration window you can enter a description of your servlet and you can decide when and how that servlet must be loaded, as you can see in the following screen:


IBM ServletExpress - ServletExpress - Lotus Domino Go Webserver - North American Editio...

ServletExpress 1.0

Name: snoop

Description: Echo Back Sample Servlet

Class Name: SnoopServlet

Load at Startup: Yes [✓ No]

Loaded Now: Yes [✓ No]

Load Servlet Class Remotely

Load Remotely: Yes [✓ No]

Class File URL:

Working...

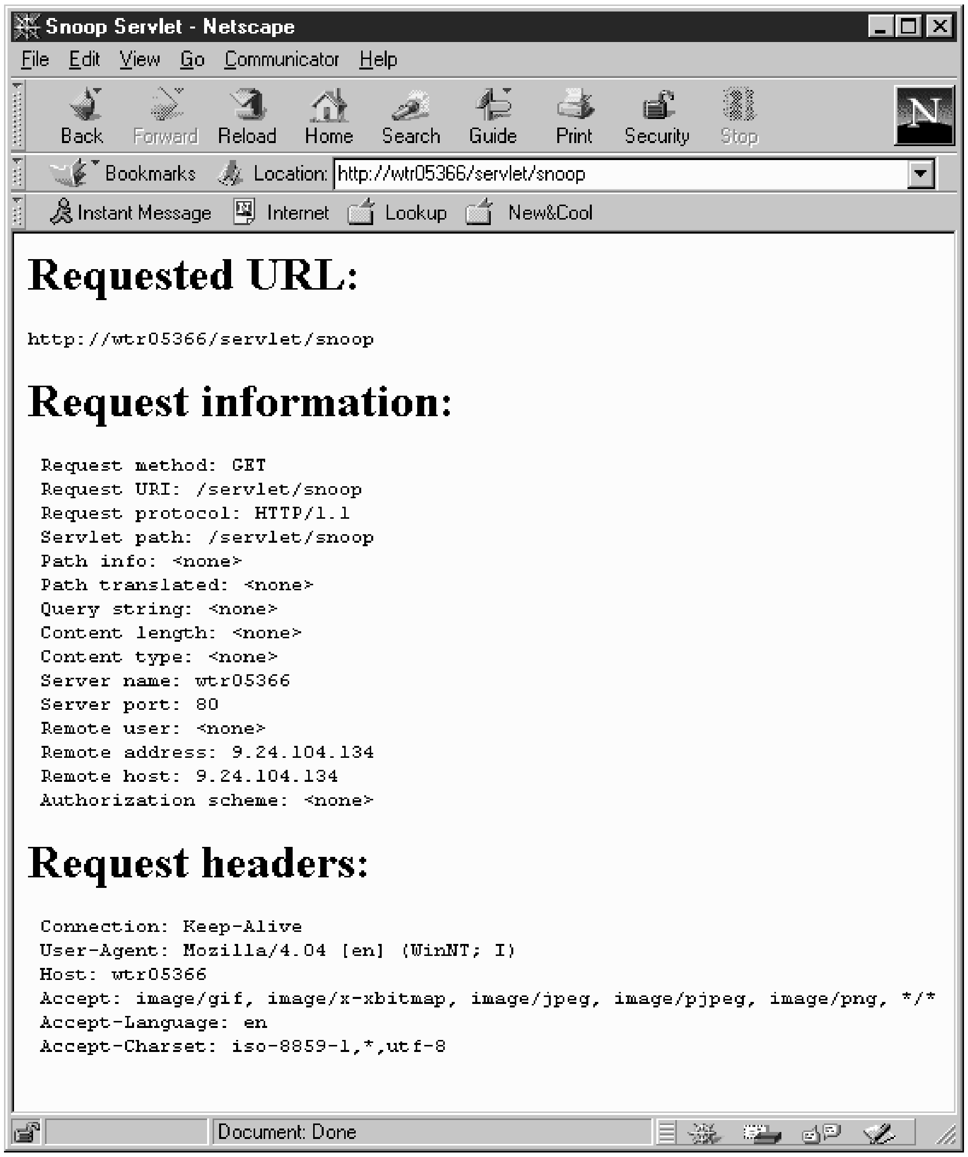
Signed by:

Figure 312. Configuration Window for the SnoopServlet Servlet

Pressing the **Load** button would force the Web server to load the SnoopServlet servlet. Otherwise, that servlet will be loaded as soon as the first user invokes it from his Web browser.

Clicking the **Properties** tag, you can then set the initialization parameters for the servlet, by specifying the name of each parameter and its value. However, the SnoopServlet does not need any initialization parameter, so we did not use the Properties window.

This sequence of operations is exactly what you need to add a servlet to SerlvetExpress. From now on, you will be permitted to invoke the SnoopServlet servlet by pointing your Web browser to the URL http://*yourservername*/servlet/snoop or http://*yourservername*/servlet/SnoopServlet, where *yourservername* is the fully qualified name for your Web server machine. Notice that servlet is the symbolic servlet location that the Web server replaces with the actual servlet directory, while snoop is the name that we gave to the SnoopServlet servlet. The following window demonstrates this result:


Snoop Servlet - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://wtr05366/servlet/snoop
Instant Message Internet Lookup New&Cool
Document: Done

Requested URL:

```
http://wtr05366/servlet/snoop
```

Request information:

```
 Request method: GET
 Request URI: /servlet/snoop
 Request protocol: HTTP/1.1
 Servlet path: /servlet/snoop
 Path info: <none>
 Path translated: <none>
 Query string: <none>
 Content length: <none>
 Content type: <none>
 Server name: wtr05366
 Server port: 80
 Remote user: <none>
 Remote address: 9.24.104.134
 Remote host: 9.24.104.134
 Authorization scheme: <none>
```

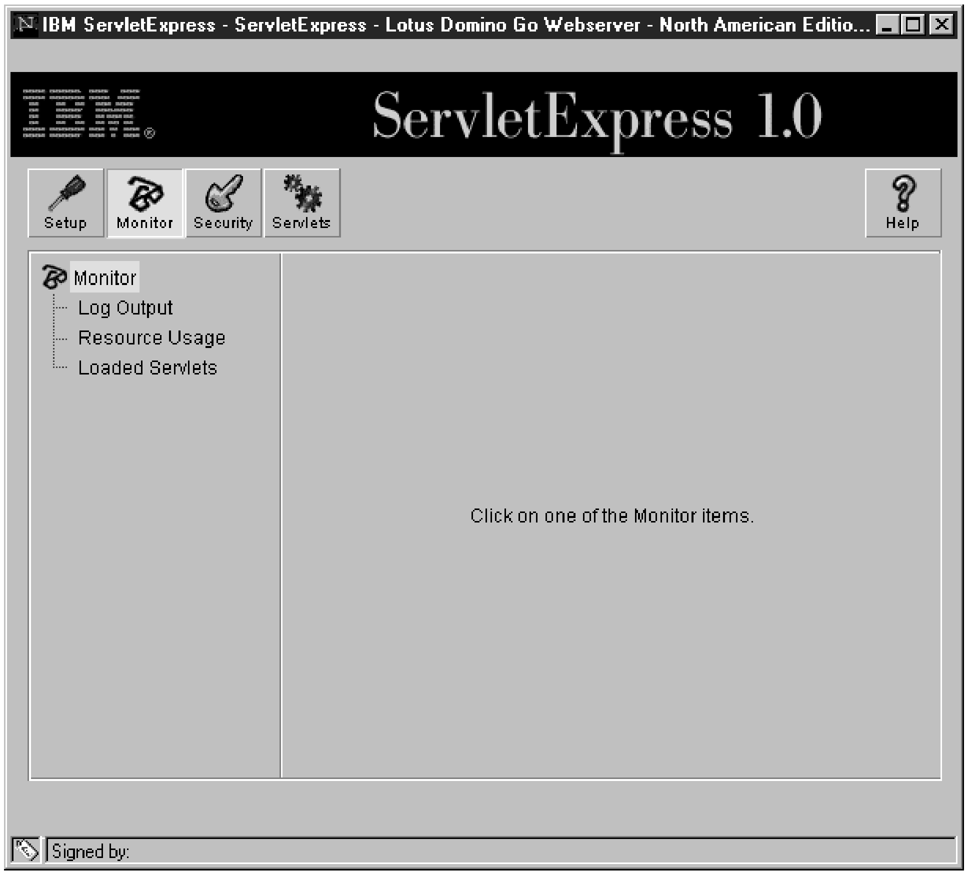
Request headers:

```
 Connection: Keep-Alive
 User-Agent: Mozilla/4.04 [en] (WinNT; I)
 Host: wtr05366
 Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
 Accept-Language: en
 Accept-Charset: iso-8859-1,*,utf-8
```

Figure 313. Invoking the SnoopServlet Servlet

7.3.6 How to Monitor Loaded Servlets with ServletExpress

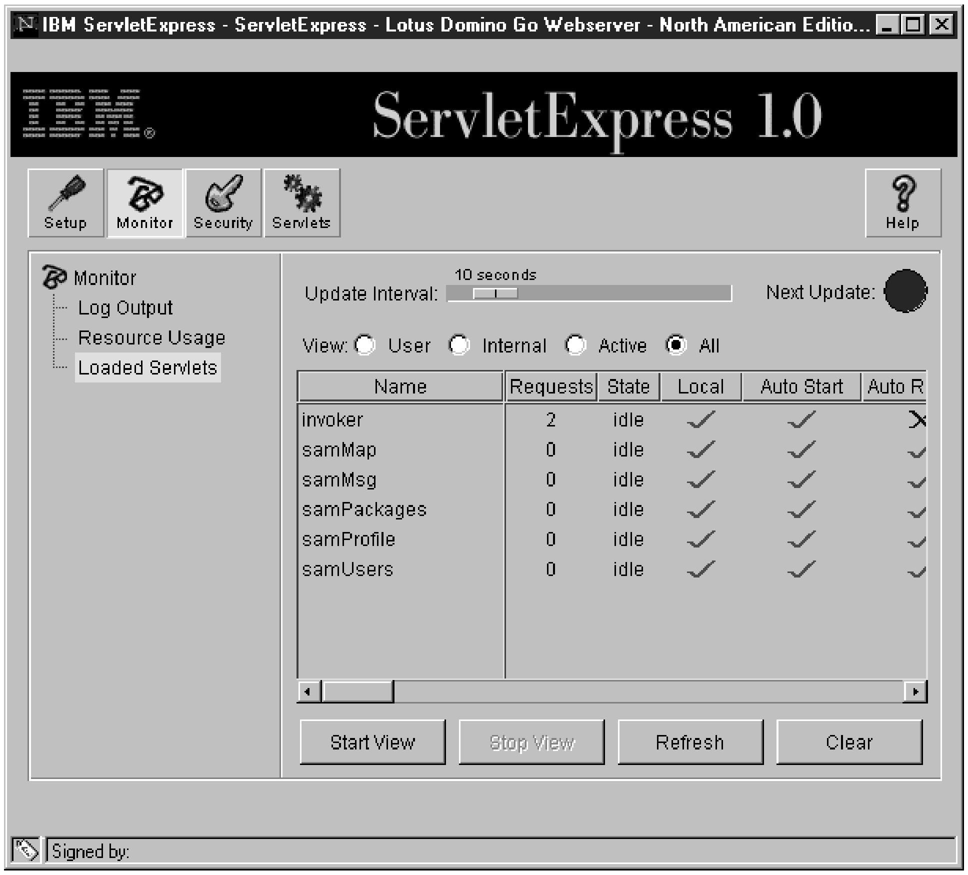
You can monitor loaded servlets by using ServletExpress Manager. All you have to do is press the **Monitor** button for the ServletExpress Manager Configuration window (see Figure 312 on page 331) and you will get the following screen:



Click on one of the Monitor items.

Figure 314. ServletExpress Manager Monitor Window

Then you should select **Loaded Servlets**. Let's suppose that, since you started up your Web server, no user has invoked the SnoopServlet servlet that you added to ServletExpress in 7.3.5, “How to Add Servlets into ServletExpress” on page 327. The ServletExpress Monitor window shows this situation, since SnoopServlet does not appear among the currently loaded servlets, as you can see in the following figure:



Update Interval: 10 seconds

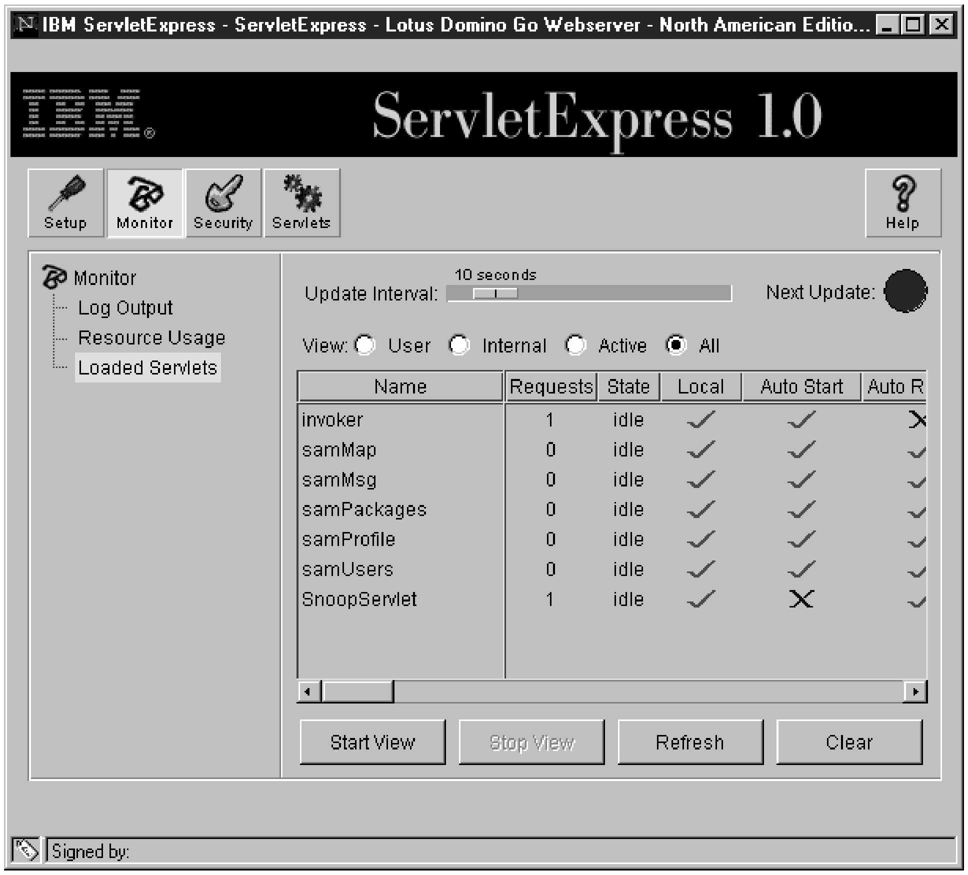
Next Update: ●

View: ( ) User ( ) Internal ( ) Active [✓ All]

Name	Requests	State	Local	Auto Start	Auto R
invoker	2	idle	✓	✓	✗
samMap	0	idle	✓	✓	✓
samMsg	0	idle	✓	✓	✓
samPackages	0	idle	✓	✓	✓
samProfile	0	idle	✓	✓	✓
samUsers	0	idle	✓	✓	✓

Figure 315. Monitoring Loaded Servlets with the ServletExpress Manager

As soon as the Web server loads the SnoopServlet (either by pressing the **Load** button in the Configuration window for the SnoopServlet servlet or by pointing a Web browser to the URL http://*yourservername*/servlet/SnoopServlet), the ServletExpress Manager Monitor window will register it among the loaded servlets, as you can see in the following screen:



Monitor
- Log Output
- Resource Usage
- Loaded Servlets

Update Interval: 10 seconds Next Update:

View: ( ) User ( ) Internal ( ) Active [✓ All]

Name	Requests	State	Local	Auto Start	Auto R
invoker	1	idle	✓	✓	✗
samMap	0	idle	✓	✓	✓
samMsg	0	idle	✓	✓	✓
samPackages	0	idle	✓	✓	✓
samProfile	0	idle	✓	✓	✓
samUsers	0	idle	✓	✓	✓
SnoopServlet	1	idle	✓	✗	✓

Figure 316. The ServletExpress Monitor Window Registers the SnoopServlet Servlet

We could also have invoked the SnoopServlet servlet by using its servlet name snoop. In that case, the ServletExpress Manager Monitor window would register the servlet loaded as snoop.

7.3.7 How to Migrate the Servlet Classes for ServletExpress

If you already have your servlet programs running on the servlet API support function of Lotus Domino Go Webserver, you should first deactivate the servlet support of Lotus Domino Go Webserver before installing ServletExpress (see Figure 293 on page 314). Then you will need to move the Servlet and Eservlet directives stored in the servlet.cnf file to the ServletExpress configuration. Notice that the installation of Lotus Domino Go Webserver does not generate the file servlet.cnf if you do not select **Java Servlet** in the installed component list (see Figure 277 on page 303).

The migration of servlet classes can be accomplished as described in the following list:

1. Print or view the Servlet and Eservlet directives, which are stored in the servlet.cnf file.
2. Connect to port 9090 to open the ServletExpress Manager by using a Java-enabled Web browser with an adequate Java 1.1 support.
3. Log in by entering the user ID and password for the ServletExpress administrator.

4. Each Servlet and Eservlet directive will need to be added, then configured. ServletExpress does not distinguish between internal and external servlets like Lotus Domino Go Webserver does. To see how you can add servlets to ServletExpress, you can refer to 7.3.5, “How to Add Servlets into ServletExpress” on page 327.
5. You will also need to move the servlets from their current location *server_root*\servlets\public (in our platform *server_root* was D:\WWW) or another previously specified directory to *servletexpress_root*\servlets directory (in our case *servletexpress_root* was D:\ServletExpress).
6. Once this is done, restart the Web server. ServletExpress can only be run with one Web server at a time.

However we didn't need to test this procedure, since we installed Lotus Domino Go Webserver without its Java Servlets component (see Figure 277 on page 303) and for this reason, when we installed ServletExpress, it was not necessary to migrate the Java Servlet support.

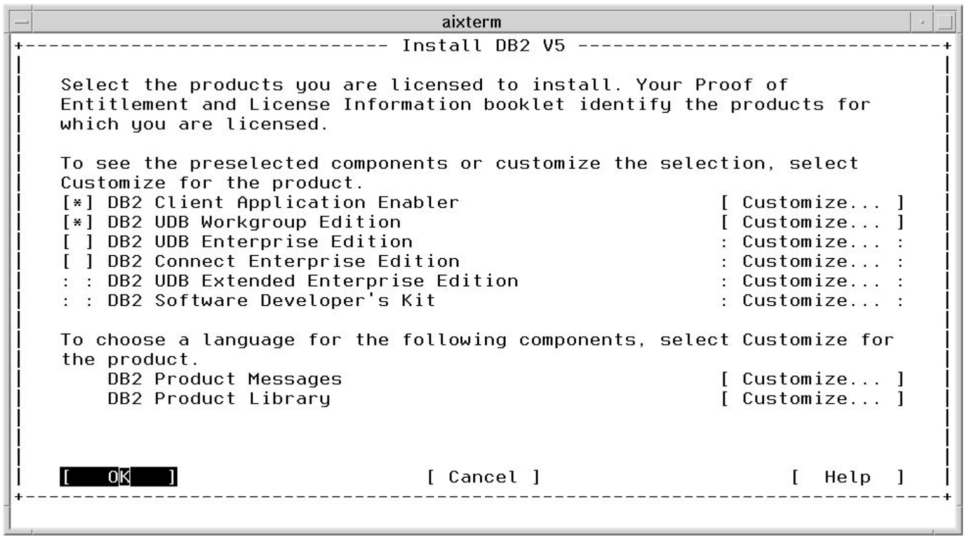
7.4 IBM DB2 Universal Database 5.0 for AIX

IBM DB2 Universal Database (UDB) is a relational database management system. We used IBM DB2 UDB Version 5.0 Workgroup Edition for AIX in the backend server tier of our test environment, where we created the sample database provided with the installation. The machine where we installed the database was an IBM RS/6000 43P and its operating system was IBM AIX 4.3.

The Workgroup Edition for the IBM DB2 UDB provides functions to create and manage databases in LAN servers and it is designed for use by small departments and groups. This is the reason why, for our test environment, we decided to install the Workgroup Edition rather than the Enterprise Edition, which provides also a gateway to access Distributed Relational Database Architecture (DRDA) application servers.

We also installed, in the same machine, IBM DB2 Client Application Enabler (CAE) Version 5.0 for AIX to test the connection to the DB2 UDB server locally.

When we entered the `db2setup` command to start the DB2 Installer, we got the following window:



```
aixterm
+-------------------------------- Install DB2 V5 --------------------------------+
|                                                                                 |
|    Select the products you are licensed to install. Your Proof of              |
|    Entitlement and License Information booklet identify the products for       |
|    which you are licensed.                                                      |
|                                                                                 |
|    To see the preselected components or customize the selection, select        |
|    Customize for the product.                                                   |
|    [*] DB2 Client Application Enabler                    [ Customize... ]      |
|    [*] DB2 UDB Workgroup Edition                         [ Customize... ]      |
|    [ ] DB2 UDB Enterprise Edition                        : Customize... :      |
|    [ ] DB2 Connect Enterprise Edition                    : Customize... :      |
|    : : DB2 UDB Extended Enterprise Edition               : Customize... :      |
|    : : DB2 Software Developer's Kit                      : Customize... :      |
|                                                                                 |
|    To choose a language for the following components, select Customize for    |
|    the product.                                                                 |
|        DB2 Product Messages                              [ Customize... ]      |
|        DB2 Product Library                               [ Customize... ]      |
|                                                                                 |
|                                                                                 |
|                                                                                 |
|    [   OK   ]                [ Cancel ]                  [  Help  ]           |
+---------------------------------------------------------------------------------+
```

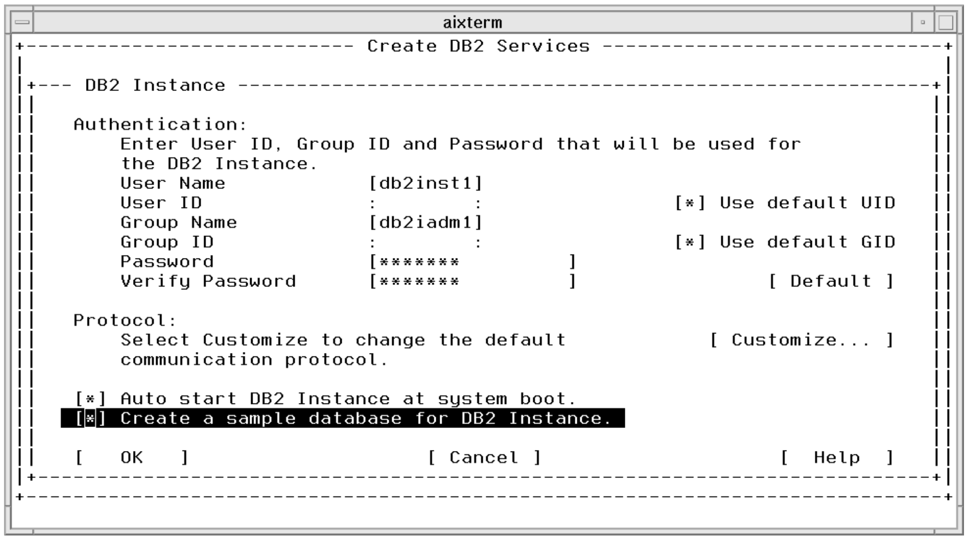
Figure 317. DB2 Component Selection

We selected the following items:

- DB2 Client Application Enabler
- DB2 UDB Workgroup Edition

Then we pressed **OK**.

We defined the information to create a DB2 instance in the following window:



```
aixterm
+---------------------------- Create DB2 Services -----------------------------+
|                                                                               |
|+--- DB2 Instance -----------------------------------------------------------+|
||                                                                            ||
||   Authentication:                                                          ||
||       Enter User ID, Group ID and Password that will be used for           ||
||       the DB2 Instance.                                                    ||
||       User Name            [db2inst1]                                      ||
||       User ID              :        :              [*] Use default UID     ||
||       Group Name           [db2iadm1]                                      ||
||       Group ID             :        :              [*] Use default GID     ||
||       Password             [*******        ]                               ||
||       Verify Password      [*******        ]               [ Default ]     ||
||                                                                            ||
||   Protocol:                                                                ||
||       Select Customize to change the default            [ Customize... ]   ||
||       communication protocol.                                              ||
||                                                                            ||
||   [*] Auto start DB2 Instance at system boot.                              ||
||   [*] Create a sample database for DB2 Instance.                           ||
||                                                                            ||
||   [   OK   ]               [ Cancel ]                    [  Help  ]        ||
|+----------------------------------------------------------------------------+|
+-------------------------------------------------------------------------------+
```

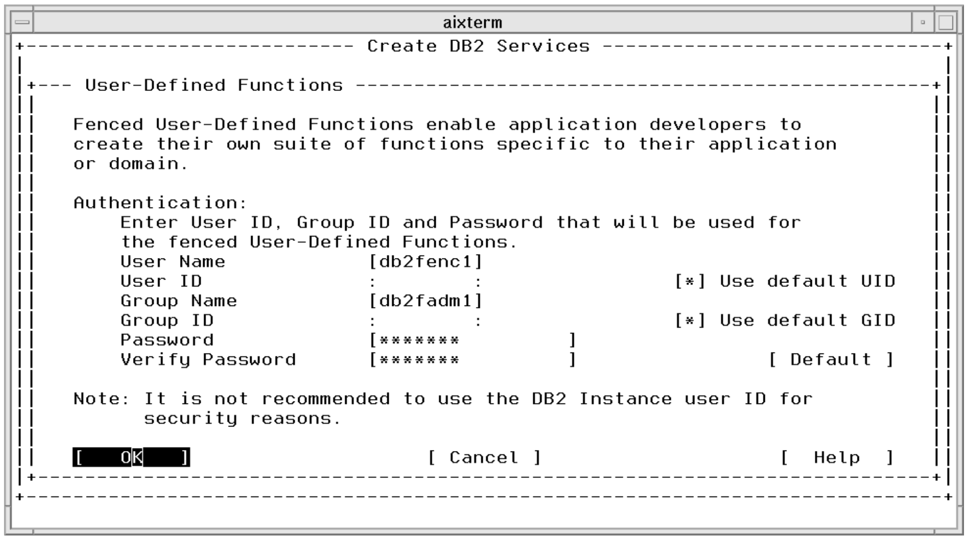
Figure 318. Creation of the DB2 Instance and of the Sample Database

An important decision that we had to make was the choice of a password for the DB2 instance we were going to create. By default, the password for the db2inst1 instance is set to ibmdb2. You should change it as soon as possible; otherwise, you would have a security hole in your environment.

Figure 318 shows that we selected also the option **Create a sample database for DB2 Instance**. The sample database provided by IBM DB2 UDB Version 5.0 requires approximately 9 MB of disk space. You should increase your disk space before installing the sample database, if it it less than 9 MB. The sample database is by default named SAMPLE.

Application developers often need to create their own suite of functions specific to their application or domain. User-Defined Functions (UDFs) make this possible, expanding the scope of DB2 to include, for example, customized business or scientific functions. UDFs can operate over all database types. Fenced UDFs are particular UDFs that ensure data integrity.

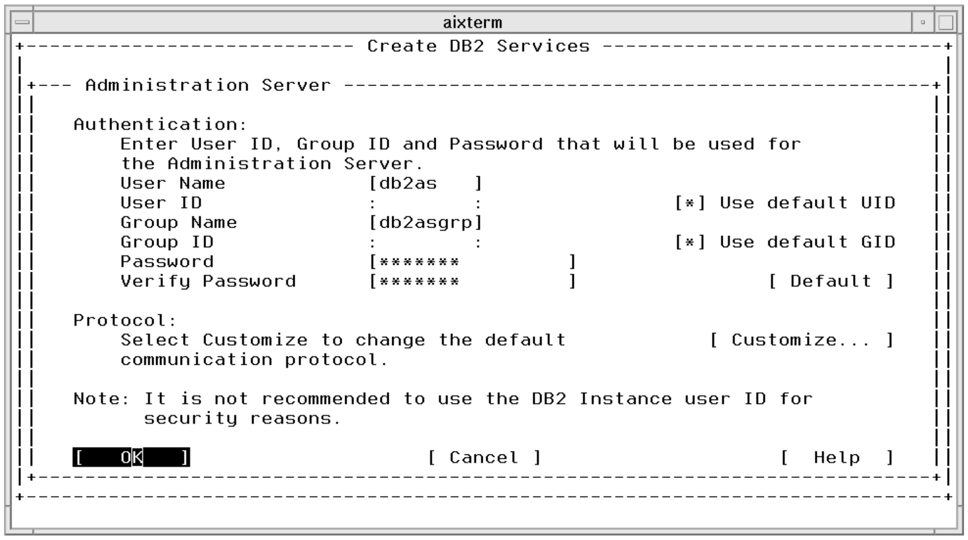
We had to define Authentication information to use fenced UDFs. For this reason, we created a user and a group for fenced UDFs to execute under. We did this in the following window:



```
aixterm
+---------------------------- Create DB2 Services -----------------------------+
|                                                                              |
|+--- User-Defined Functions -------------------------------------------------+|
||                                                                            ||
||   Fenced User-Defined Functions enable application developers to           ||
||   create their own suite of functions specific to their application        ||
||   or domain.                                                               ||
||                                                                            ||
||   Authentication:                                                          ||
||       Enter User ID, Group ID and Password that will be used for           ||
||       the fenced User-Defined Functions.                                   ||
||       User Name            [db2fenc1]                                      ||
||       User ID              :        :                [*] Use default UID   ||
||       Group Name           [db2fadm1]                                      ||
||       Group ID             :        :                [*] Use default GID   ||
||       Password             [*******         ]                              ||
||       Verify Password      [*******         ]                [ Default ]   ||
||                                                                            ||
||   Note: It is not recommended to use the DB2 Instance user ID for          ||
||         security reasons.                                                  ||
||                                                                            ||
||   [     OK     ]                [ Cancel ]                    [  Help  ]   ||
|+----------------------------------------------------------------------------+|
+------------------------------------------------------------------------------+
```

Figure 319. User-Defined Functions Authentication Window

Once again, as you can see, the choice of the password is very important. You must pay attention to the password also when you enter User Name and Group Name for the Administration Server in the following window:



```
aixterm
+---------------------------- Create DB2 Services -----------------------------+
|                                                                              |
|+--- Administration Server --------------------------------------------------+|
||                                                                            ||
||   Authentication:                                                          ||
||       Enter User ID, Group ID and Password that will be used for           ||
||       the Administration Server.                                           ||
||       User Name            [db2as   ]                                      ||
||       User ID              :        :                [*] Use default UID   ||
||       Group Name           [db2asgrp]                                      ||
||       Group ID             :        :                [*] Use default GID   ||
||       Password             [*******         ]                              ||
||       Verify Password      [*******         ]                [ Default ]   ||
||                                                                            ||
||   Protocol:                                                                ||
||       Select Customize to change the default            [ Customize... ]   ||
||       communication protocol.                                              ||
||                                                                            ||
||   Note: It is not recommended to use the DB2 Instance user ID for          ||
||         security reasons.                                                  ||
||                                                                            ||
||   [     OK     ]                [ Cancel ]                    [  Help  ]   ||
|+----------------------------------------------------------------------------+|
+------------------------------------------------------------------------------+
```

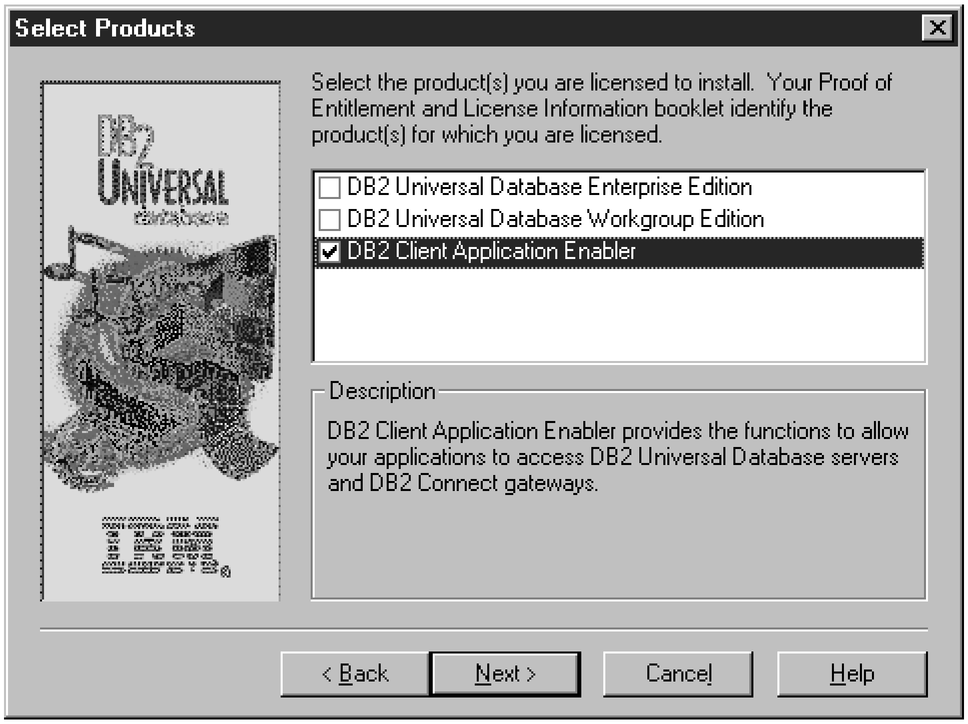
Figure 320. Administration Server Authentication Window

7.5 IBM DB2 Client Application Enabler 5.0 for Windows NT

The IBM DB2 CAE provides a run-time environment that enables client applications to access remote databases. We used the DB2 CAE as a connector. We installed it in the middle tier machine of our test environment, the same NT 4.0 machine where we installed also the JDK, Lotus Domino Go Webserver and ServletExpress. The CAE module had a key role in our environment, because it interfaced to JDBC and DB2 local protocols and transferred data between the Web server and the DB2 server application.

7.5.1 Installation

When we ran the installation program in our middle tier Windows NT Server machine, we had to select the DB2 products we wanted to install, as shown in the following figure:


Select Products

Select the product(s) you are licensed to install. Your Proof of Entitlement and License Information booklet identify the product(s) for which you are licensed.

- [ ] DB2 Universal Database Enterprise Edition
- [ ] DB2 Universal Database Workgroup Edition
- [✓] DB2 Client Application Enabler

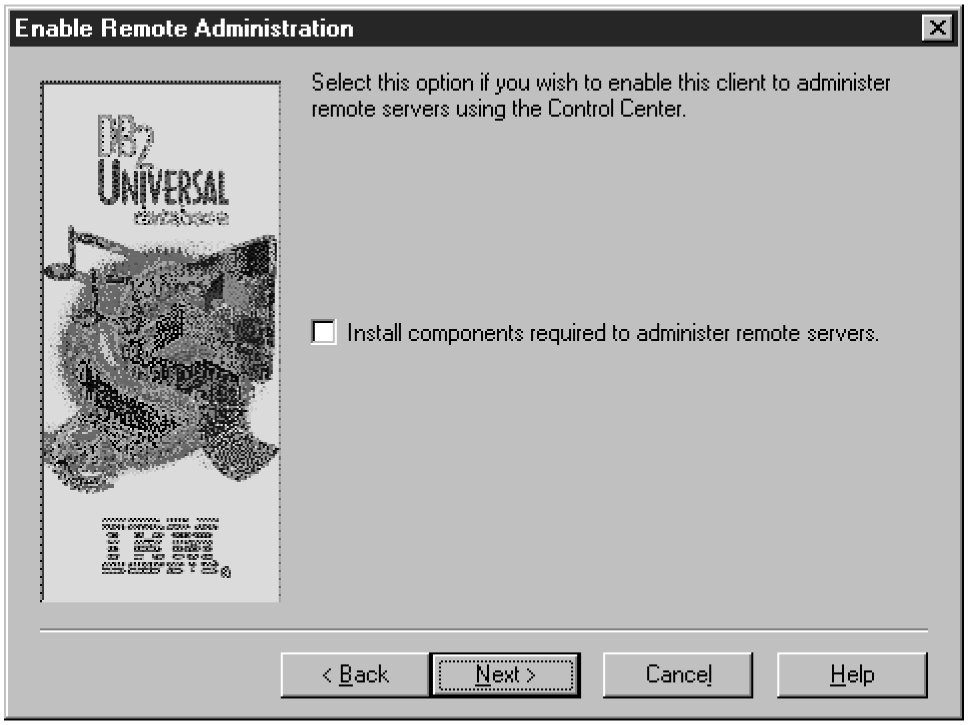
Description

DB2 Client Application Enabler provides the functions to allow your applications to access DB2 Universal Database servers and DB2 Connect gateways.

Figure 321. DB2 Installed Component List

We only selected **DB2 Client Application Enabler**, because the middle tier machine only needs this application the CAE to connect with the DB2 UDB server.

We were then prompted with the option to install the components required to administer remote servers. We did not select this option, since we did not need to administer any remote DB2 server through our middle tier machine.

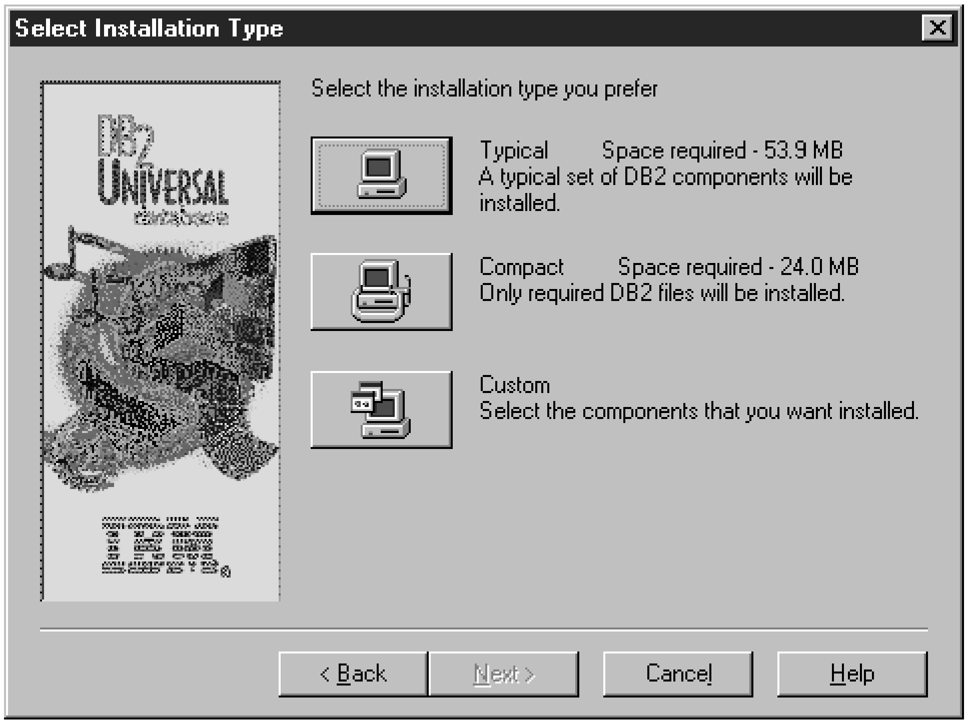

Enable Remote Administration

Select this option if you wish to enable this client to administer remote servers using the Control Center.

☐ Install components required to administer remote servers.

Figure 322. DB2 Remote Administration

When we had to decide what type of installation we wanted to do, we selected **Typical**, since we just needed to install the default set of DB2 components.


Select Installation Type

Select the installation type you prefer

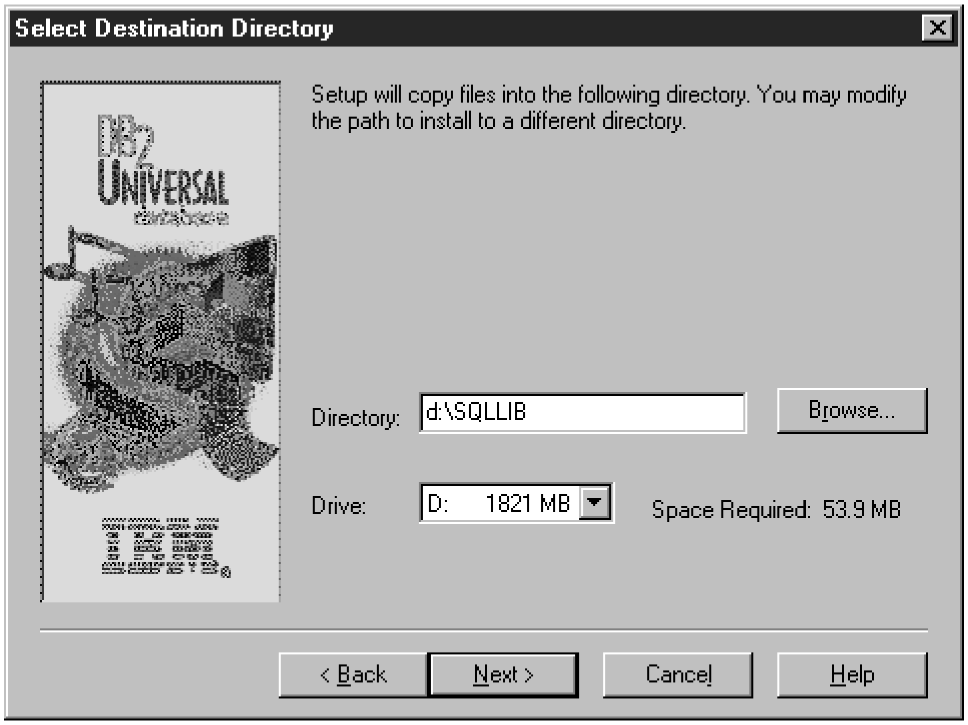
Typical Space required - 53.9 MB
A typical set of DB2 components will be installed.

Compact Space required - 24.0 MB
Only required DB2 files will be installed.

Custom
Select the components that you want installed.

Figure 323. DB2 Installation Type

All the DB2 installation files are usually installed in a directory named SQLLIB, that the DB2 installer program automatically creates in the drive that the user specifies. We selected the D drive and accepted the creation of the SQLLIB directory.


Select Destination Directory

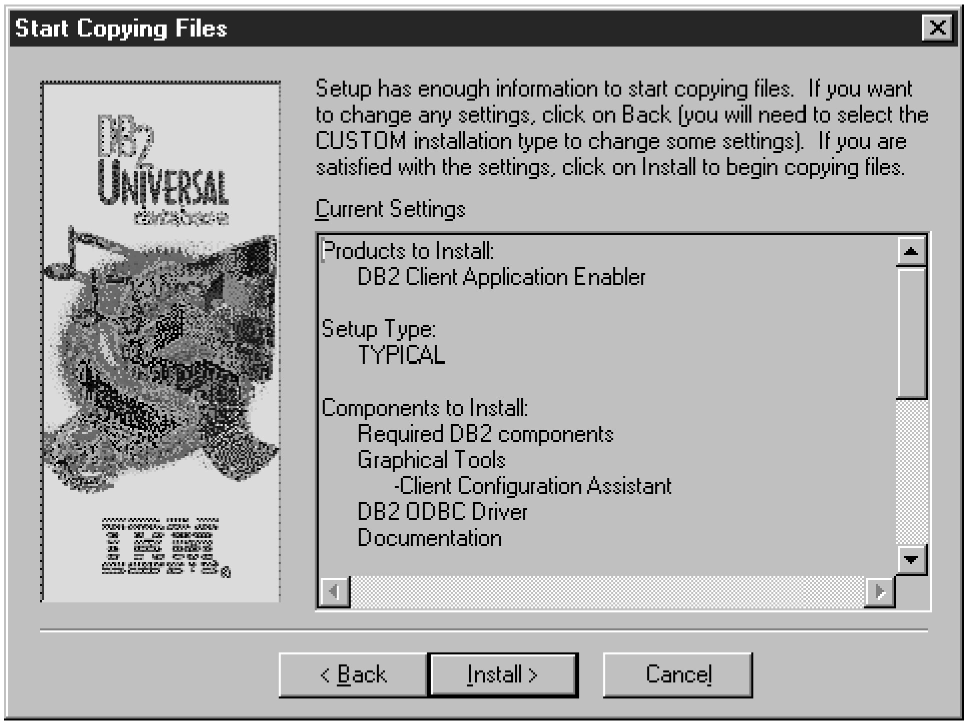
Setup will copy files into the following directory. You may modify the path to install to a different directory.

Directory: d:\SQLLIB

Drive: D: 1821 MB

Space Required: 53.9 MB

Figure 324. Selecting the DB2 Installation Directory


Start Copying Files

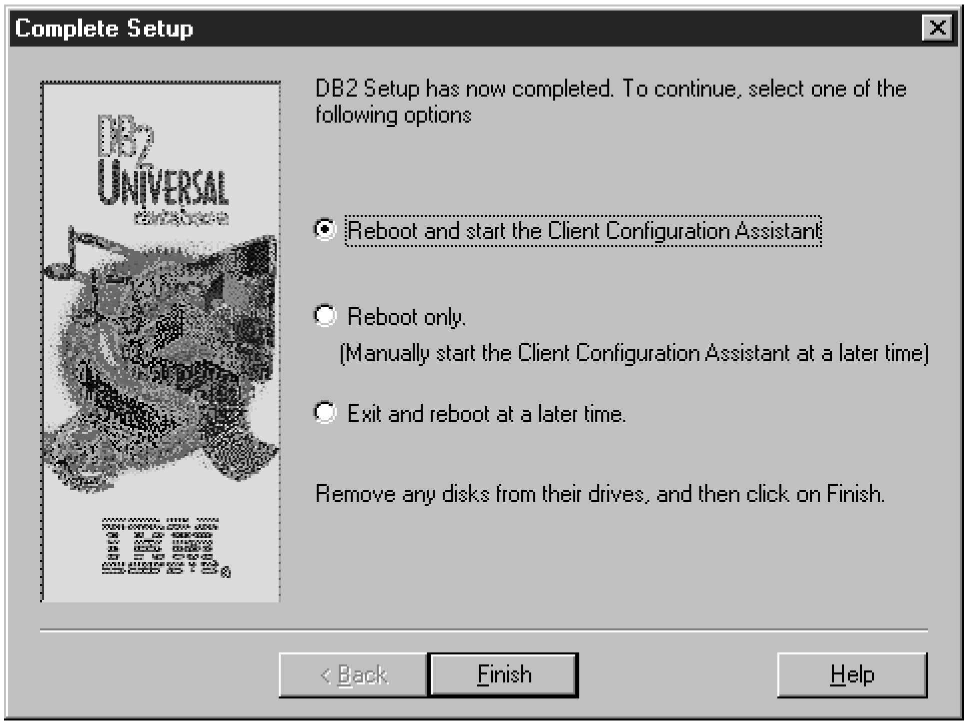
Setup has enough information to start copying files. If you want to change any settings, click on Back (you will need to select the CUSTOM installation type to change some settings). If you are satisfied with the settings, click on Install to begin copying files.

Current Settings

```
Products to Install:
     DB2 Client Application Enabler

Setup Type:
     TYPICAL

Components to Install:
     Required DB2 components
     Graphical Tools
          -Client Configuration Assistant
     DB2 ODBC Driver
     Documentation
```

Figure 325. DB2 CAE Confirmation Window

At the end of the DB2 CAE installation, it was enough to restart our machine. The next step will be to start the Client Configuration Assistant (CCA) to configure the connection between the DB2 client and server (see 7.5.2, “DB2 CAE Connection Configuration” on page 342). Setup offers you also the ability to reboot and automatically start the Client Configuration Assistant, which is what we did.



DB2 Setup has now completed. To continue, select one of the following options

[✓ Reboot and start the Client Configuration Assistant]

Reboot only.
(Manually start the Client Configuration Assistant at a later time)

Exit and reboot at a later time.

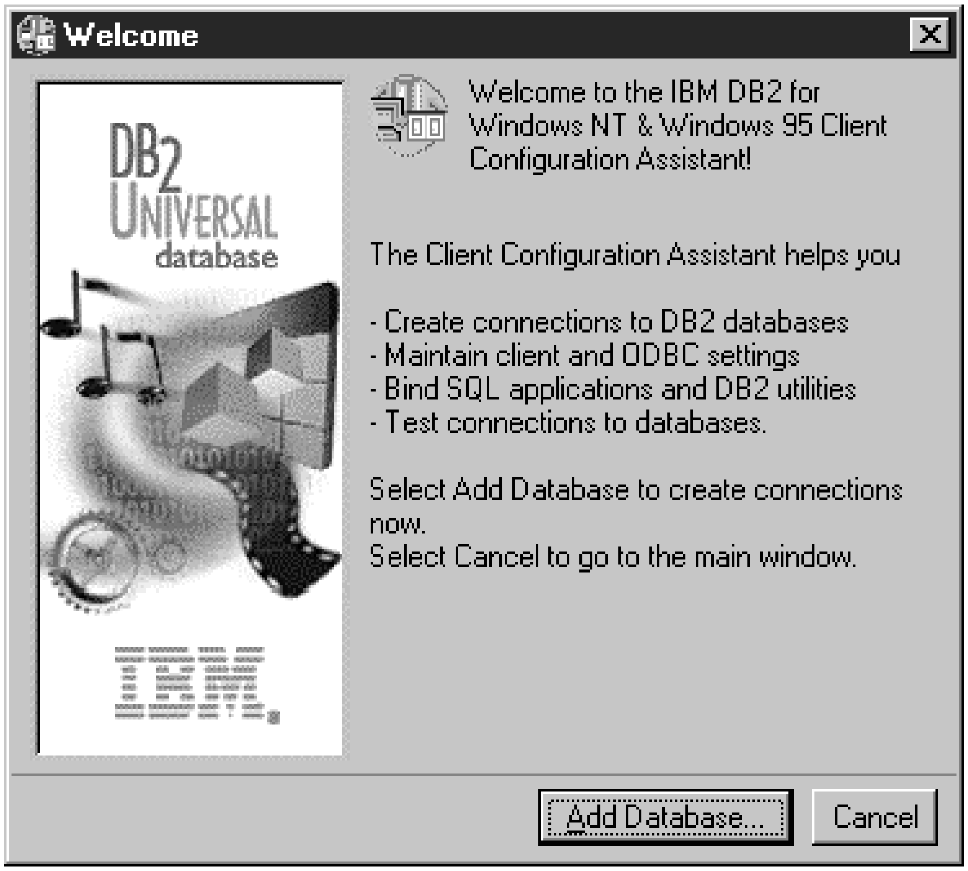
Remove any disks from their drives, and then click on Finish.

Figure 326. DB2 CAE Completion

7.5.2 DB2 CAE Connection Configuration

In order to run a NCF three-tier scenario involving the DB2 sample database installed on the application server machine, it was necessary to connect the DB2 CAE in the Web server NT machine with the remote database that we had created in the application server AIX machine. This operation was very simple, since we used the Client Configuration Assistant (CCA) provided by DB2 CAE for Windows NT.

If you did not select the option to reboot your system and automatically start the CCA, you will need to launch the CCA manually. This operation, on Windows NT, can be accomplished by clicking on **Start** on the Taskbar. Then you should select **Programs** and **DB2 for Windows NT**. At that point click on **Client Configuration Assistant** and the following Welcome window will appear:



Welcome to the IBM DB2 for Windows NT & Windows 95 Client Configuration Assistant!

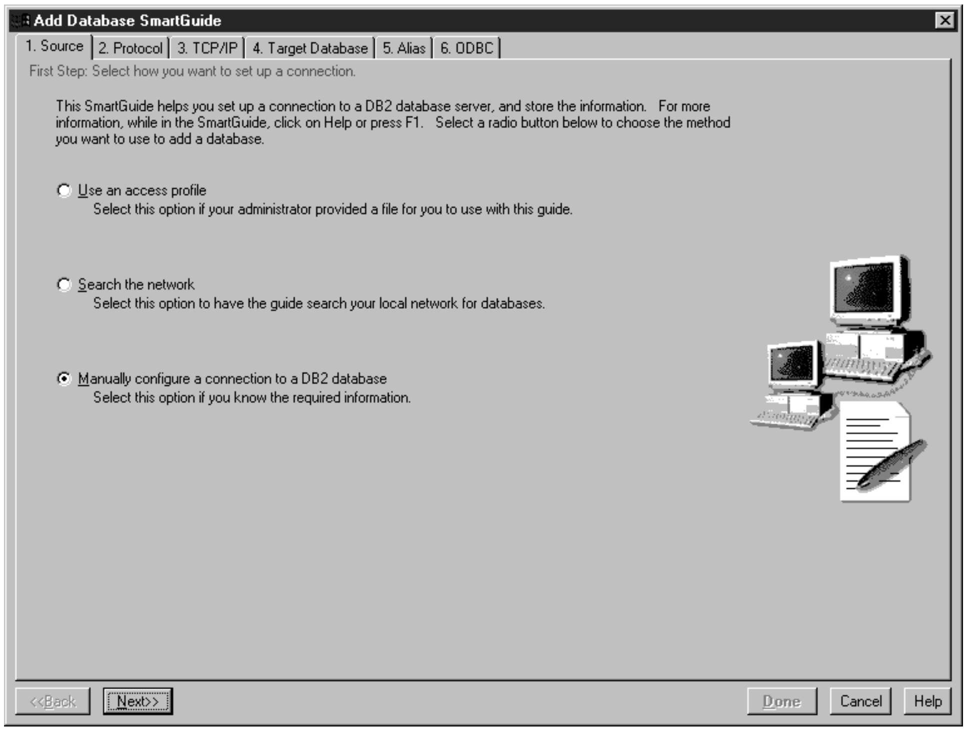
The Client Configuration Assistant helps you

- Create connections to DB2 databases
- Maintain client and ODBC settings
- Bind SQL applications and DB2 utilities
- Test connections to databases.

Select Add Database to create connections now.
Select Cancel to go to the main window.

Figure 327. DB2 Client Configuration Assistant Welcome Window

We pressed the **Add Database...** button and the Add Database SmartGuide window was automatically opened:



First Step: Select how you want to set up a connection.

This SmartGuide helps you set up a connection to a DB2 database server, and store the information. For more information, while in the SmartGuide, click on Help or press F1. Select a radio button below to choose the method you want to use to add a database.

○ Use an access profile
Select this option if your administrator provided a file for you to use with this guide.

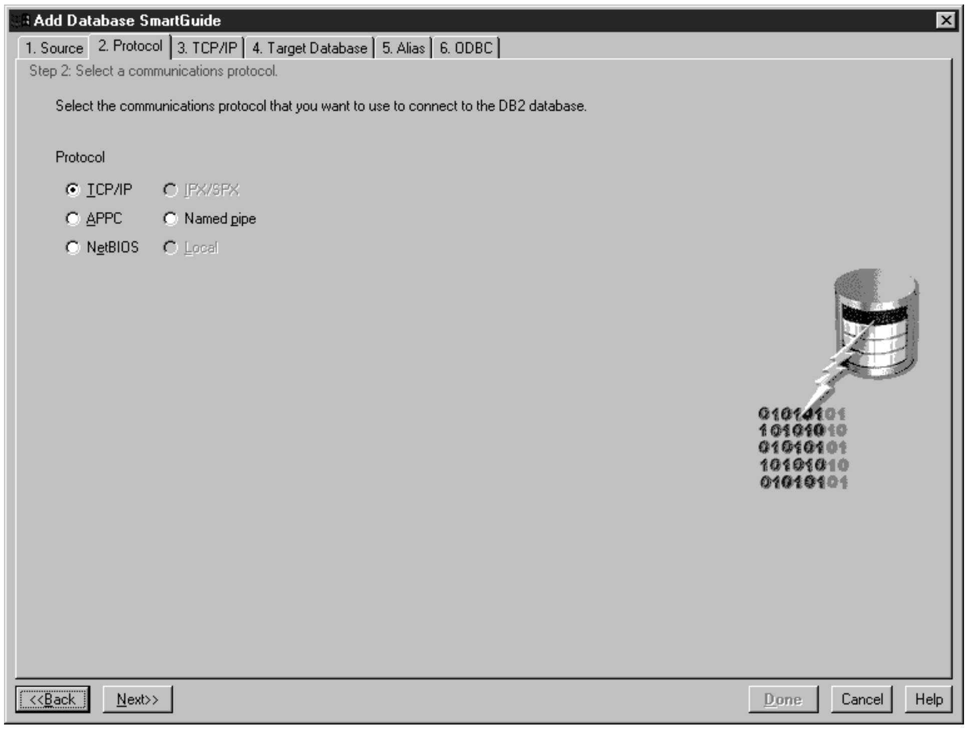
○ Search the network
Select this option to have the guide search your local network for databases.

● Manually configure a connection to a DB2 database
Select this option if you know the required information.

Figure 328. Add Database SmartGuide Window

First of all, we had to select the source database. The Source tab on the top of the window was selected. We marked the radio button named **Manually Configure a Connection to a DB2 Database** and then we pressed the **Next** button.

The Protocol page for the Add Database SmartGuide window was automatically selected:



Step 2: Select a communications protocol.

Select the communications protocol that you want to use to connect to the DB2 database.

Protocol

[✓ TCP/IP] IPX/SPX
APPC Named pipe
NetBIOS Local

Figure 329. Protocol Page for the Add Database SmartGuide Window

This page had options to select the network protocol which the DB2 CAE would use to communicate with the remote database. Of course we marked **TCP/IP** and then we pressed **Next** again.

Once the TCP/IP page was displayed (see Figure 330 on page 346), we had to fill out some fields related to the TCP/IP configuration. The field named Hostname could be filled with the IP address, and we entered the IP address of the DB2 server machine. There was also a field named Port number, and we had to fill in that field with the port number associated with the DB2 instance containing the target database in the DB2 server machine. This is the port where the DB2 server will be listening in order to accept connection requests from a DB2 client using TCP/IP. For this reason this port is named main connection port. Its default value is 50000. A discussion on how port numbers are assigned is presented in 6.1.2, "Expert IP Filters Using IBM eNetwork Firewall 3.2" on page 271 and in 8.5, "IP Filters Configuration for Three-Tier Applications" on page 390.

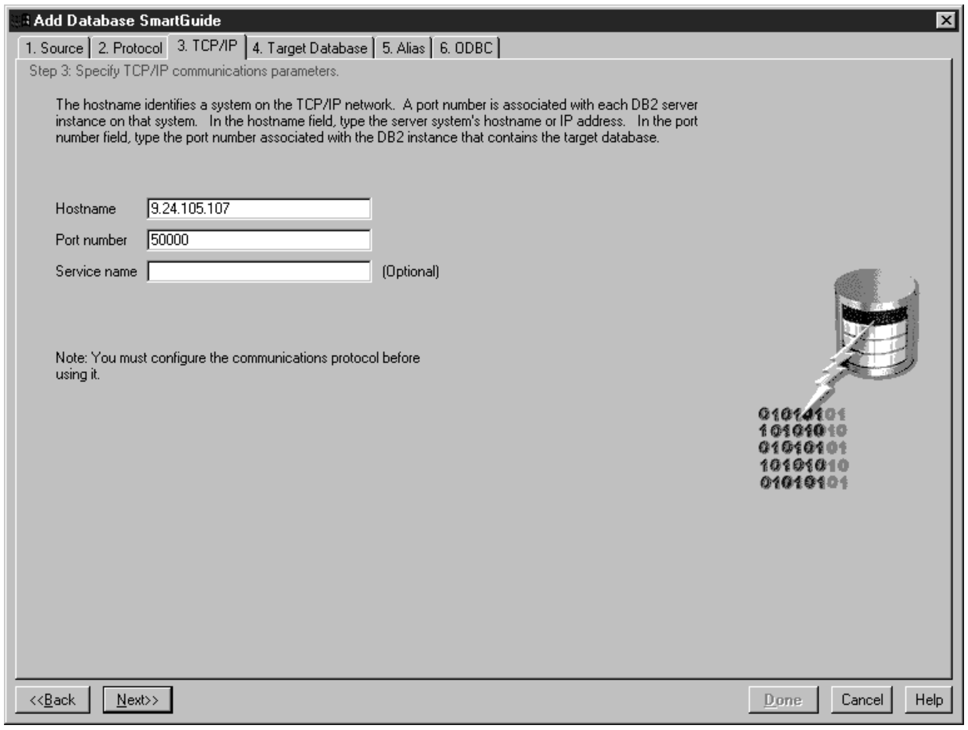
Actually there is also another port named interrupt port. The DB2 server uses the interrupt port to handle interrupt requests from DB2 clients. The interrupt port number is one number greater than the main connection port number, so by default it is 50001.

It is possible to customize the value for the main connection port since it is one of the parameters in the DB2 Manager Configuration File.

DB2 Manager Configuration File

The DB2 Manager Configuration file is named db2systm. On AIX, it is stored in the sqllib subdirectory for the instance of the database manager. On NT, it is in the directory SQLLIB\DB2. The main connection port is configured through a variable named svcename, found in the DB2 Manager Configuration file. If you want to modify the value for the svcename variable or for another variable handled by the DB2 Manager Configuration file, you cannot directly edit the db2systm file, or you will corrupt it, making your system unusable. You should instead use the DB2 Control Center or the DB2 Command Line Processor.

We filled the Port number field with the default value assigned to the main connection port and so we typed `50000`. We didn't fill the Service name field, which was optional. The following figure summarizes all our selections:



Step 3: Specify TCP/IP communications parameters.

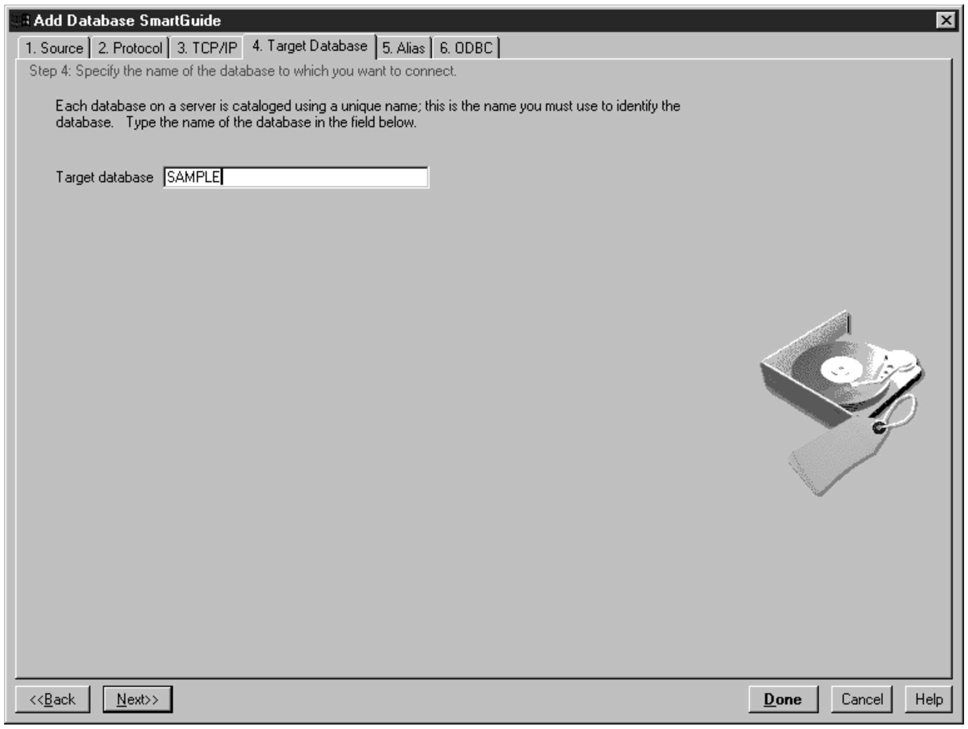
The hostname identifies a system on the TCP/IP network. A port number is associated with each DB2 server instance on that system. In the hostname field, type the server system's hostname or IP address. In the port number field, type the port number associated with the DB2 instance that contains the target database.

Hostname: 9.24.105.107
Port number: 50000
Service name: (Optional)

Note: You must configure the communications protocol before using it.

Figure 330. TCP/IP Configuration

We pressed the **Next** button and saw the Target Database page:


Add Database SmartGuide

1. Source | 2. Protocol | 3. TCP/IP | 4. Target Database | 5. Alias | 6. ODBC

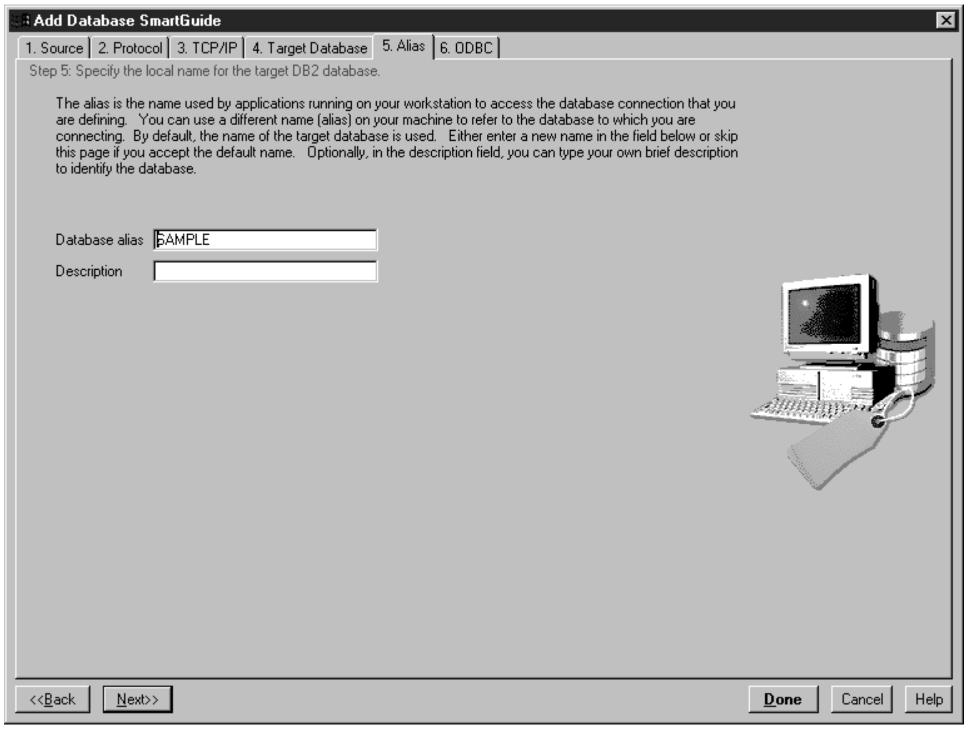
Step 4: Specify the name of the database to which you want to connect.

Each database on a server is cataloged using a unique name; this is the name you must use to identify the database. Type the name of the database in the field below.

Target database: SAMPLE

Figure 331. Selecting the Target Database

We typed SAMPLE in the Target database field, since this was the name that had been automatically given to the sample database created in the DB2 server machine (see Figure 318 on page 337). Then we clicked the **Next** button, opening the Alias page for the Add Database SmartGuide window:


Add Database SmartGuide

1. Source | 2. Protocol | 3. TCP/IP | 4. Target Database | 5. Alias | 6. ODBC

Step 5: Specify the local name for the target DB2 database.

The alias is the name used by applications running on your workstation to access the database connection that you are defining. You can use a different name (alias) on your machine to refer to the database to which you are connecting. By default, the name of the target database is used. Either enter a new name in the field below or skip this page if you accept the default name. Optionally, in the description field, you can type your own brief description to identify the database.

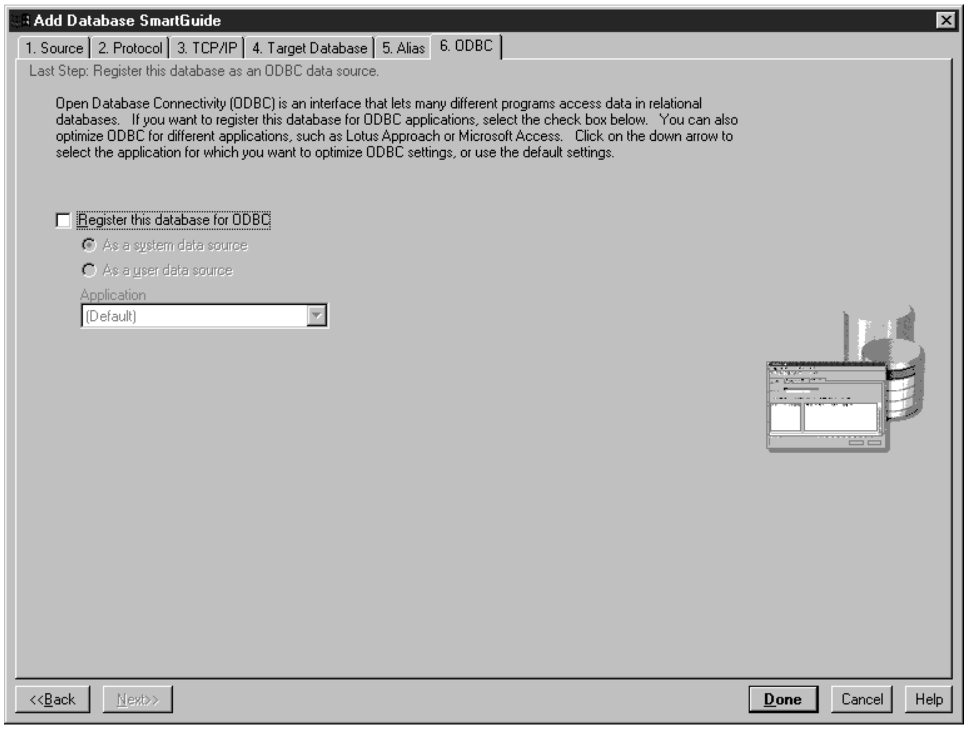
Database alias: SAMPLE

Description:

Figure 332. Database Alias

This page contained a field named Database alias. You can use that field to assign a different name to the remote database with which you connect. The new name will be used only locally by all the applications running on the workstation where the DB2 CAE is installed. For example, a servlet running on your Web server machine will use the database alias to connect to a remote database through the CAE. We accepted the value SAMPLE that the CCA assigned by default, as it was also the name of the remote database. Since the Description field was optional, we didn't fill that field and we pressed the **Next** button again.

The last page of the Add Database SmartGuide window is used to register the database you are configuring for Open Database Connectivity (ODBC) applications:



Last Step: Register this database as an ODBC data source.

Open Database Connectivity (ODBC) is an interface that lets many different programs access data in relational databases. If you want to register this database for ODBC applications, select the check box below. You can also optimize ODBC for different applications, such as Lotus Approach or Microsoft Access. Click on the down arrow to select the application for which you want to optimize ODBC settings, or use the default settings.

☐ Register this database for ODBC

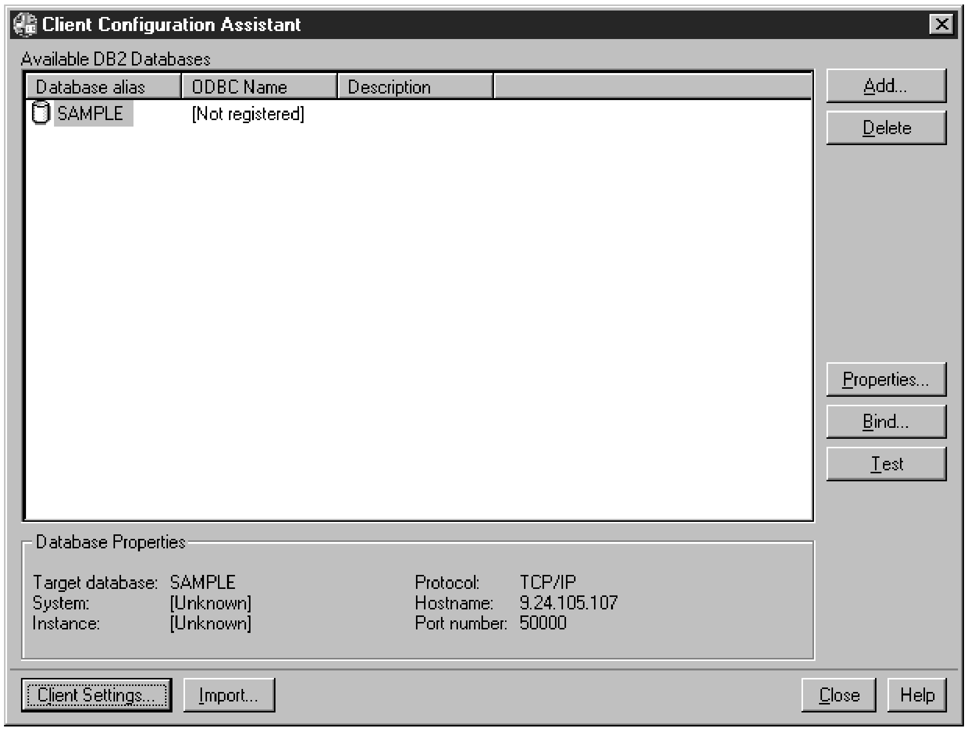
[● As a system data source]

○ As a user data source

Application: (Default)

Figure 333. ODBC Registration

We didn't select the option to register our SAMPLE database for ODBC applications, since we intended to use it with JDBC applications. Once we pressed **Done**, we got the confirmation that the connection configuration for the SAMPLE database had been added successfully and then we saw the following window:



Database alias	ODBC Name	Description
SAMPLE	[Not registered]	

Database Properties

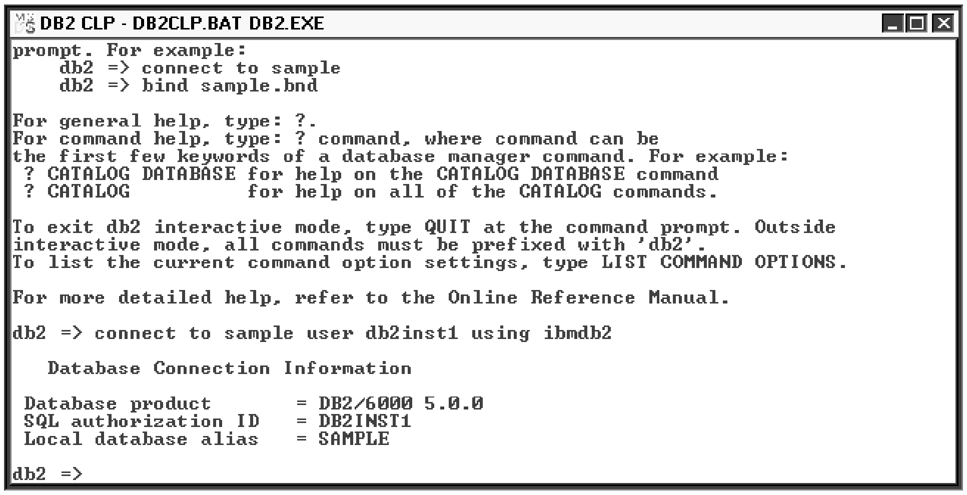
```
Target database:  SAMPLE          Protocol:     TCP/IP
System:           [Unknown]       Hostname:     9.24.105.107
Instance:         [Unknown]       Port number:  50000
```

Figure 334. Available DB2 Databases

We could then test the connection by using the Command Line Processor (CLP) for DB2 CAE. This tool can be launched by selecting **Command Line Processor** from the DB2 for Windows NT menu. As soon as the tool was available, we entered the command:

```
connect to sample user db2inst1 using ibmdb2
```

where db2inst1 was the DB2 instance name and ibmdb2 was the default password in the DB2 UDB server (see Figure 318 on page 337). The result was successful, as you can see in the following window:



```
prompt. For example:
    db2 => connect to sample
    db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
 ? CATALOG DATABASE for help on the CATALOG DATABASE command
 ? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to sample user db2inst1 using ibmdb2

   Database Connection Information

 Database product       = DB2/6000 5.0.0
 SQL authorization ID   = DB2INST1
 Local database alias   = SAMPLE

db2 =>
```

Figure 335. Testing the Connection with the Command Line Processor

7.6 IBM Firewall 3.1.1 and IBM eNetwork Firewall 3.2 for AIX

Until now we have shown how to install all the software products on the machines that would be part of our scenarios. In order to grant security, a firewall machine was integrated in our scenarios. This section describes now how to install and configure the IBM Firewall. We installed this product on an IBM RS/6000 370 machine running AIX as operating system.

IBM Firewall 3.1.1. and 3.2

When the project began, Version 3.1.1 was available for the IBM Firewall. We installed it on AIX 4.2.0 and made several experiments on that platform. These experiments are documented in the book.

During the project, Version 3.2 for the IBM eNetwork Firewall became available, so we uninstalled the old version, built the operating system AIX 4.2.1 necessary to run the new version of the firewall and installed IBM eNetwork Firewall 3.2. We made several experiments using this platform too, and they are documented in the book as well.

With both versions of this product, our experiments were successful. However, we found it easier to use the new Version 3.2 to configure SSL tunneling, since 3.1.1 requires a fix to be installed (see 10.1, “SSL Tunneling Scenario” on page 488).

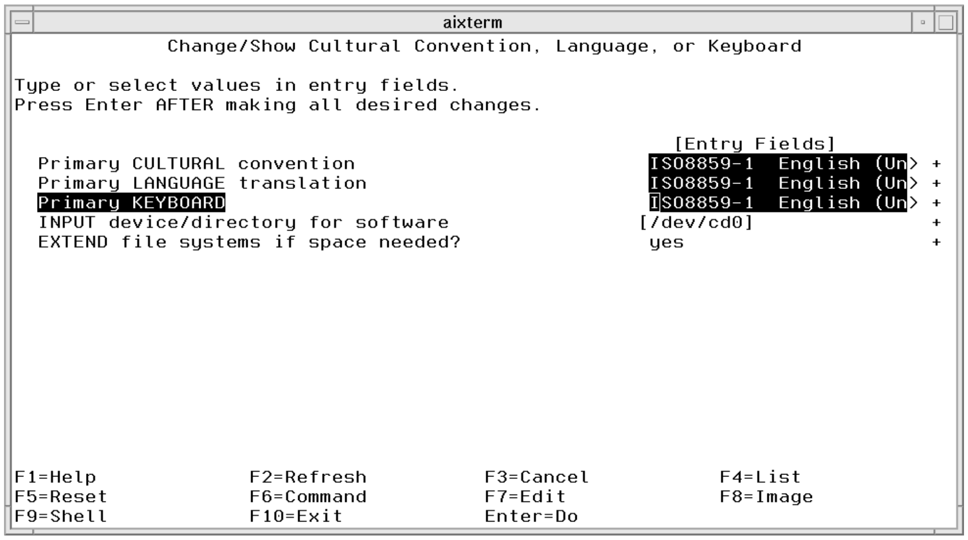
This explains why some of the tests that are described in this book use Version 3.1.1 of the IBM Firewall on AIX 4.2.0 while others use Version 3.2 on AIX 4.2.1.

7.7 Installation

Before you install the IBM Firewall, you must set what language environment you will use on your machine. In our environment, we did not use the C (POSIX) language environment, which is the default language environment of AIX. To change the language environment, we entered the command:

```
smitty mle_cc_cust_hdr
```

on an aixterm window. We selected **ISO8859-1 (en_US)** as our language environment, as you can see in the following screen:



```
aixterm
              Change/Show Cultural Convention, Language, or Keyboard

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

                                                      [Entry Fields]
  Primary CULTURAL convention                       ISO8859-1  English (Un> +
  Primary LANGUAGE translation                      ISO8859-1  English (Un> +
  Primary KEYBOARD                                  ISO8859-1  English (Un> +
  INPUT device/directory for software              [/dev/cd0]              +
  EXTEND file systems if space needed?              yes                    +

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

Figure 336. Language Environment Selection

You also need to install the following filesets:

1. For IBM Firewall 3.1.1 on AIX 4.2.0:
   - bos.net.tcp.server 4.2.0.0
   - bos.acct 4.2.0.0
   - bos.sysmgt.sysbr 4.2.0.0
   - X11.base.lib 4.2.1.0
   - bos.net.tcp.client 4.2.0.11
   - bos.rte.libc 4.2.0.11
   - bos.rte.libs 4.2.0.8
2. For IBM eNetwork Firewall 3.2 on AIX 4.2.1:
   - bos.net.tcp.server 4.2.1.0
   - bos.acct 4.2.1.0
   - bos.sysmgt.sysbr 4.2.1.0
   - bos.net.tcp.client 4.2.1.0
   - Java.rte 1.1.2
   - fw320aixprereqs.tar.Z package

Before installing the above filesets, you should enter the command:

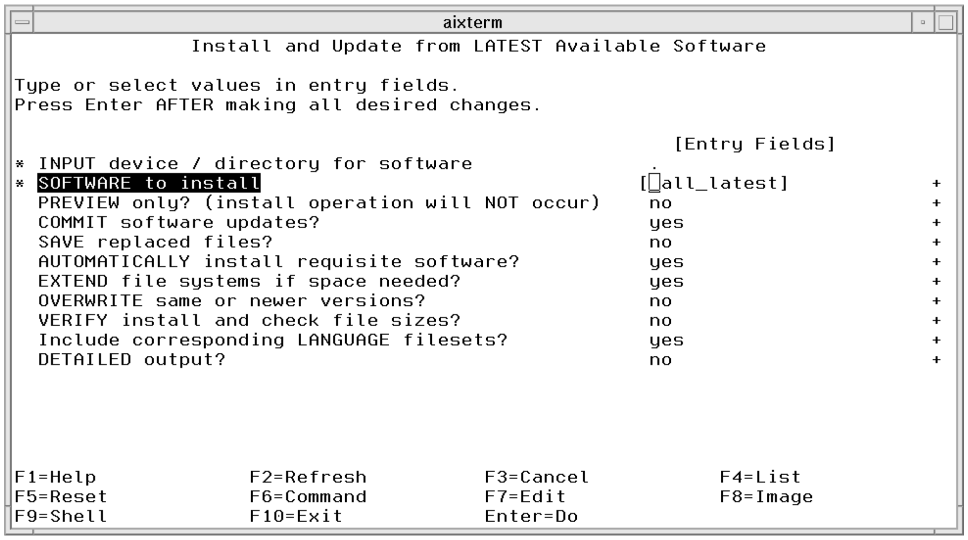
```
lslpp -l
```

to see if they are already installed on your machine. If they are needed, you can then use the command:

```
smitty install_latest
```

to install them.

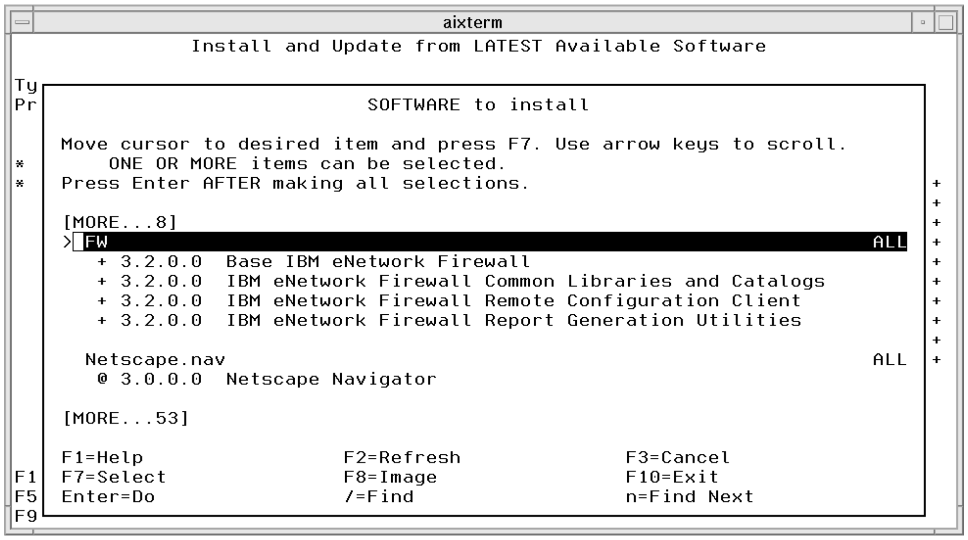
When this operation has been accomplished, you simply use `smit` to install the IBM Firewall, as shown in the following figure:



```
aixterm
                  Install and Update from LATEST Available Software

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

                                                        [Entry Fields]
* INPUT device / directory for software                  .
* SOFTWARE to install                                  [all_latest]          +
  PREVIEW only? (install operation will NOT occur)      no                   +
  COMMIT software updates?                              yes                  +
  SAVE replaced files?                                  no                   +
  AUTOMATICALLY install requisite software?             yes                  +
  EXTEND file systems if space needed?                  yes                  +
  OVERWRITE same or newer versions?                     no                   +
  VERIFY install and check file sizes?                  no                   +
  Include corresponding LANGUAGE filesets?              yes                  +
  DETAILED output?                                      no                   +




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

Figure 337. Installation of IBM Firewall Using smit

Pressing the F4 key, we could see all the software to install and we had the ability to select IBM Firewall by pressing the F7 key:



```
aixterm
                  Install and Update from LATEST Available Software

Ty
Pr                            SOFTWARE to install

    Move cursor to desired item and press F7. Use arrow keys to scroll.
*       ONE OR MORE items can be selected.
*   Press Enter AFTER making all selections.                                 +
                                                                             +
    [MORE...8]                                                               +
    > FW                                                                 ALL +
        + 3.2.0.0  Base IBM eNetwork Firewall                                +
        + 3.2.0.0  IBM eNetwork Firewall Common Libraries and Catalogs       +
        + 3.2.0.0  IBM eNetwork Firewall Remote Configuration Client         +
        + 3.2.0.0  IBM eNetwork Firewall Report Generation Utilities         +
                                                                             +
      Netscape.nav                                                       ALL +
        @ 3.0.0.0  Netscape Navigator

    [MORE...53]

    F1=Help                 F2=Refresh              F3=Cancel
F1  F7=Select               F8=Image                F10=Exit
F5  Enter=Do                /=Find                  n=Find Next
F9
```

Figure 338. Selecting IBM Firewall Software Using smit

7.8 Basic Configuration

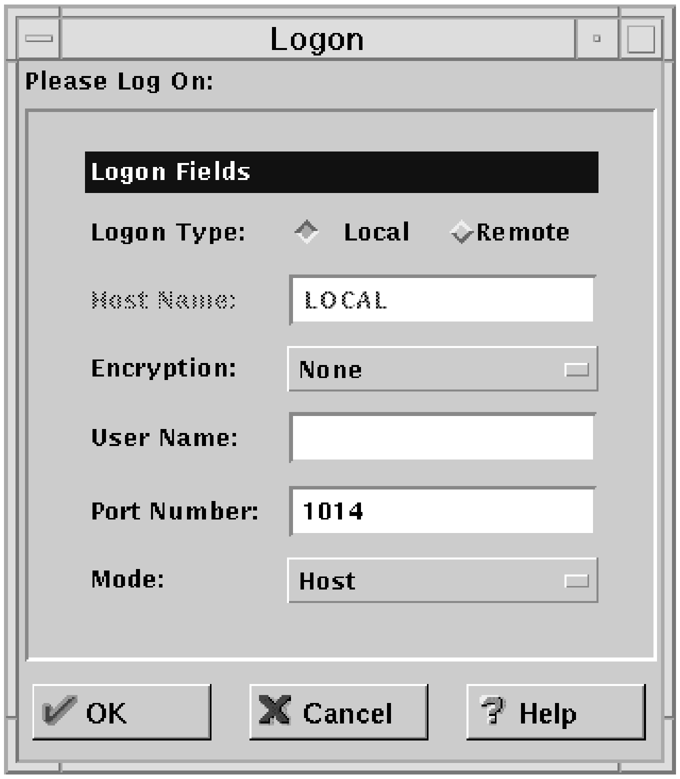
In order to configure the IBM Firewall, you have to set the secure network adapter, activate the remote configuration and add the administration user. In this section we will describe how to activate the remote configuration and how to add the administration user. The configuration of the secure network adapter may depend on the particular scenario that you want to implement and it can change depending on whether the scenario is based on a three-tier model or on a two-tier model. For this reason, you can refer to 8.4.4, “Setting the Secure Network Adapter on the Firewall” on page 389 and 9.5, “Firewall Basic Configuration” on page 444 to see how to set a network adapter on the firewall as secure.

7.8.1 Remote Configuration

IBM Firewall can be configured remotely. For this reason, you can install the Configuration Client on a remote machine, without the need to install the whole IBM Firewall package. Every time you install the IBM Firewall, the Configuration Client is automatically installed locally, so that you can configure the firewall on the same firewall machine.

We used the `fwconfig` program to configure the firewall, since it offers an easy Graphical User Interface (GUI) that makes the configuration easier. You could also use `smit`, but it is more difficult than `fwconfig` and it does not offer the same graphical facilities. The fwconfig program is launched in an aixterm window by entering the `fwconfig` command.

As soon as you enter the `fwconfig` command, the Logon page for the configuration program is displayed:



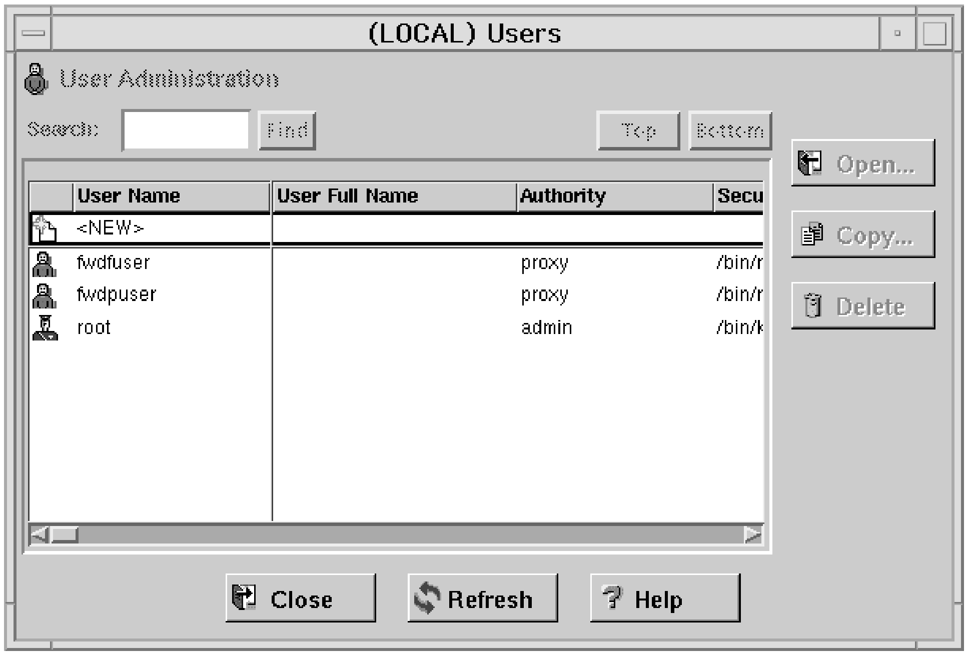
Logon Type: [✓ Local] Remote
Host Name: LOCAL
Encryption: None
User Name:
Port Number: 1014
Mode: Host

Figure 339. Logon Screen

We logged on as root to start the configuration. In fact, once the firewall is installed, root is the only user that has administrative privileges on the firewall. Logging on the Configuration Client as root, you can designate other users as firewall administrators, since root is the only user enabled to create new firewall administrators.

7.8.2 Administrator User Addition

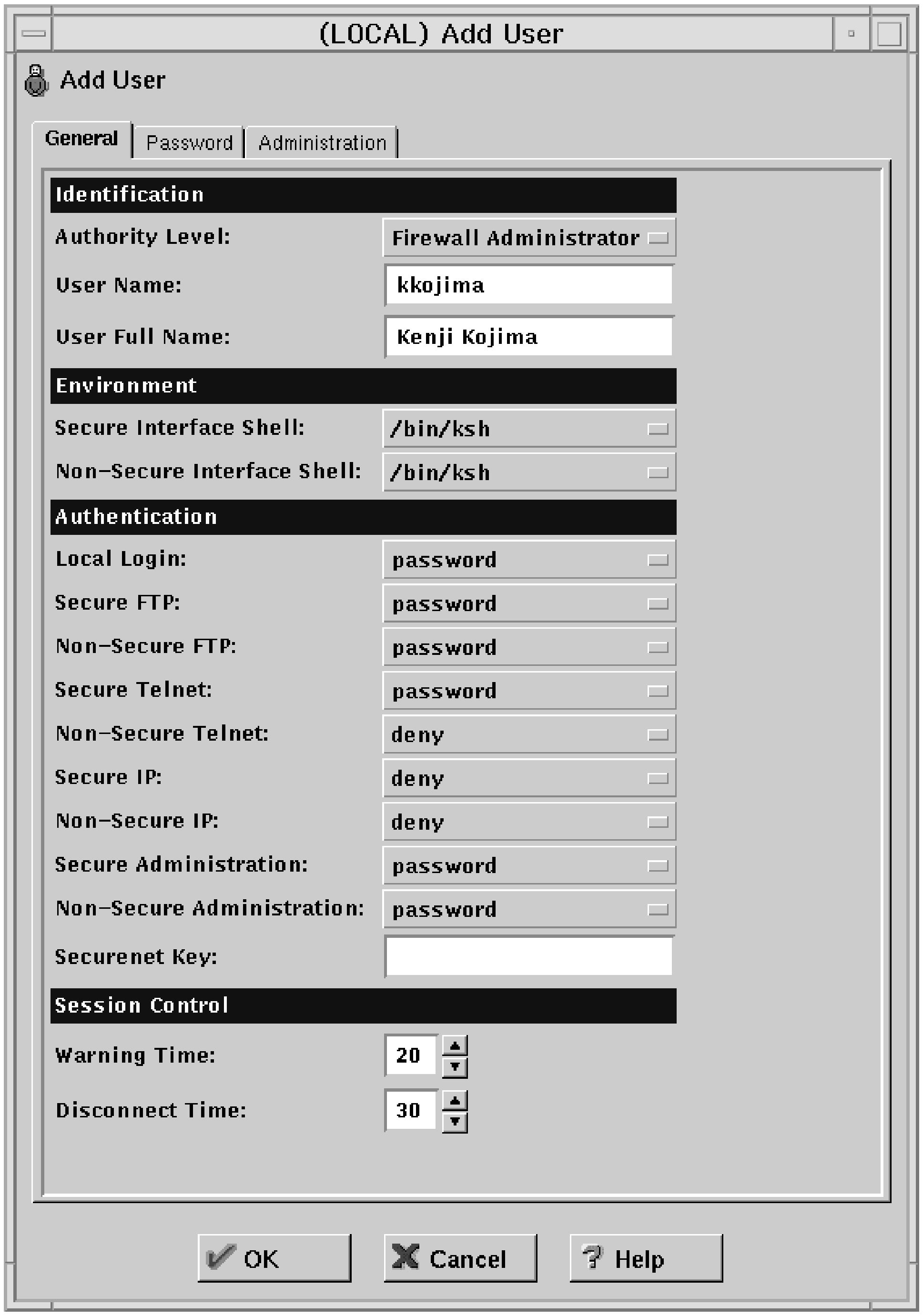
The `fwconfig` program allows you to define a new firewall administrator. Aftern launching the `fwconfig` command, you should log on as root and select **Users** from the Configuration Client navigation tree (see Figure 370 on page 392). You will get the window below:



User Name	User Full Name	Authority	Secu
<NEW>			
fwdfuser		proxy	/bin/r
fwdpuser		proxy	/bin/r
root		admin	/bin/k

Figure 340. Users Window

Then select **NEW** and click on **Open...**. The Add User window will be brought up:


(LOCAL) Add User

Add User

General | Password | Administration

Identification

Authority Level:	Firewall Administrator
User Name:	kkojima
User Full Name:	Kenji Kojima

Environment

Secure Interface Shell:	/bin/ksh
Non-Secure Interface Shell:	/bin/ksh

Authentication

Local Login:	password
Secure FTP:	password
Non-Secure FTP:	password
Secure Telnet:	password
Non-Secure Telnet:	deny
Secure IP:	deny
Non-Secure IP:	deny
Secure Administration:	password
Non-Secure Administration:	password
Securenet Key:	

Session Control

Warning Time:	20
Disconnect Time:	30

Figure 341. Add User Window

This dialog box is used to define the properties of the new administrator. Figure 341 shows the Add User window after we made all our choices.

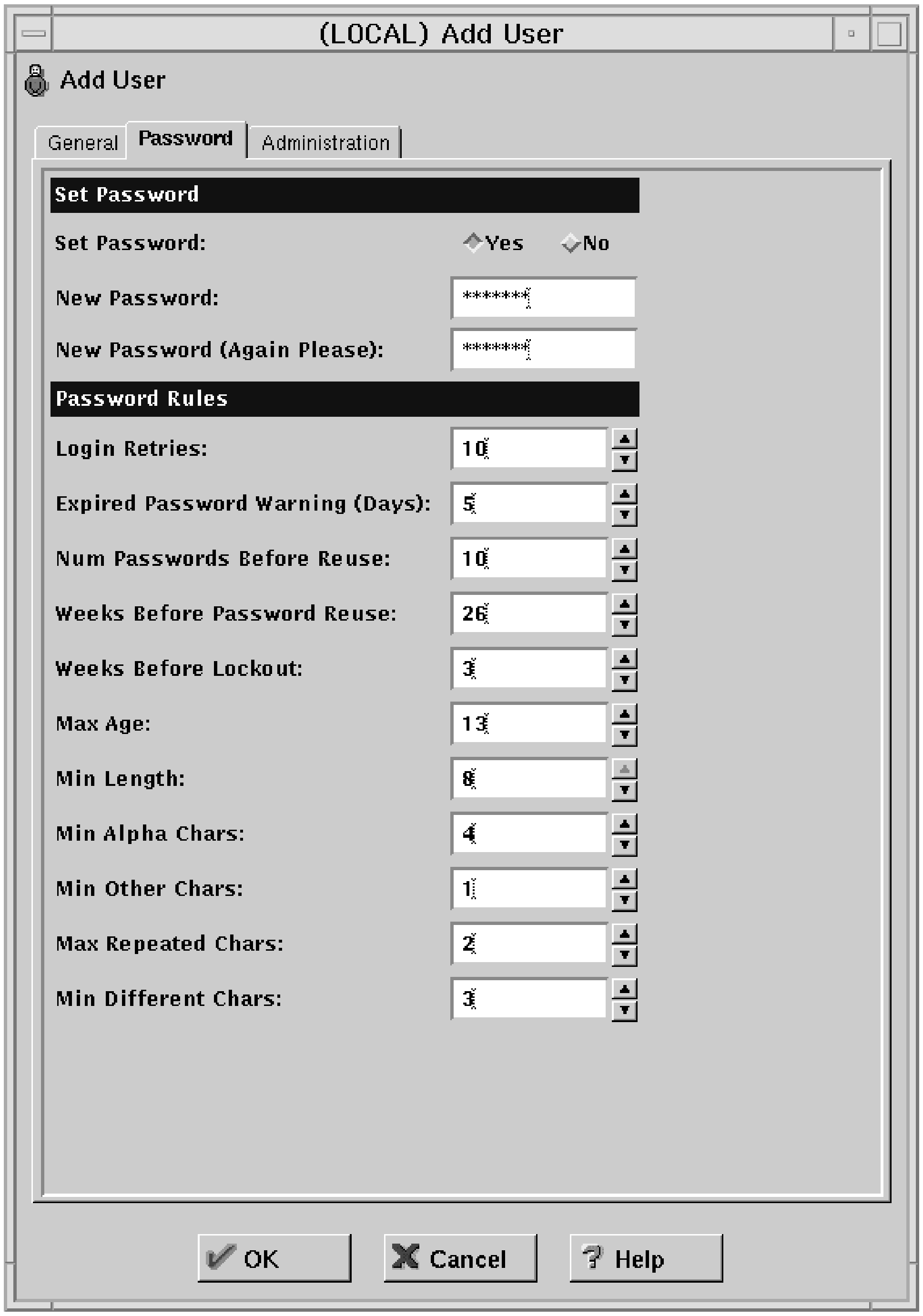
The Authority Level field by default is set to `Socks/Proxy User`, but in this case the user wouldn't have administrative privileges. We opened the pull-down menu and selected **Firewall Administrator**. Also we typed `kkojima` in the User Name field and `Kenji Kojima` in the User Full Name field.

The Environment section of the Add User window allows you to select the Secure Interface Shell, which is the shell program that will run when the user kkojima logs in from the secure network, and the Non-Secure Interface Shell, which is the shell program that will run when the user kkojima logs in from the network connected to the non-secure network interface. We opened the pull-down menu and selected **/bin/ksh** for both.

The Add User window also has a number of fields for the Authentication section. By default these fields are set to **deny**, meaning that the IBM Firewall forbids access to the server. As you can see in Figure 341 on page 355, we kept the default values only for Non-Secure Telnet, Secure IP and Non-Secure IP, but we changed it for all the other fields, selecting **password**, meaning that for those services the server asks for your password before letting you proceed. That password will not be displayed and it is the same password with which the user was added.

The Session Control section of the Add User window has two fields: Warning Time and Disconnect Time. Warning Time is the maximum time in minutes that a user can remain idle before a warning message is issued. We set this value to **20**. Disconnect Time is the maximum time in minutes that a user can remain idle before the user is disconnected. We set this value to **30**. Of course the value in the Disconnect Time field must be greater than the value in the Warning Time field

After all these fields were filled, we clicked the **Password** tab on the top of the Add User window:


(LOCAL) Add User

Add User

General | Password | Administration

Set Password

Set Password:	[✓ Yes] No
New Password:	*******
New Password (Again Please):	*******

Password Rules

Login Retries:	10
Expired Password Warning (Days):	5
Num Passwords Before Reuse:	10
Weeks Before Password Reuse:	26
Weeks Before Lockout:	3
Max Age:	13
Min Length:	8
Min Alpha Chars:	4
Min Other Chars:	1
Max Repeated Chars:	2
Min Different Chars:	3

Figure 342. Setting Password and Password Rules for the Firewall Administrator

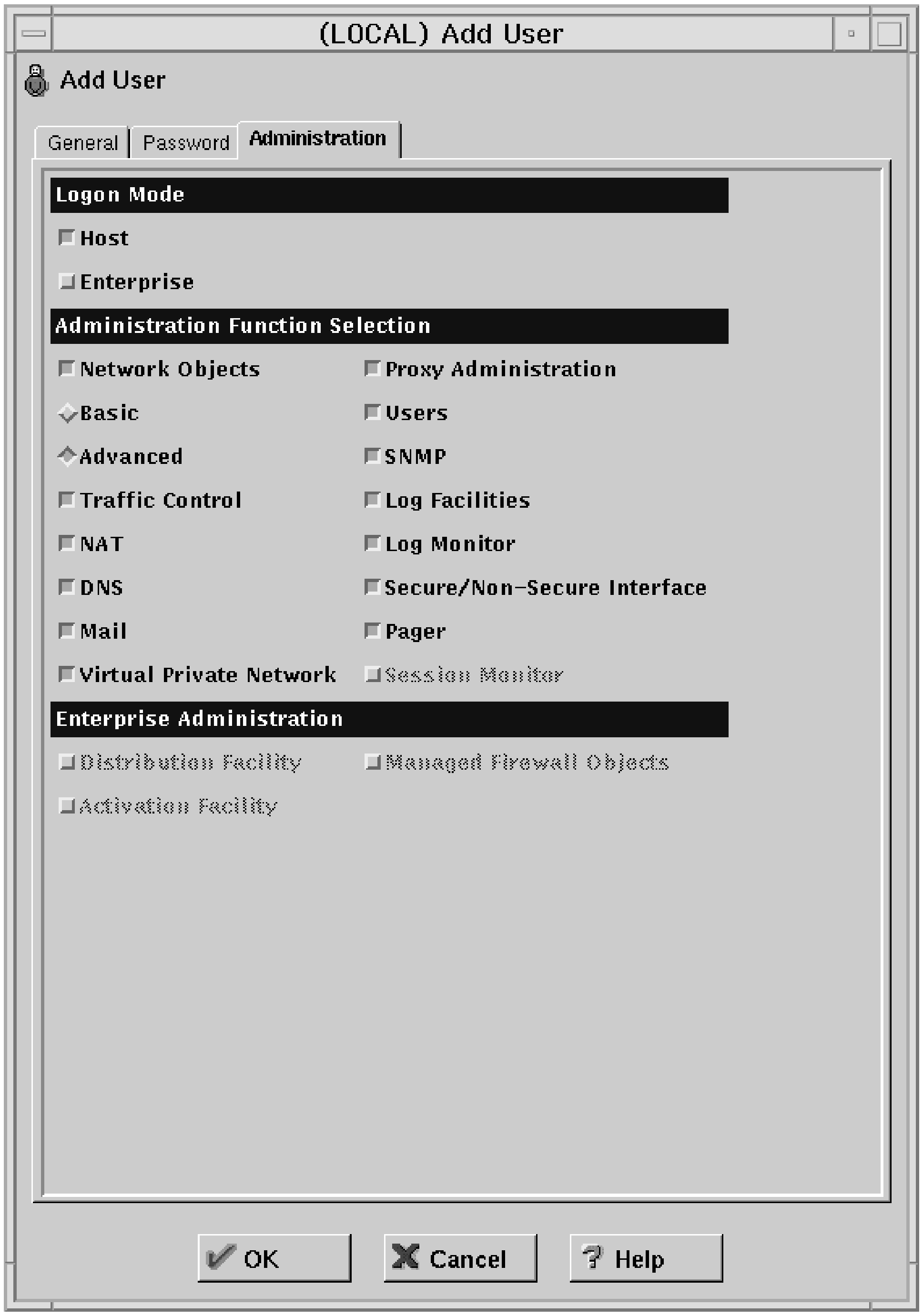
You should select a password carefully, especially if you have permitted to configure the firewall remotely. However, the new administrator user that you have defined will have to change the password the first time they log on.

The Password Rules section gives you the possibility to set eleven password rules:

1. Login retries
2. Number of days to warn the user before the password expires
3. Number of passwords before reuse

4. Weeks before password expiration
5. Weeks before password lockout
6. Maximum age of the password
7. Minimum length of the password
8. Minimum alphabetic characters
9. Minimum other characters
10. Maximum number of repeated characters
11. Minimum number of different characters

We changed the default values as shown in Figure 342 on page 357. Then we pressed the **Administration** tab on the top of the Add User window:


(LOCAL) Add User

Add User

General | Password | Administration

Logon Mode

- [✓] Host
- [ ] Enterprise

Administration Function Selection

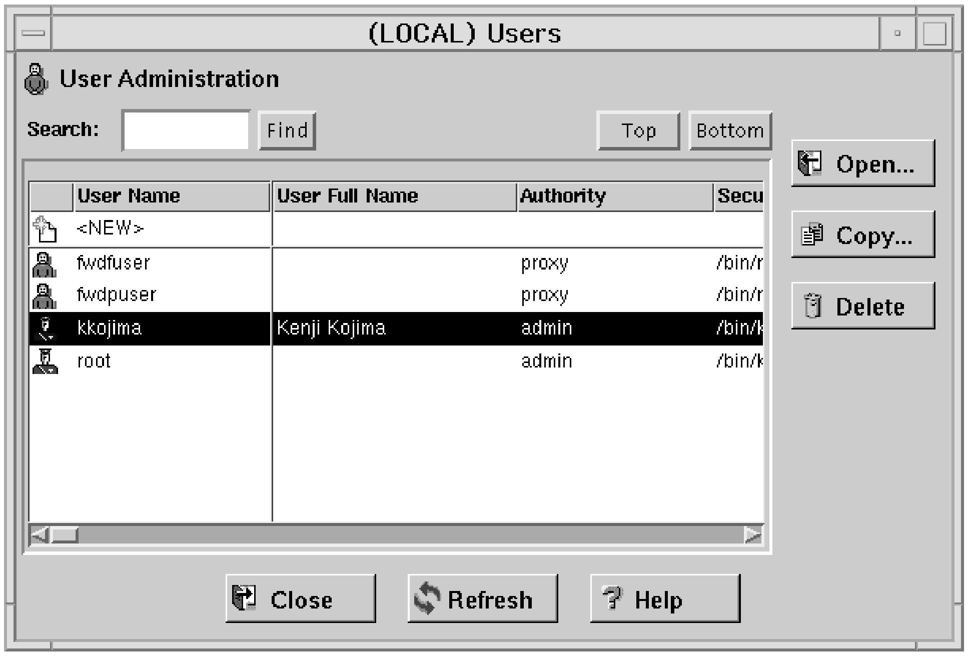
[✓] Network Objects	[✓] Proxy Administration
◇ Basic	[✓] Users
◆ Advanced	[✓] SNMP
[✓] Traffic Control	[✓] Log Facilities
[✓] NAT	[✓] Log Monitor
[✓] DNS	[✓] Secure/Non-Secure Interface
[✓] Mail	[✓] Pager
[✓] Virtual Private Network	[ ] Session Monitor

Enterprise Administration

[ ] Distribution Facility	[ ] Managed Firewall Objects
[ ] Activation Facility	

Figure 343. Setting Administration Properties

We accepted all default values in this window and then we clicked **OK**. The following window confirmed that the user kkojima had been added successfully:


(LOCAL) Users

User Administration

Search: Find Top Bottom

	User Name	User Full Name	Authority	Secu
	<NEW>			
	fwdfuser		proxy	/bin/r
	fwdpuser		proxy	/bin/r
	kkojima	Kenji Kojima	admin	/bin/k
	root		admin	/bin/k

Figure 344. Confirmation Screen for the New User

Chapter 8. Three-Tier Applications in Firewall-Protected Network Environments

This chapter describes how to build a three-tier secure application in an NCF environment protected by a firewall.

8.1 Overview

The three-tier architecture in our test environment, shown in Figure 345 on page 362, consisted of three machines:

1. Client tier

   This machine was a Windows 95 IBM ThinkPad installed with the Netscape Communicator 4.05 Web browser.

2. Server tier

   This machine was a Windows NT Server 4.0 IBM Personal Computer. It was installed with Lotus Domino Go Webserver Version 4.6.2.2; IBM ServletExpress Version 1.0 beta 2.1 had the role of servlet engine and IBM DB2 Client Application Enabler Version 5.0 was installed as connector. Sun Java Development Kit 1.1.6 was also installed on this machine.

3. Application server tier

   In our scenarios, IBM DB2 Universal Database Version 5.0 for AIX was installed on the third-tier machine, which was an IBM RISC/6000.

The tiers were connected through a set of standard protocols:

1. HTTP, HTTPS and TCP/IP were used between the client and the server tier.
2. IBM DB2 local protocol over TCP/IP was used between the server tier and the application server tier machine.

In view of building the firewall, we consider the simple case which supposes that an employee needs to retrieve corporate information through the Internet, perhaps from a foreign country. The user can connect with the corporate system using the nearby access point of a service provider, as if they were working in their office. But the corporate system must be protected against hackers and intruders and we need to build a firewall to provide this security service. A firewall is a tool combining hardware and software that separates a protected network from an unprotected network. The default rule for a firewall is to deny any traffic flow between a secure and a non-secure network. Our basic policy was to select a set of protocols that were permitted to pass through the firewall and a list of services that we wanted to run on our firewall. Then we set the type of permissions we wanted to guarantee for each service (direction of the connection, list of machines where a connection can be issued and eventually others). In other words, we defined a set of connections that allow specific types of IP traffic to flow between secure and non-secure networks, under certain conditions.

The user contacts the Web server using a Web browser installed on a client machine. The interaction between the client machine and the DB2 server is handled by the server tier, where Lotus Domino Go Webserver is installed. A Java servlet running in the Web server is automatically activated by the user's request.

Since the purpose of this chapter is to demonstrate how to use different technologies in a three-tier model environment to grant security, we did not write the code for a new application, but we built this scenario using the JDBCServlet sample application provided by ServletExpress 1.0.

Note: Although ServletExpress has been incorporated in the IBM WebSphere Application Server 1.0, you can safely refer to what we describe here to build an NCF secure three-tier application using the JDBCServlet, since no major modifications have been applied. Only notice that the directory structure has been modified.

The flow of our application is described in the following list:

1. The user accesses the application page on the Web site of his company using a common Web browser.
2. The Web server sends to the browser an HTML page containing a form.
3. The user fills out the form and submits it.
4. The Web server invokes the Java servlet specified by the HTML form.
5. The Java servlet, based upon the JDBC protocol, connects to the DB2 CAE client application located on the same machine by using the JDBC driver.
6. The DB2 CAE forwards the request coming from the servlet to the DB2 UDB database server application, located on the backend server machine.
7. The DB2 UDB database server application sends the requested data back to the DB2 CAE database client application.
8. The DB2 CAE database client application inside the Web server machine passes the retrieved data back to the servlet by using the JDBC driver.
9. The Java servlet dynamically formats the response in an HTML page and passes it to the Web server.
10. The Web server sends the dynamic HTML page back to the client tier, that displays the retrieved data on the Web browser.

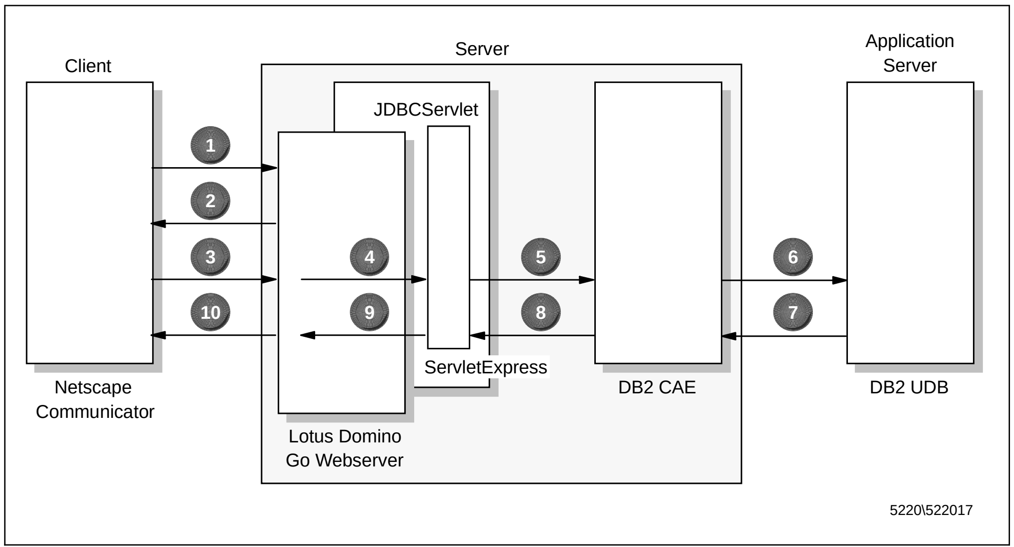


Figure 345. Graphical Representation of the Application Flow

In order to grant security in our firewall-based scenario, we used all the security measures listed below:

- SSL with client authentication
- IP filters for HTTP and SSL in the client/server communication
- IP filters for the DB2 communication protocol over TCP/IP used between the IBM DB2 CAE client and the IBM DB2 UDB server application
- WebSphere access control
- DB2 access control based on user ID and password

The following figure offers a logical representation of the security measures we applied to our environment:

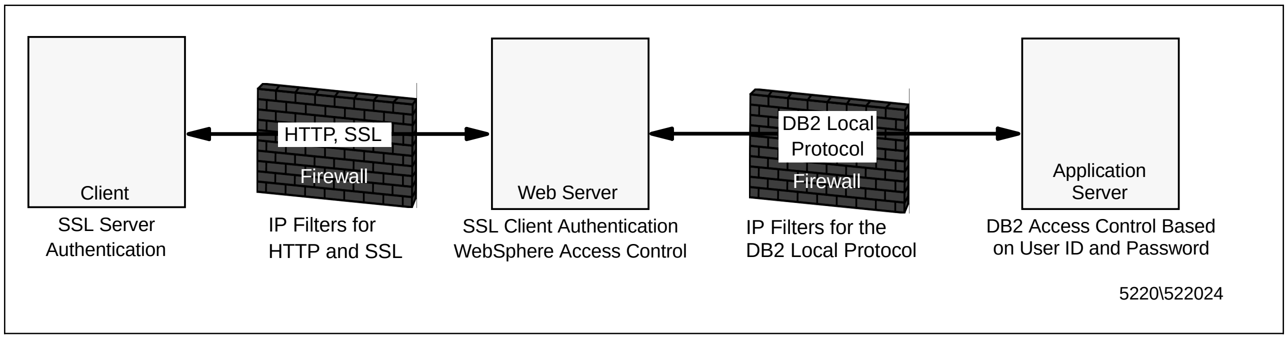

Client
HTTP, SSL
Firewall
Web Server
DB2 Local Protocol
Firewall
Application Server
SSL Server Authentication
IP Filters for HTTP and SSL
SSL Client Authentication WebSphere Access Control
IP Filters for the DB2 Local Protocol
DB2 Access Control Based on User ID and Password
5220\522024

Figure 346. Logical Representation of the Security Measures Applied

Actually we used only one firewall machine with three network adapters, as shown in Figure 367 on page 387, and we connected it to the three tiers of our scenario. This solution is logically equivalent to implementing two firewalls, as explained in 6.2.4, “Screened Subnet” on page 292.

The JDBCServlet that we used in this scenario is invoked from a static HTML page that, as you will see, does not have a sensitive nature. For this reason, we didn't protect access to that HTML page by using the access control features provided by Lotus Domino Go Webserver. Instead, we concentrated our efforts on protecting access to the JDBCServlet, and to do this we used the security measures offered by ServletExpress. If your application requires access control for Web pages, you can set it up by following the steps we described in 3.3, “Access Control with Lotus Domino Go Webserver” on page 104 and 3.3.6, “Lotus Domino Go Webserver Access Control Lists” on page 114.

In order to see how to install and configure the key products that were used in this scenario, you can refer to Chapter 7, “How to Install and Configure a NCF Secure Environment” on page 297.

8.2 System Configuration

This section describes the operations we did after installing all the products, in order to configure the environment and to have the JDBC application run in our three-tier scenario.

We configured the environment before the firewall was activated.

8.2.1 CLASSPATH Variable and ncf.jvm.classpath Property

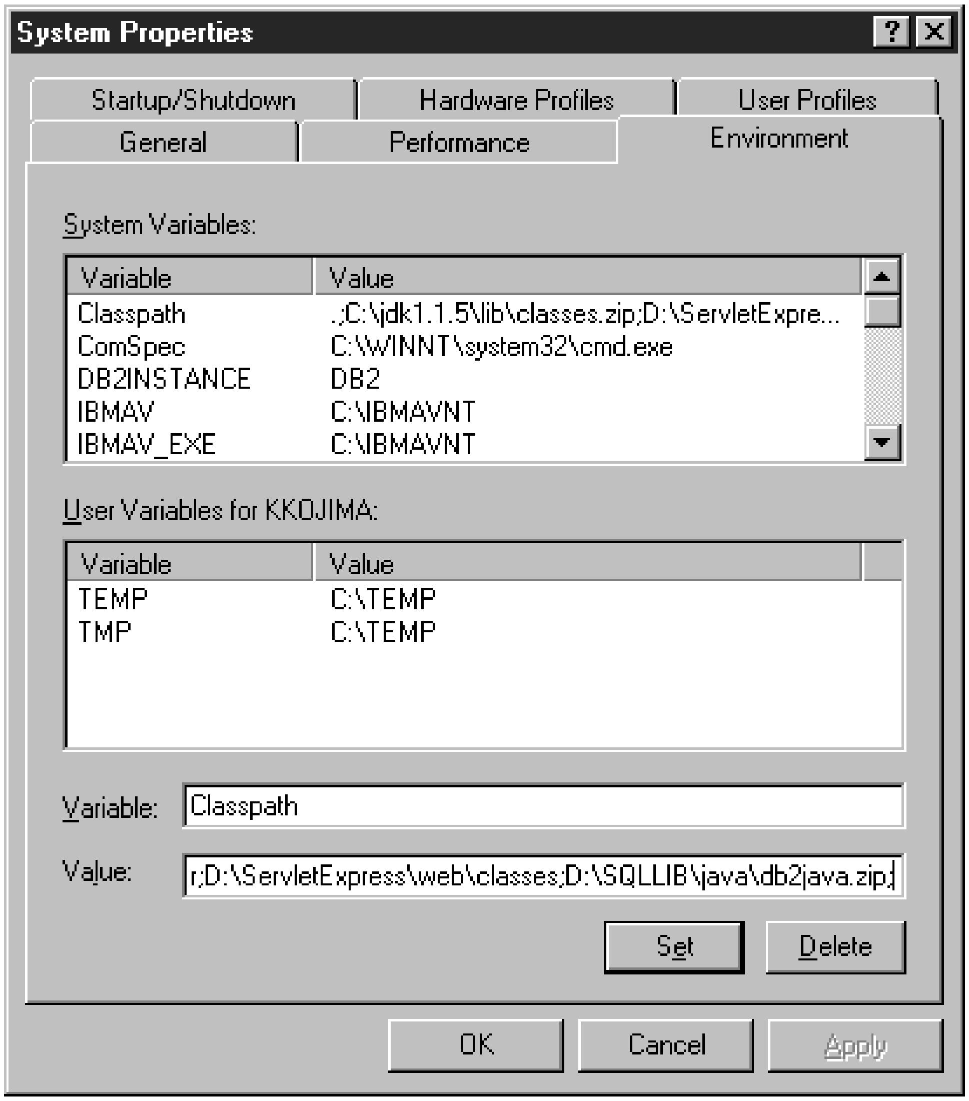
IBM DB2 CAE includes Java class files that are necessary to implement applications that interface to DB2. In particular, the JDBCServlet application that we were going to install interfaces to the DB2 CAE through the JDBC protocol and it uses the JDBC driver COM.ibm.db2.jdbc.app.DB2Driver.class. We can understand this if we look at these few lines of code for the JDBCServlet.java file:

```
try
{
  // register the driver with DriverManager
  Class.forName("COM.ibm.db2.jdbc.app.DB2Driver");
}
catch (ClassNotFoundException e)
{
  e.printStackTrace();
}
```

Once the Class.forName method is invoked, the string COM.ibm.db2.jdbc.app.DB2Driver is passed to it as a parameter. In this way the forName() method locates, loads and links the class DB2Driver, which is the JDBC driver, in the package COM.ibm.db2.jdbc.app.

This Java class is part of the file D:\SQLLIB\java\db2java.zip, so we needed to include this zip file in the CLASSPATH environment variable in order for the Java compiler `javac` to find the DB2Driver class and we also needed to add the same zip file in the ncf.jvm.classpath property for the jvm.properties file (see Figure 308 on page 326). This value represents the Java classpath that the servlet engine will use to make a servlet run.

To modify the value for the CLASSPATH system environment variable, we opened the System Properties window for Windows NT (in 7.1.1, “Path System Environment Variable” on page 299 we explained how to have access to the System Properties window), we added `D:\SQLLIB\java\db2java.zip` to the current value of the CLASSPATH variable (see Figure 302 on page 320) and clicked the **Apply** button:


System Properties

System Variables:

Variable	Value
Classpath	.;C:\jdk1.1.5\lib\classes.zip;D:\ServletExpre...
ComSpec	C:\WINNT\system32\cmd.exe
DB2INSTANCE	DB2
IBMAV	C:\IBMAVNT
IBMAV_EXE	C:\IBMAVNT

User Variables for KKOJIMA:

Variable	Value
TEMP	C:\TEMP
TMP	C:\TEMP

Variable: Classpath

Value: r;D:\ServletExpress\web\classes;D:\SQLLIB\java\db2java.zip;

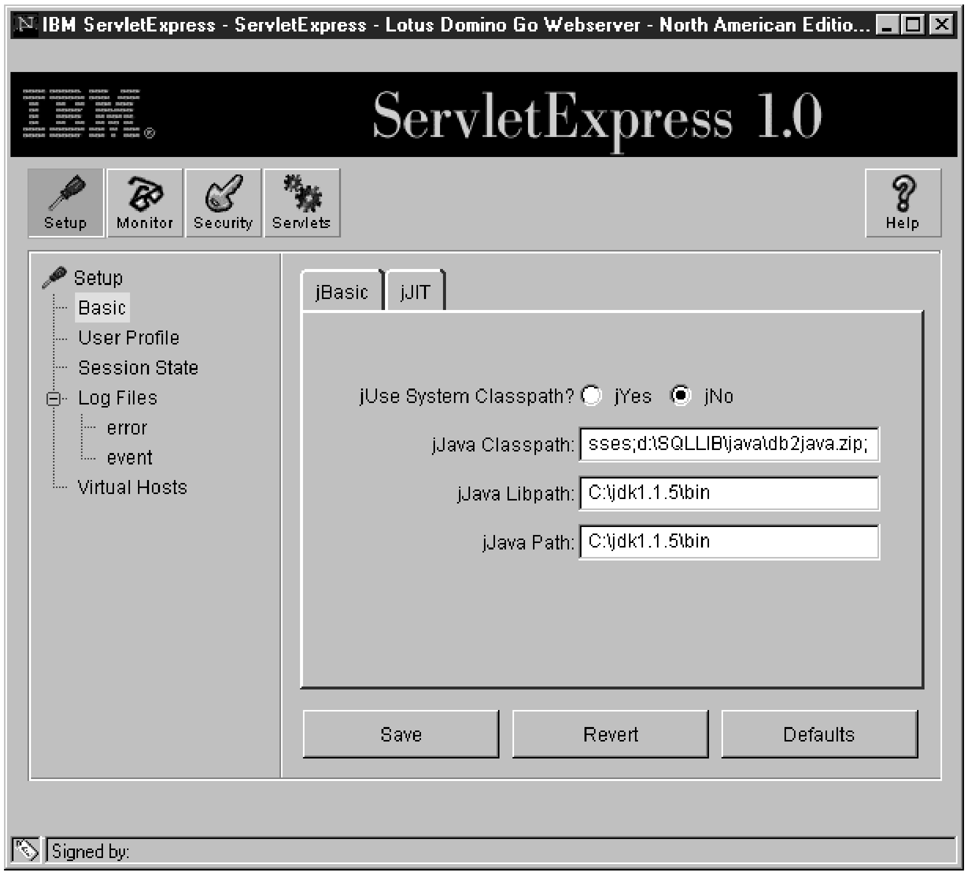
Figure 347. CLASSPATH System Environment Variable

It was then possible to run the Java compiler `javac` against the file JDBCServlet.java. To do this, we opened an MS-DOS Command Prompt window and entered the command:

```
javac JDBCServlet.java
```

when D:\ServletExpress\web\resources\en_US\JDBCServlet was the current directory.

To modify the ncf.jvm.property for the jvm.properties file, it is still possible to open that file with a text editor different from Notepad and edit the property, as we did in 7.3.4, “How to Configure ServletExpress Log Files” on page 322 to enable native DLL logging and Java standard out logging. ServletExpress gives you also the possibility to edit that property through the ServletExpress Manager. It is enough to select the **Basic** item for the Setup menu and to click on **Save** after having added `D:\SQLLIB\java\db2java.zip` to the current value of the Java Classpath variable:



jUse System Classpath? jYes [✓ jNo]
jJava Classpath: sses;d:\SQLLIB\java\db2java.zip;
jJava Libpath: C:\jdk1.1.5\bin
jJava Path: C:\jdk1.1.5\bin

Figure 348. Setting the ncf.jvm.classpath Property for ServletExpress

8.2.2 Servlet Registration in ServletExpress

The JDBCServlet application, that ServletExpress includes as a sample servlet in the directory *servletexpress_root*\web\resource\en_US\JDBCServlet, was the application that we used to test our firewall-based security scenario. Since the servlet engine used by Lotus Domino Go Webserver was provided by ServletExpress, we had to use the ServletExpress Manager to register the JDBCServlet as a new servlet.

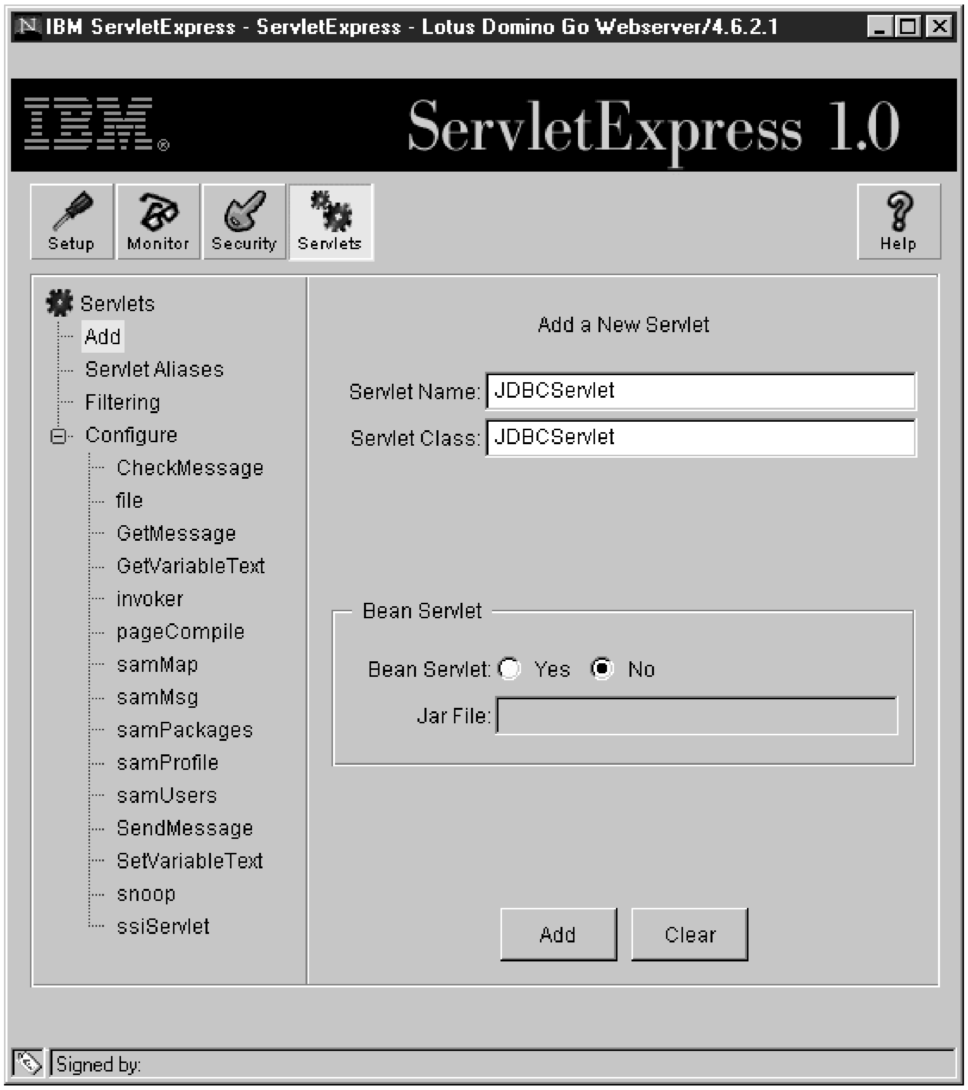
First of all, we had to compile the servlet as indicated in 8.2.1, “CLASSPATH Variable and ncf.jvm.classpath Property” on page 364 and then we had to copy or move the resulting class files JDBCServlet.class and ConInfo.class into the servlet directory *servletexpress_root*\servlets. The *Web Programmer's Guide* provided with ServletExpress 1.0 suggests compiling each sample servlet in its own individual directory and only then to copy or move the resulting class files. This is important in order for the Java compiler `javac` to find all the packages that are imported. If you move the java file in the servlet directory and then you compile it, you run the risk that the Java compiler does not find the necessary packages. We noticed that the JDBCServlet.java file can be compiled without problems in the servlet directory.

We also had to run the Java compiler `javac` against the messages resource file samples_en_US.java, that comes in the *servletexpress_root*\web\resources\en_US directory. We simply entered the command:

```
javac samples_en_US.java
```

in an MS-DOS Command Prompt window. *servletexpress_root*\web\resources\en_US was the current directory. Then the samples_en_US.class file also had to be copied or moved into the servlet directory. Notice that the JDBCServlet does not work if the samples_en_US.class file is not in the servlet directory.

Once these installations are completed, you should launch the Web server and then the ServletExpress Manager. Following all the steps that we explained in 7.3.5, “How to Add Servlets into ServletExpress” on page 327, you should be able to register the JDBCServlet servlet in ServletExpress using the ServletExpress Manager.



Add a New Servlet

Servlet Name: JDBCServlet

Servlet Class: JDBCServlet

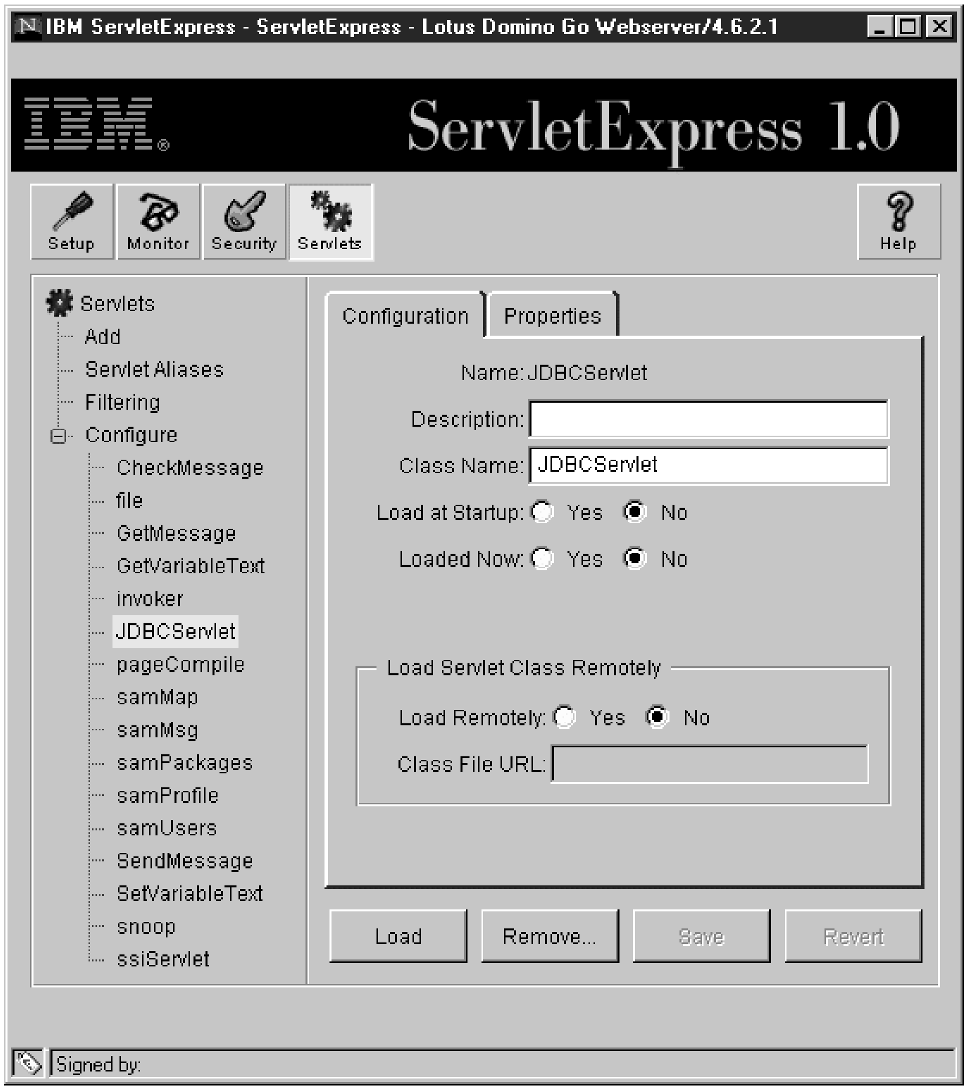
Bean Servlet

Bean Servlet: Yes [✓ No]

Jar File:

Figure 349. JDBCServlet Registration through the ServletExpress Manager

After having entered the Servlet Name and the Servlet Class both as `JDBCServlet`, we pressed the **Add** button. Then the servlet appeared in the list of all the available servlets and the Configuration window appeared:



Name: JDBCServlet

Description:

Class Name: JDBCServlet

Load at Startup: Yes [✓ No]

Loaded Now: Yes [✓ No]

Load Servlet Class Remotely

Load Remotely: Yes [✓ No]

Class File URL:

Figure 350. JDBCServlet Configuration Window

We accepted the default values displayed in the Configuration window. Since this servlet does not have any initial parameters, we did not have to click on the **Property** tag on the top of the window to enter a name and value for each parameter.

8.3 Testing the Application without Using the Firewall

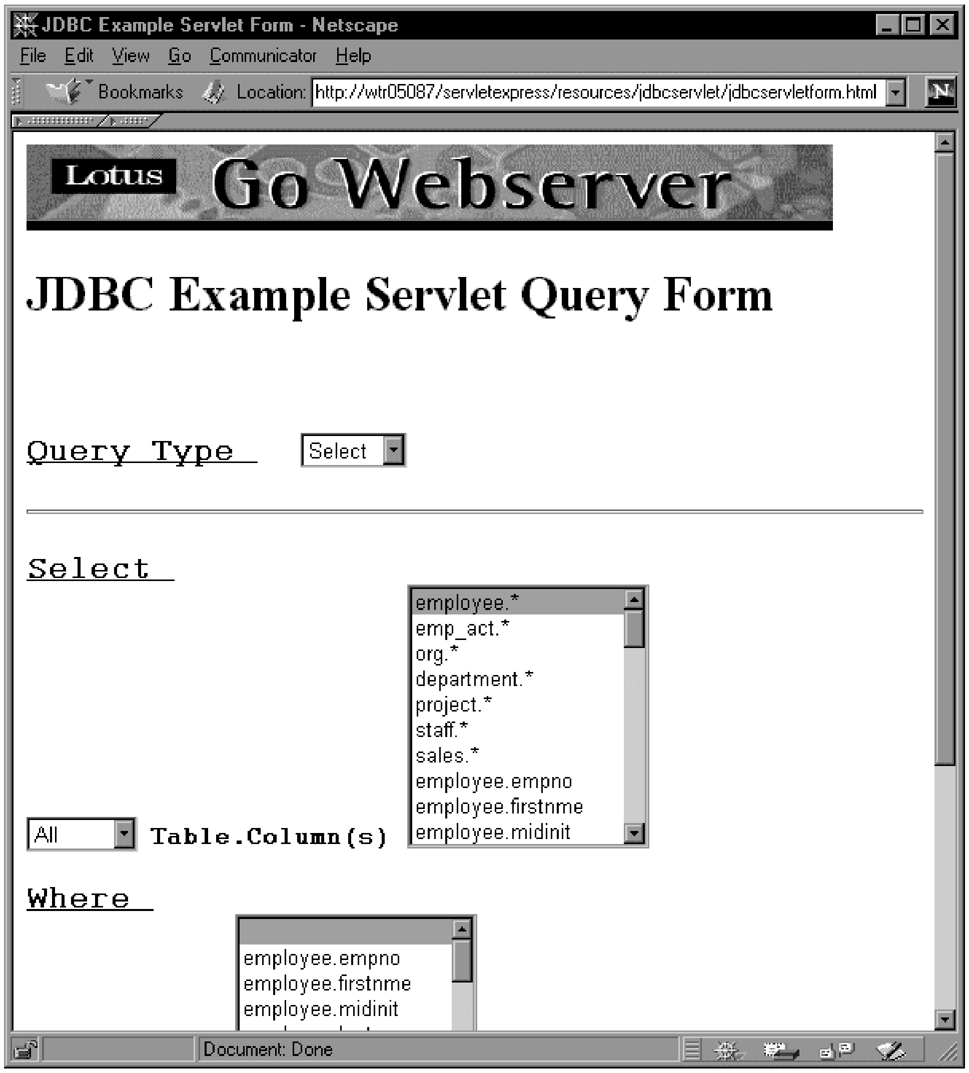
After the parts were installed on the three systems for the scenario, it was time to test our work before proceeding with the firewall installation. We had been making configuration decisions all along the way, and we were able to test some of them, but the real test would be a Web page, a servlet and a database.

The JDBCServlet.class file is called from an HTML form also supplied as a sample. Its name is JDBCServletForm.html. After installing ServletExpress, the file JDBCServletForm.html is placed in the directory *servexp_root*\web\resources\en_US\JDBCServlet.

If you examine the Lotus Domino Go Webserver configuration file, named httpd.cnf, you will find the following mapping rules added by the ServletExpress installation:

```
Pass /ServletExpress/resources/* d:\ServletExpress\web\resources\en_US\*
Pass /ServletExpress/* d:\ServletExpress\web\*
```

For our test, the HTML source would be referred to as http://*ourwebserver*/servletexpress/resources/JDBCServlet/JDBCServletForm.html Pointing the Web browser on the client machine to this URL, we saw the HTML form displayed:


JDBC Example Servlet Form - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://wtr05087/servletexpress/resources/jdbcservlet/jdbcservletform.html
Lotus Go Webserver
Document: Done

JDBC Example Servlet Query Form

Query Type Select

Select

All Table.Column(s)

employee.*
emp_act.*
org.*
department.*
project.*
staff.*
sales.*
employee.empno
employee.firstnme
employee.midinit

Where

employee.empno
employee.firstnme
employee.midinit

Figure 351. JDBCServlet Access Form

Our first attempt to use the form produced an error from DB2. The sample database was created with user ID and password security. The DB2 error told us that we had not presented a user ID and password. We examined with a text editor the JDBCServlet.java source code file for the JDBCServlet servlet and there were no provisions for a user ID and password. One possibility to solve this problem was to hard code a user ID and a password into the JDBCServlet.java file. The code lines we added are shown in the following screen:

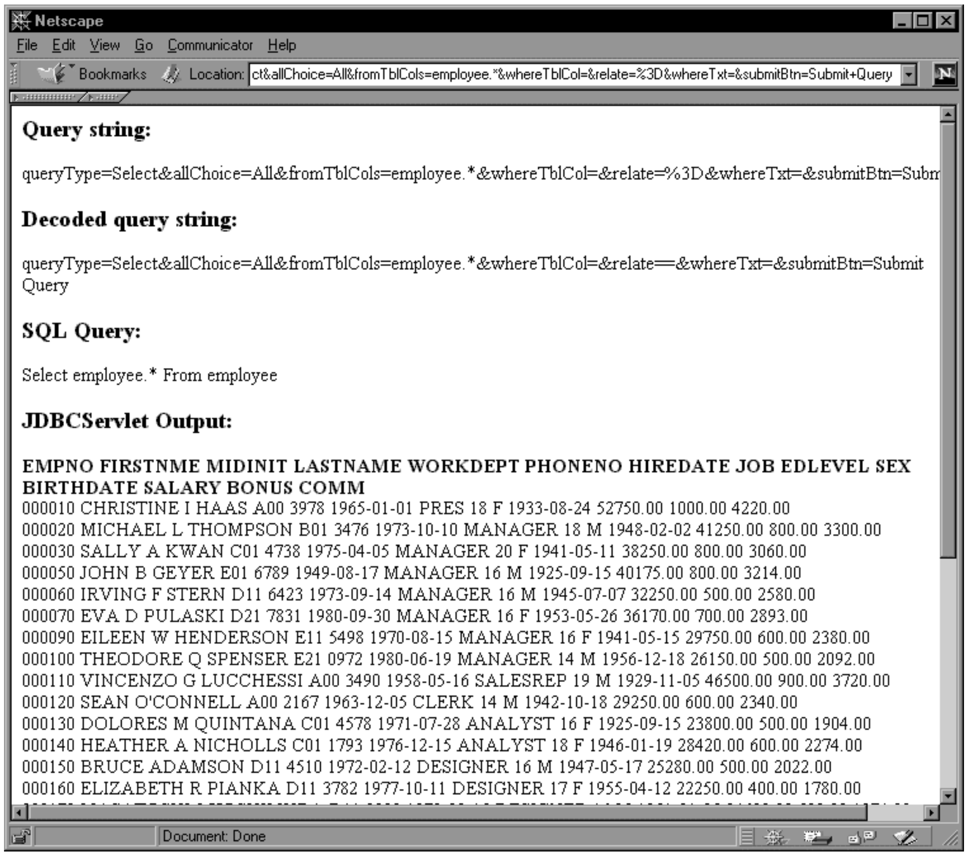
```
   :::
String userid = "db2inst1";
String password = "ibmdb2";

try
{
   // connect with default user ID and password
   con0 = DriverManager.getConnection(url, userid, password);
   con1 = DriverManager.getConnection(url, userid, password);
}

catch (Exception e)
{
   :::
```

We then recompiled the JDBCServlet.java file in its original directory *servexp_root*\web\resources\en_US\JDBCServlet and then copied the resulting class files to the servlet directory.

We restarted the Web server to make sure that the new servlet was activated. Because we had not selected to have the JDBCServlet loaded when ServletExpress starts (see Figure 350 on page 368), stopping and restarting the Web server had only the effect of stopping the servlet. Then the servlet would be reloaded when the first client invoked it.

We could have also restarted the servlet by using the ServletExpress Manager. In fact in the JDBCServlet Configuration window (see again Figure 350 on page 368) the **Load** button changes to read **Unload** if you press it and then **Unload** changes to read **Load** if you press the button again. In this way, you can restart only the servlet, without completely shutting down the Web server and ServletExpress.

The following test of the form was successful and resulted in the return of the data in the HTML page:



Netscape

File Edit View Go Communicator Help

Bookmarks Location: ct&allChoice=All&fromTblCols=employee.*&whereTblCol=&relate=%3D&whereTxt=&submitBtn=Submit+Query

Query string:

queryType=Select&allChoice=All&fromTblCols=employee.*&whereTblCol=&relate=%3D&whereTxt=&submitBtn=Subm

Decoded query string:

queryType=Select&allChoice=All&fromTblCols=employee.*&whereTblCol=&relate==&whereTxt=&submitBtn=Submit Query

SQL Query:

Select employee.* From employee

JDBCServlet Output:

```
EMPNO FIRSTNME MIDINIT LASTNAME WORKDEPT PHONENO HIREDATE JOB EDLEVEL SEX
BIRTHDATE SALARY BONUS COMM
000010 CHRISTINE I HAAS A00 3978 1965-01-01 PRES 18 F 1933-08-24 52750.00 1000.00 4220.00
000020 MICHAEL L THOMPSON B01 3476 1973-10-10 MANAGER 18 M 1948-02-02 41250.00 800.00 3300.00
000030 SALLY A KWAN C01 4738 1975-04-05 MANAGER 20 F 1941-05-11 38250.00 800.00 3060.00
000050 JOHN B GEYER E01 6789 1949-08-17 MANAGER 16 M 1925-09-15 40175.00 800.00 3214.00
000060 IRVING F STERN D11 6423 1973-09-14 MANAGER 16 M 1945-07-07 32250.00 500.00 2580.00
000070 EVA D PULASKI D21 7831 1980-09-30 MANAGER 16 F 1953-05-26 36170.00 700.00 2893.00
000090 EILEEN W HENDERSON E11 5498 1970-08-15 MANAGER 16 F 1941-05-15 29750.00 600.00 2380.00
000100 THEODORE Q SPENSER E21 0972 1980-06-19 MANAGER 14 M 1956-12-18 26150.00 500.00 2092.00
000110 VINCENZO G LUCCHESSI A00 3490 1958-05-16 SALESREP 19 M 1929-11-05 46500.00 900.00 3720.00
000120 SEAN O'CONNELL A00 2167 1963-12-05 CLERK 14 M 1942-10-18 29250.00 600.00 2340.00
000130 DOLORES M QUINTANA C01 4578 1971-07-28 ANALYST 16 F 1925-09-15 23800.00 500.00 1904.00
000140 HEATHER A NICHOLLS C01 1793 1976-12-15 ANALYST 18 F 1946-01-19 28420.00 600.00 2274.00
000150 BRUCE ADAMSON D11 4510 1972-02-12 DESIGNER 16 M 1947-05-17 25280.00 500.00 2022.00
000160 ELIZABETH R PIANKA D11 3782 1977-10-11 DESIGNER 17 F 1955-04-12 22250.00 400.00 1780.00
```

Document: Done

Figure 352. JDBCServlet Response Showing Data Retrieved

Putting the user ID and password for the database access into the Java source code showed us that the HTML form and servlet combination would work, and that we configured the major pieces of our application properly. If you implement this solution, you give access to the database to whomever accesses the servlet, since DB2 automatically authenticates and authorizes the user. You should then use the security features provided by ServletExpress to protect your servlet and give access to it only to authorized users.

We wanted to change the implementation a little for security purposes by allowing the Web page user to put their user ID and password into the form and have the servlet pick up those values and pass them along to the database. This would allow us to have DB2 security from the second-tier Web server to the third-tier database server.

We needed to make changes to the HTML form and to the Java source code to accomplish this goal. We added the following lines of code to the HTML form:

```
<p>
User ID
<BR>
<INPUT Type="text" Name="userid" Size="8">

<p>
Password
<BR>
<INPUT Type="password" Name="passwd" Size="8">
```

The changes to the java code were a little more complex. We needed to have the servlet get the field data from the HTML incoming stream. In the first test of our source code we set the values of the String instances to the values of the user ID and password, and then made a connection instance. This was all carried out in the init() method. A section of the two source code files for the Java servlet is shown next. Only the sections of the source code that were changed from the original example are shown, so these listings are not complete. The full source is available as a sample with ServletExpress 1.0.

```
/*
 * This servlet connects to the DB2 Sample database using JDBC and prints out the
 * query result set to the browser page. The servlet is invoked from an HTML page
 * form's Submit button. The HTML form is used to dynamically specify the query
 * parameters. The parameters are parsed by the servlet from the query string sent
 * from the form and used to build an SQL query that is executed on the database
 * via the JDBC driver. The servlet also opens two connections to the database that
 * it keeps open and cycles use for service method calls.
 */

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.util.*;
import java.text.*;
import com.sun.server.util.*;

/**
 * This is an example of a JDBC Database Servlet that dynamically builds an SQL
 * query, executes it on the database, then prints out the result set returned.
 */

public class JDBCServlet extends HttpServlet
{

   // ResourceBundle class constant
   public final String BASENAME = "samples";

   // con is initialized at startup of servlet
   static String url=null;
   static String userid=null;
   static String password=null;
   static int [] Constat = {0,0};
   static Connection con0;
   static Connection con1;

   static
   {
      try
      {
         // register the driver with DriverManager
         Class.forName("COM.ibm.db2.jdbc.app.DB2Driver");

      }
      catch (ClassNotFoundException e)
      {
         e.printStackTrace();
      }
   }
```
Figure 353 (Part 1 of 2). Source Code for the First Servlet Test

```
public void init(ServletConfig sc)
{
   try
   {
      super.init(sc);
   }
   catch (ServletException e)
   {
      e.printStackTrace();
   }

   // URL is jdbc:db2:dbname  1
   url = "jdbc:db2:SAMPLE";  2
   userid = "db2inst1";  3
   password = "ibmdb2";  5


   try  6
   {
      // connect with default id/password  7
      con0 = DriverManager.getConnection(url, userid, password);  8
      con1 = DriverManager.getConnection(url, userid, password);  9
   }  10

   catch(Exception e)  11
   {  12
      e.printStackTrace();  13
   }  14

} // init

public void service(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
{
   ResourceBundle rb = null;
   MessageFormat mf = null;
   ServletOutputStream out = null;
   String qry, qs, qryStr, state;
   String [] vars = {""};
   String tbl = null, col = null;
   String colStr = null;
   Object colVal = null;
   String [] qArgs = new String [6];
   PrintWriter pw = null;

   res.setContentType("text/html");
   out = res.getOutputStream();
   pw = new PrintWriter(out);
```

Figure 353 (Part 2 of 2). Source Code for the First Servlet Test

In the changes made to accept the user ID and password from the HTML form, the block of code to set the value of the String instances and create the connection was moved to the service() method to be able to make use of the getParameter() method for the HttpServletRequest req object. The lines of code that were moved from the init() to the service() method are marked with highlighted numbers from

1 to **14**. Look for the changes in the way that the user ID and password String instances were handled, and for the way they were moved from the init() method to the service() method. Again, only a section of the source code in which changes were made follows.

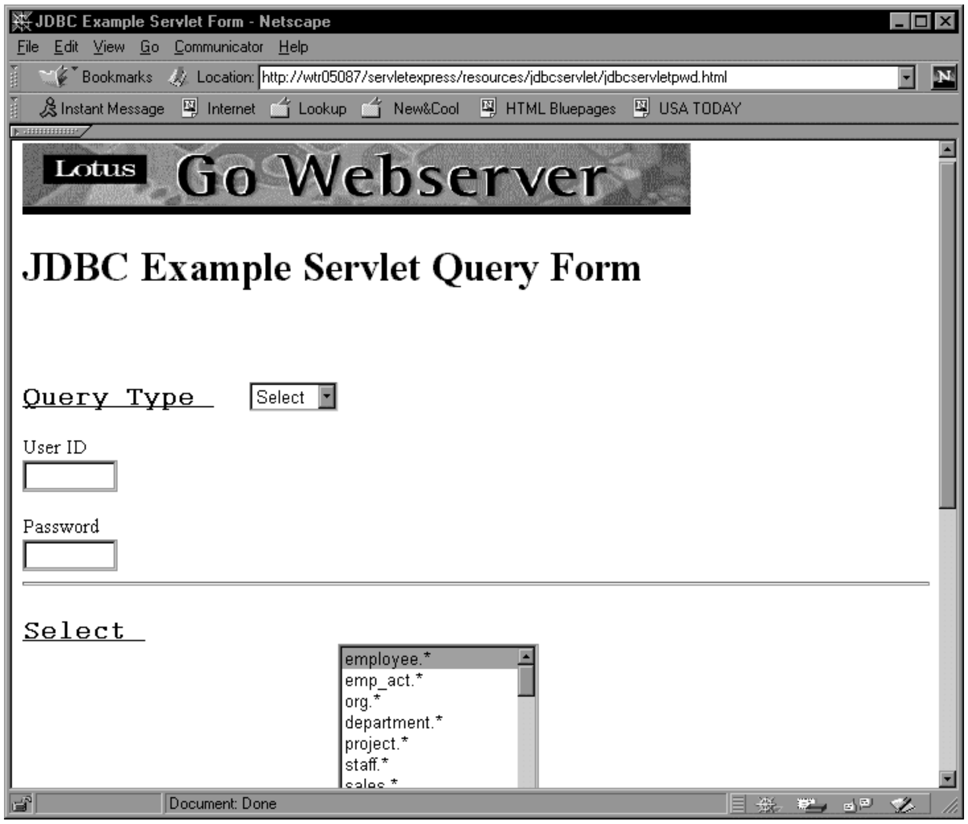
```
/*
 * This servlet connects to the DB2 Sample database using JDBC and prints out the
 * query result set to the browser page. The servlet is invoked from an HTML page
 * form's Submit button. The HTML form is used to dynamically specify the query
 * parameters. The parameters are parsed by the servlet from the query string sent
 * from the form and used to build an SQL query that is executed on the database
 * via the JDBC driver. The servlet also opens two connections to the database that
 * it keeps open and cycles use for service method calls.
 */

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.util.*;
import java.text.*;
import com.sun.server.util.*;

/**
 * This is an example of a JDBC Database Servlet that dynamically builds an SQL
 * query, executes it on the database, then prints out the result set returned.
 */

public class JDBCServlet extends HttpServlet
{

   // ResourceBundle class constant
   public final String BASENAME = "samples";

   // con is initialized at startup of servlet
   static String url=null;
   static String userid=null;
   static String password=null;
   static int [] Constat = {0,0};
   static Connection con0;
   static Connection con1;

   static
   {
      try
      {
         // register the driver with DriverManager
         Class.forName("COM.ibm.db2.jdbc.app.DB2Driver");

      }
      catch (ClassNotFoundException e)
      {
         e.printStackTrace();
      }
   }
```

Figure 354 (Part 1 of 2). Source Code for the Second Servlet Test

```
public void init(ServletConfig sc)
{
   try
   {
      super.init(sc);
   }
   catch (ServletException e)
   {
      e.printStackTrace();
   }

} // init

public void service(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
{
   ResourceBundle rb = null;
   MessageFormat mf = null;
   ServletOutputStream out = null;
   String qry, qs, qryStr, state;
   String [] vars = {""};
   String tbl = null, col = null;
   String colStr = null;
   Object colVal = null;
   String [] qArgs = new String [6];
   PrintWriter pw = null;
   res.setContentType("text/html");
   out = res.getOutputStream();
   pw = new PrintWriter(out);

   // URL is jdbc:db2:dbname  1
   url = "jdbc:db2:SAMPLE";  2
   userid = req.getParameter("userid");  3
   password = req.getParameter("passwd");  4

   try  5
   {  6
      // connect with default id/password  7
      con0 = DriverManager.getConnection(url, userid, password);  8
      con1 = DriverManager.getConnection(url, userid, password);  9
   }  10
   catch(Exception e)  11
   {  12
      e.printStackTrace();  13
   }  14
```

Figure 354 (Part 2 of 2). Source Code for the Second Servlet Test

The result of these changes was the addition of user ID and password fields to the form and the possibility for the servlet to pick up the values entered. This allowed us to set security on the database and implement that security in our application.


JDBC Example Servlet Form - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://wtr05087/servletexpress/resources/jdbcservlet/jdbcservletpwd.html
Instant Message Internet Lookup New&Cool HTML Bluepages USA TODAY
Lotus Go Webserver
Document: Done

JDBC Example Servlet Query Form

Query Type Select

User ID

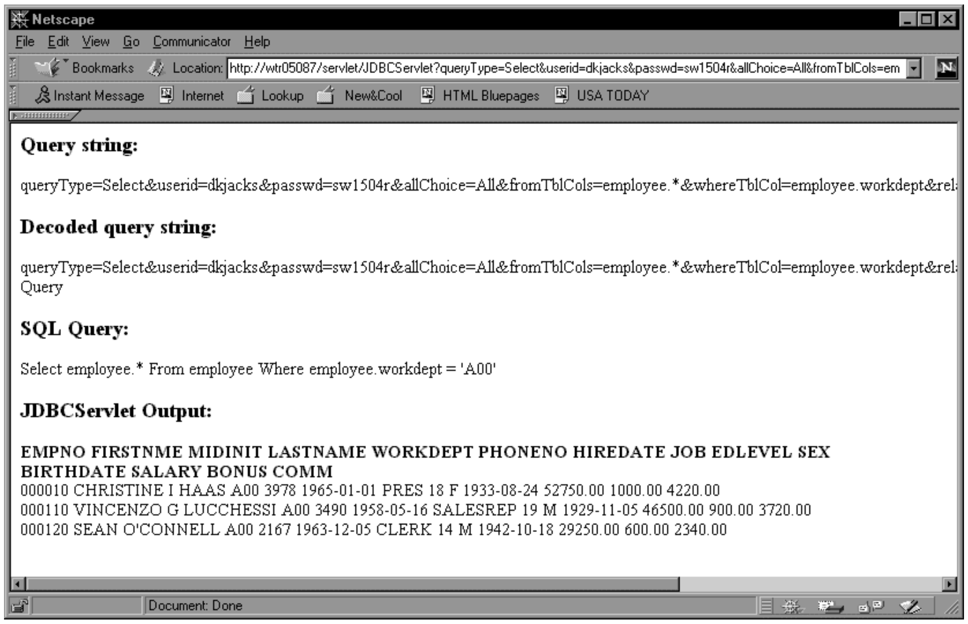
Password

Select

employee.*
emp_act.*
org.*
department.*
project.*
staff.*
sales.*

Figure 355. JDBCServlet Form with User ID and Password

Now, if you look closely at the following picture you will see the user ID and password for the database imbedded in the URL string.


Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://wtr05087/servlet/JDBCServlet?queryType=Select&userid=dkjacks&passwd=sw1504r&allChoice=All&fromTblCols=em
Instant Message Internet Lookup New&Cool HTML Bluepages USA TODAY
Document: Done

Query string:

```
queryType=Select&userid=dkjacks&passwd=sw1504r&allChoice=All&fromTblCols=employee.*&whereTblCol=employee.workdept&rel
```

Decoded query string:

```
queryType=Select&userid=dkjacks&passwd=sw1504r&allChoice=All&fromTblCols=employee.*&whereTblCol=employee.workdept&rel
Query
```

SQL Query:

```
Select employee.* From employee Where employee.workdept = 'A00'
```

JDBCServlet Output:

```
EMPNO FIRSTNME MIDINIT LASTNAME WORKDEPT PHONENO HIREDATE JOB EDLEVEL SEX
BIRTHDATE SALARY BONUS COMM
000010 CHRISTINE I HAAS A00 3978 1965-01-01 PRES 18 F 1933-08-24 52750.00 1000.00 4220.00
000110 VINCENZO G LUCCHESSI A00 3490 1958-05-16 SALESREP 19 M 1929-11-05 46500.00 900.00 3720.00
000120 SEAN O'CONNELL A00 2167 1963-12-05 CLERK 14 M 1942-10-18 29250.00 600.00 2340.00
```

Figure 356. Result of JDBCServlet Form Using GET Method

The form created for the Web page and the associated servlet use the HTTP GET method for sending data from the Web page to the servlet. The form calls the servlet and passes the data arguments on the same URL in a data element named QUERY_STRING. This presents two or three problems.

1. The URL may get truncated, losing data that is necessary for the application to operate properly.
2. If someone is near the user, the data in the URL may be seen and written down, or recovered in some other fashion.
3. If there is a Man-in-the-Middle (MIM), the data in the URL may be intercepted, and in this case a user ID and password is exposed, unless encryption is used.

This application design violates the integrity of our environment. How can we change the application so that the security is intact? The use of the POST method would solve part of these problems by sending the data stream to the servlet on a separate data stream which is not appended to the URL. Information entered in a form will still be visible by a MIM, but this problem would be solved by using SSL. You can refer to 3.2, “Examples of Security Using HTTP and SSL” on page 94 for the security implications of using GET and POST over HTTP or SSL.

Rather than changing the Java source code to permit our servlet to handle the POST method, we thought that the best solution would be to hard code a user ID and password in the servlet (so the user ID and password are no longer passed through the URL) and set security on the Web server, utilizing the security features provided by ServletExpress.

ServletExpress offers very powerful security measures, so it is not a bad idea to set security on it and hard code user ID and password. By the way, at the end of this project, the next Release 1.0 beta 3 became available for ServletExpress and we noticed that the source code of the JDBCServlet in the new release had been changed in the same way we had changed it, so that now, before compiling the Java file, you are forced to open and modify it by hard coding the user ID and password.

Of course this choice has security implications, since it means that now the class file contains the user ID and password and, if the bytecode is decompiled, this information could be retrieved. However Lotus Domino Go Webserver protects the class file of a servlet, so it cannot be retrieved remotely.

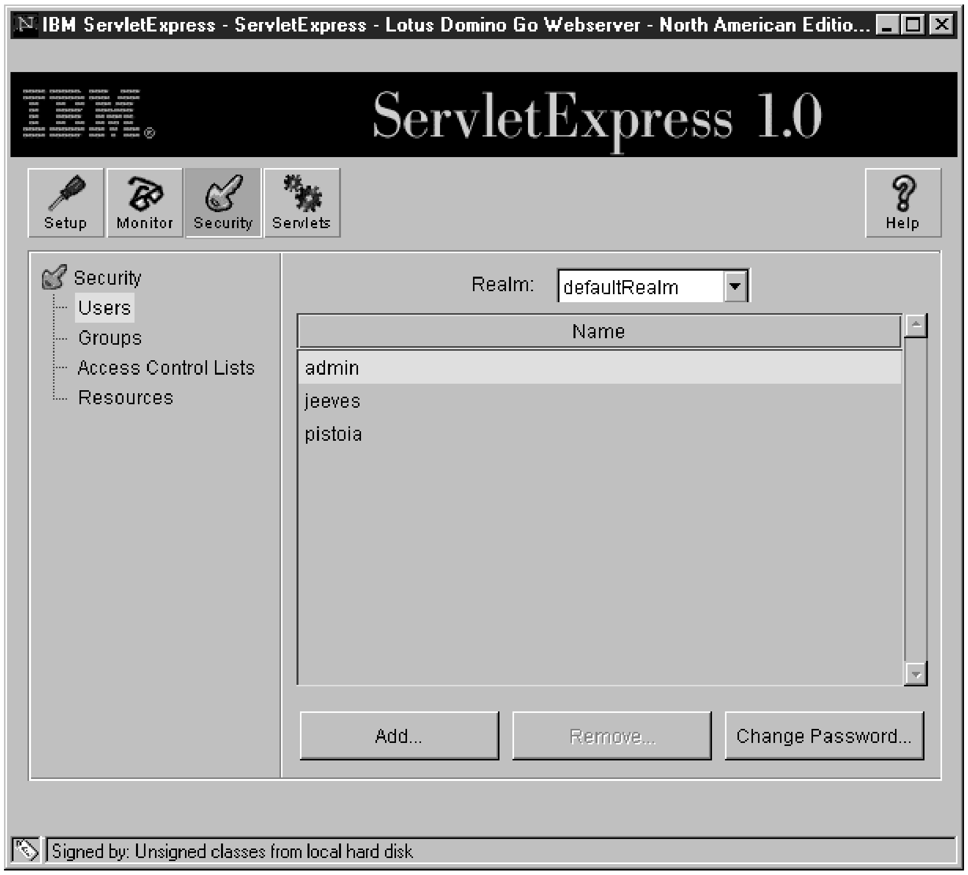
Alternatively user ID and password could be specified as initialization parameters to be read in by the servlet when the servlet is loaded. The initialization parameters of a servlet can be read in the servlet's init() method using the two methods getInitParameterNames() and getInitParameters(). This technique is described in the IBM redbook *Network Computing Framework Component Guide*, SG24-2119, Chapter 5, "Servlets" on page 157. User ID and password, retrieved as initialization parameters, can then be used to create a connection specification, allowing the servlet to be more generic. This way, user ID and password are the same for any access, so it is not necessary to pass this information over the Internet, and access control can be set using ServletExpress. Nonetheless user ID and password are not hard coded and the risk of a hacker retrieving such private information by decompiling the servlet class file is eliminated.

8.3.1 Setting Security with ServletExpress

ServletExpress considers servlets as resources and offers the possibility to protect access to them. We want to show you what we did to protect access to the JDBCServlet. This is very important since we have hard coded a user ID and password in the JDBCServlet. Whoever accesses the JDBCServlet, also accesses the database. In order to grant security, we must limit access to the JDBCServlet so that only authorized users can retrieve sensitive information. For further details about ServletExpress security, refer to 4.1, “ServletExpress Security Management” on page 122 and 4.2, “ServletExpress Advanced Security” on page 159.

In order to protect the JDBCServlet as a ServletExpress resource, first of all JDBCServlet must be one of the servlets managed by ServletExpress. In other words, it must be added to the list of all the servlets managed by ServletExpress. This is what we did in 8.2.2, “Servlet Registration in ServletExpress” on page 366.

The second step is to decide which users can have access to that servlet. Let's say that we want to define a new user, named marco, and grant marco access to the JDBCServlet. For this reason, we opened the ServletExpress Manager, pressed the **Security** button and then selected **Users**.



Realm: defaultRealm

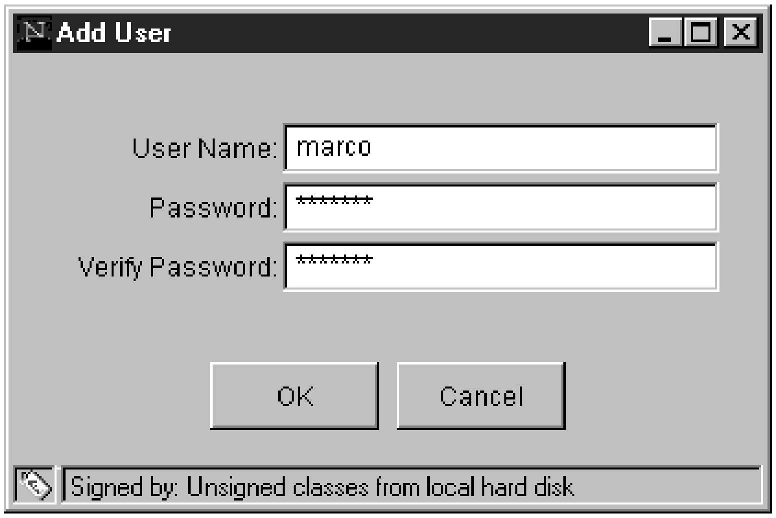
Name
admin
jeeves
pistoia

Signed by: Unsigned classes from local hard disk

Figure 357. Users Page for the ServletExpress Manager

Notice that we were going to add the new user under the defaultRealm. It didn't make sense, in this case, to add a user under the servletMgrRealm, since users in the servletMgrRealm are considered servlet-signers. The NT realm also was not good for our purposes, since it requires that each user is also a user for the underlying operating system, so adding a new user would require defining a new user for the operating system itself. The defaultRealm is the best choice when you want to control access to a servlet.

Clicking the **Add:::** button we then accessed the page to add a new user.

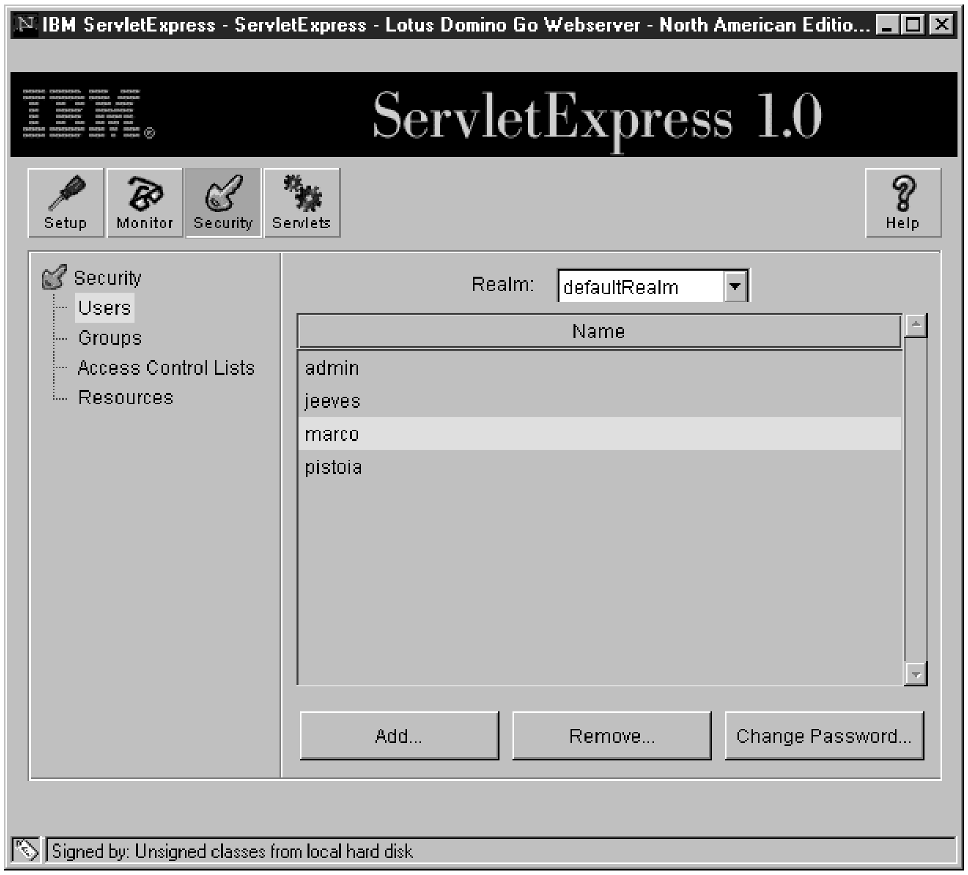


User Name: marco
Password: ********
Verify Password: ********

Signed by: Unsigned classes from local hard disk

Figure 358. Adding the New User Named marco

When we clicked **OK**, the new user was visible among all the other users defined under the defaultRealm.



Realm: defaultRealm

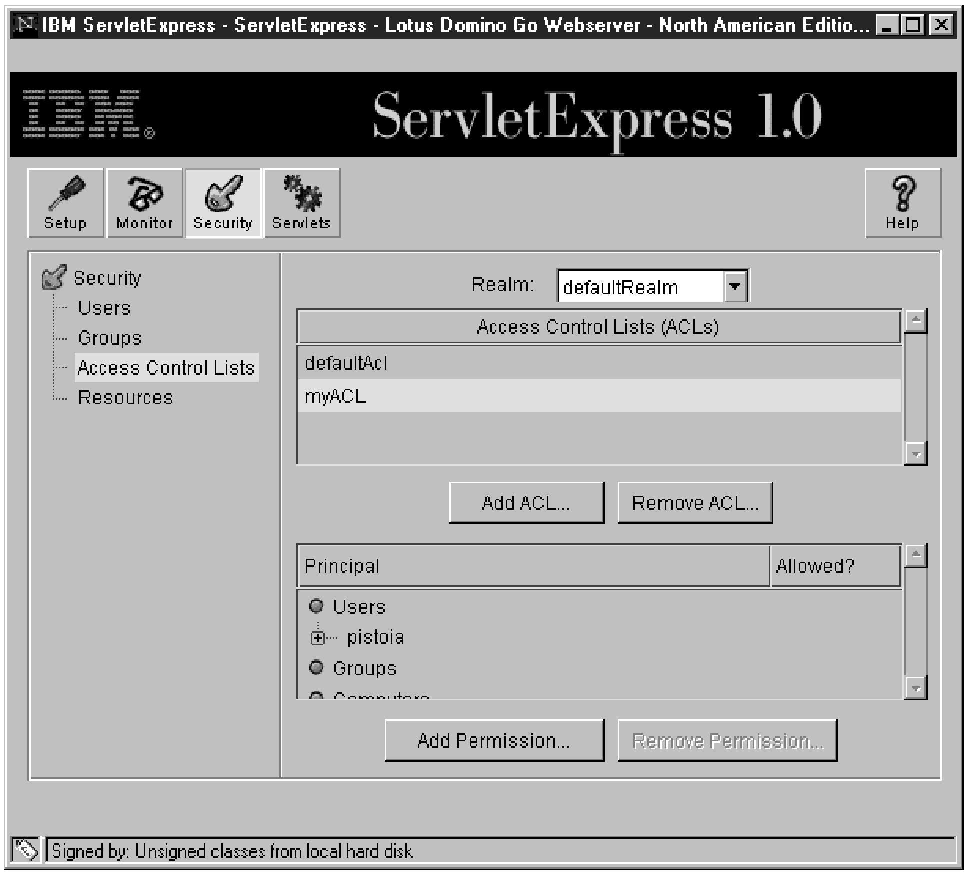
Name
admin
jeeves
marco
pistoia

Signed by: Unsigned classes from local hard disk

Figure 359. The New User Has Been Added to the defaultRealm

Of course it is also possible to make the new user a member of a group and then grant the whole group access to the JDBCServlet. We didn't perform this operation, because we just wanted to test this scenario with only one user.

Selecting **Access Control Lists** from the Security tree, the list of all the defined ACLs appears.



Realm: defaultRealm

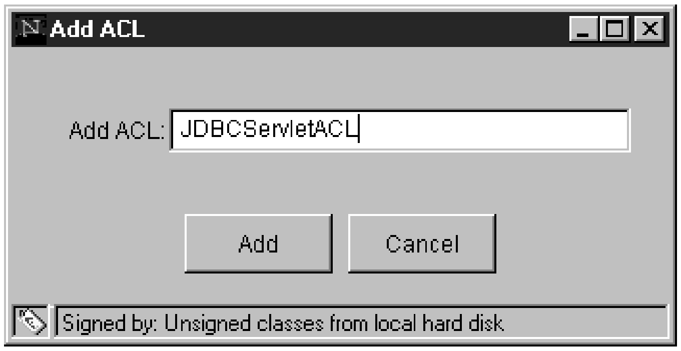
Access Control Lists (ACLs)
defaultAcl
myACL

Principal	Allowed?
Users	
pistoia	
Groups	

Signed by: Unsigned classes from local hard disk

Figure 360. ACLs List

We wanted to create a new, specific ACL, in order to control access to the JDBCServlet only. For this reason, we clicked **Add ACL:::** and in the new dialog box that appeared we entered the name of the new ACL we were going to define. We named it JDBCServletACL.

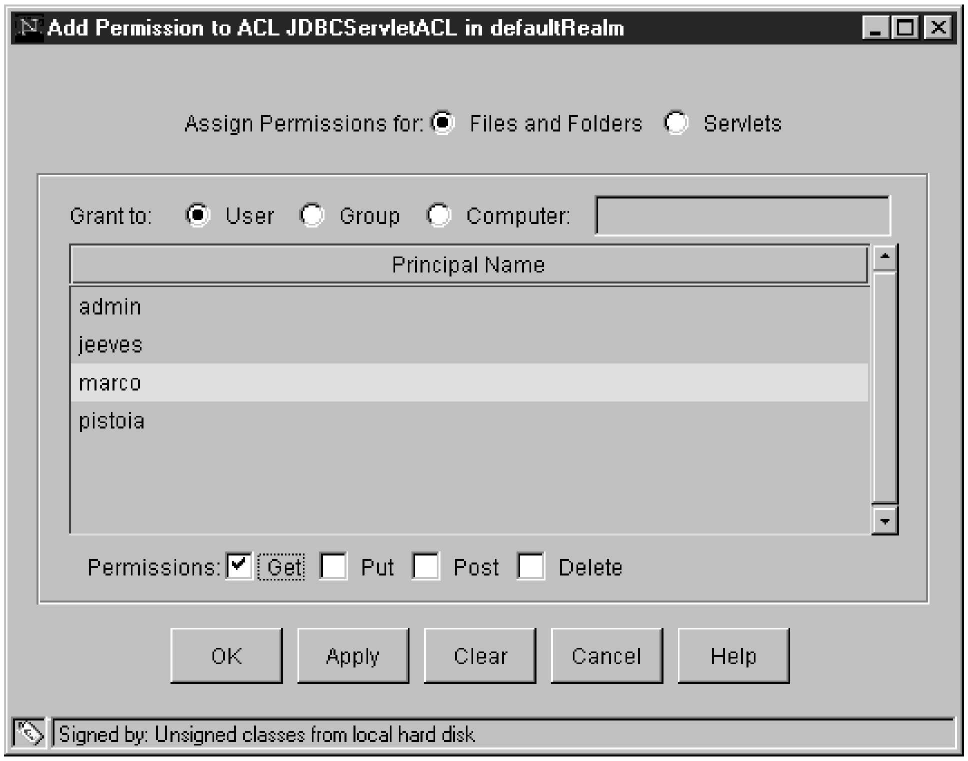


Add ACL: JDBCServletACL

Signed by: Unsigned classes from local hard disk

Figure 361. Adding the New Access Control List JDBCServletACL

Clicking **Add** we got the JDBCServletACL as a new entry in the ACLs list. We then pressed **Add Permission:::** and then we were able to grant the new user marco permission to GET.


Add Permission to ACL JDBCServletACL in defaultRealm

Assign Permissions for: [✓ Files and Folders] Servlets

Grant to: [✓ User] Group Computer:

Principal Name
admin
jeeves
marco
pistoia

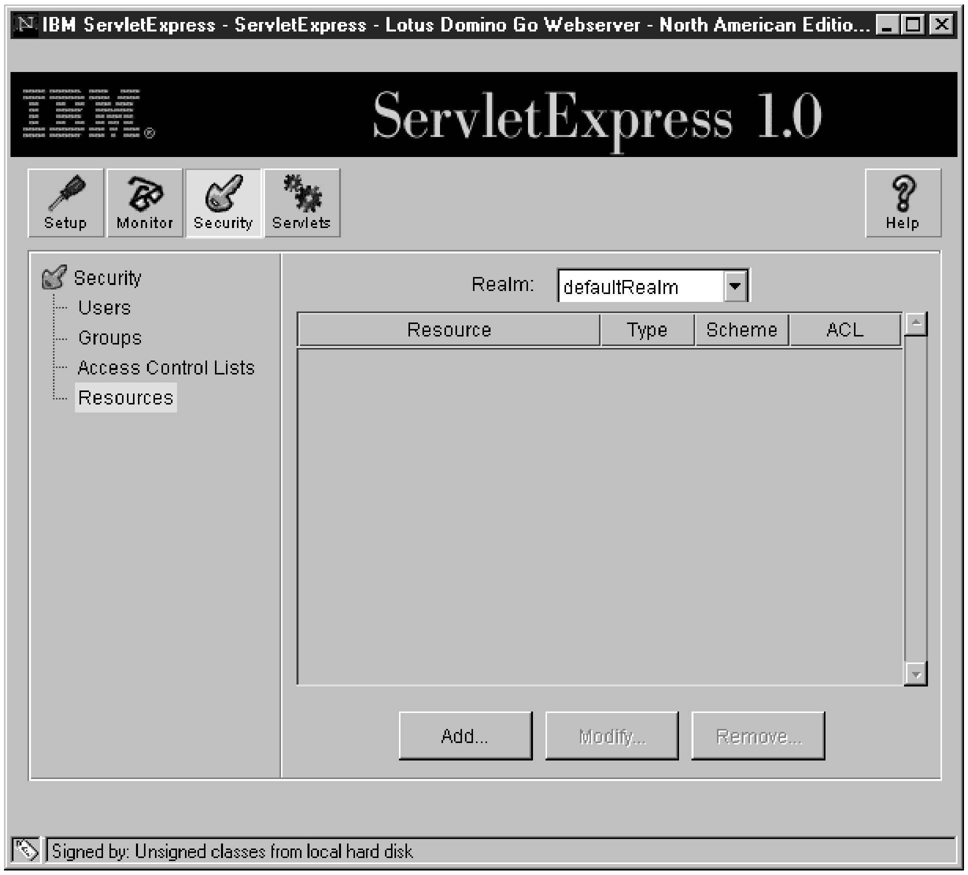
Permissions: [✓ Get] Put Post Delete

Signed by: Unsigned classes from local hard disk

Figure 362. Granting the New User marco Permission to GET

The user marco only needs permission to GET in order to run the JDBCServlet from the JDBCServletForm, since that form uses the GET method to invoke the JDBCServlet.

We then selected the **Resources** item from the Security tree.



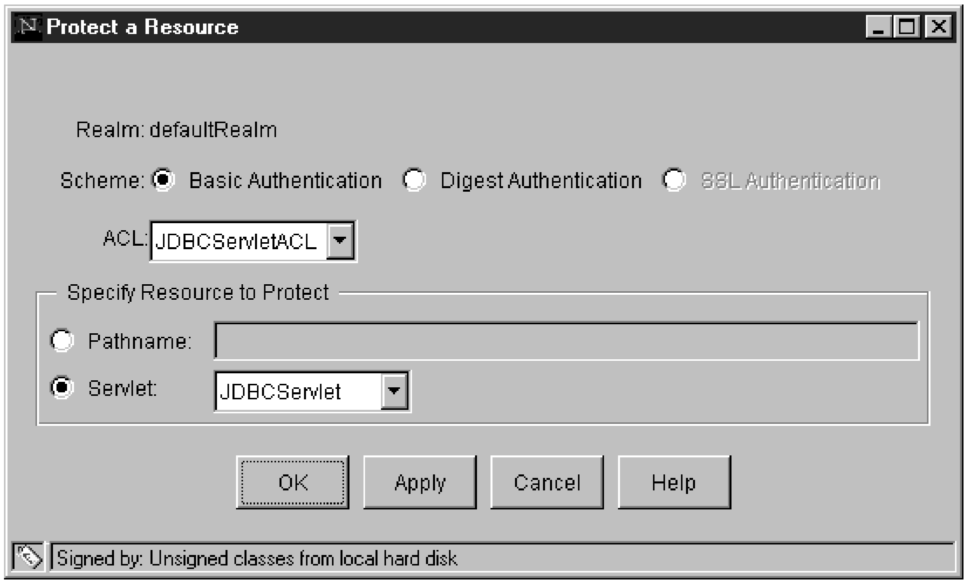
Realm: defaultRealm

Resource	Type	Scheme	ACL

Signed by: Unsigned classes from local hard disk

Figure 363. The Resources Page from the ServletExpress Manager

We clicked **Add:::** to add the JDBCServlet as a new resource in the JDBCServletACL, as shown in the following figure:



Realm: defaultRealm

Scheme: [✓ Basic Authentication] Digest Authentication SSL Authentication

ACL: JDBCServletACL

Specify Resource to Protect

Pathname:

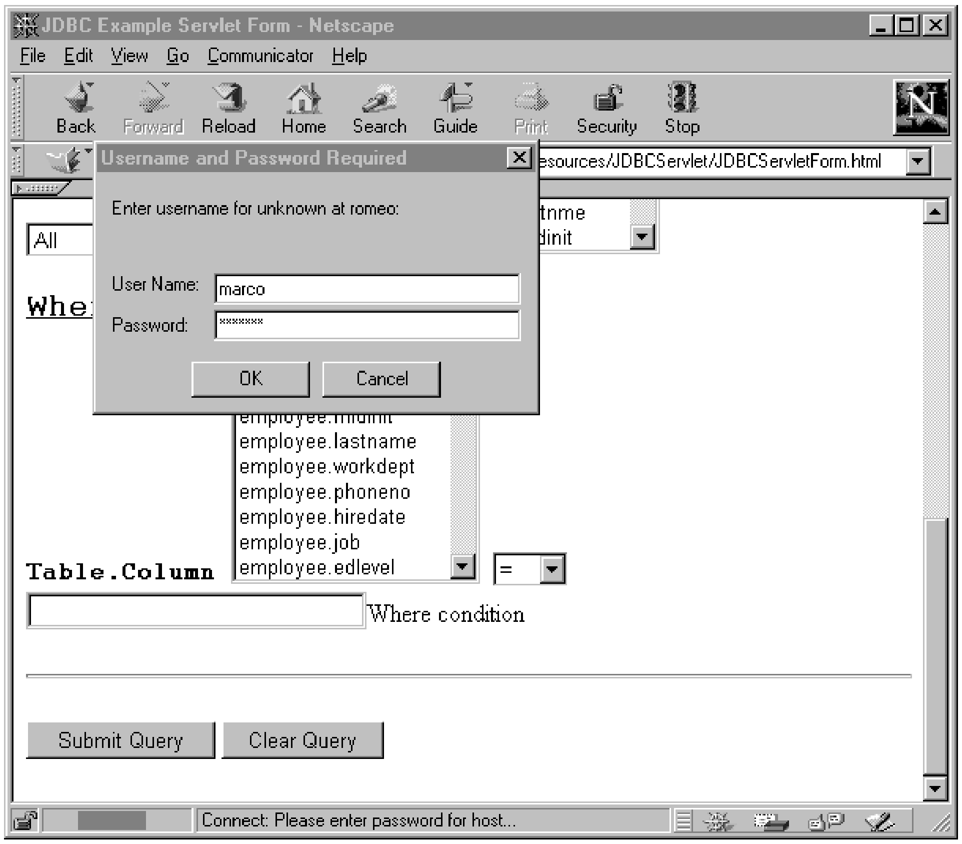
[✓ Servlet:] JDBCServlet

Signed by: Unsigned classes from local hard disk

Figure 364. Adding the JDBCServlet as a New Resource in the JDBCServletACL

These steps were enough to place security on the Web server machine, using the security measures offered by ServletExpress. From now on, the JDBCServlet will be considered a protected resource by ServletExpress. Access to the JDBCServlet will be limited to the user marco, who has permission only to GET. This user will only be allowed to invoke the JDBCServlet through the GET method, that is the method used by the JDBCServletForm.html Web page to call the JDBCServlet.

Each time any users on the client machine try to access the JDBCServlet, they will be prompted with a dialog box, where they will have to enter the defined user name and password:



Username and Password Required

Enter username for unknown at romeo:

User Name: marco

Password: *******

Connect: Please enter password for host...

Figure 365. Entering User ID and Password to Access the JDBCServlet

You can see that the lock icon in the bottom-left corner of the window is open. This means that a secure SSL communication is not in place yet. In fact this initial test was performed without adding SSL security. Later in this chapter, you will see how we will enforce the security measures step-by-step in order to run a secure application.

8.3.2 DB2 Client/Server Communication Security

Before setting security with the firewall, we wanted to make another experiment to see how secure the communication between the DB2 client and the DB2 server was.

For this reason, we acted as a MIM and we sniffed the network while the DB2 client was connecting a database on the DB2 server. As we mentioned in 3.2, “Examples of Security Using HTTP and SSL” on page 94, this also can be considered a MIM attack, even if in this case the MIM is simply copying frames in transit, trying to get sensitive information. In a more severe MIM attack, to the server the MIM masquerades as the client and to the client the MIM masquerades as the server, in order to compromise the data flowing between them.

These are the particulars of the test environment:

Table 6. Environment Configuration

	DB2 Client	**DB2 Server**
Host Name	romeo	wtr05218
IP Address	9.24.104.176	9.24.104.210

And these are the DB2 settings:

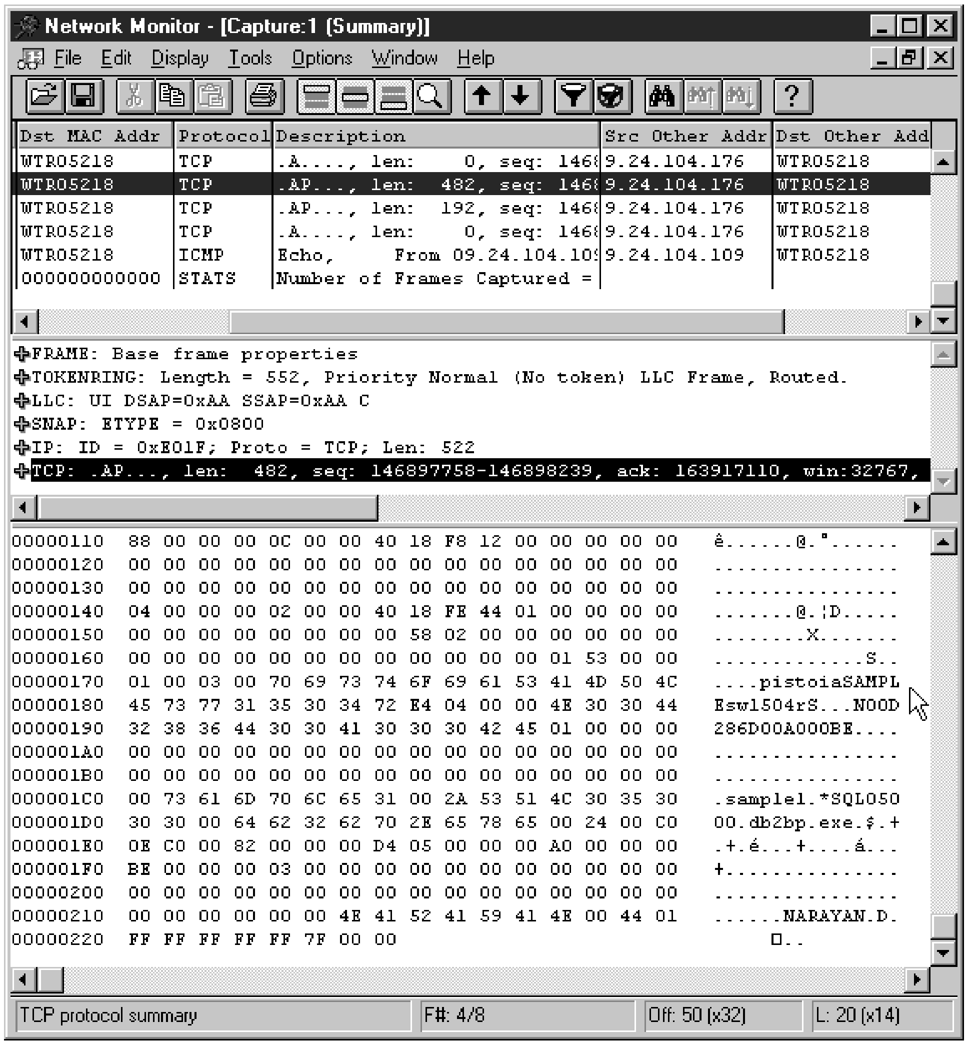
Table 7. DB2 Settings

Database Name on the DB2 Server	SAMPLE
Database Alias on the DB2 Client	SAMPLE1
User ID	pistoia
Password	sw1504r

As soon as the user on the DB2 client connects the database on the server by entering the command:

```
connect to sample1 user pistoia using sw1504r
```

on a DB2 Command Line Processor, this is what a MIM can see:


Network Monitor - [Capture:1 (Summary)]
File Edit Display Tools Options Window Help

```
Dst MAC Addr  Protocol Description                   Src Other Addr Dst Other Add
WTR05218      TCP      .A...., len:    0, seq: 146{  9.24.104.176   WTR05218
WTR05218      TCP      .AP..., len:  482, seq: 146{  9.24.104.176   WTR05218
WTR05218      TCP      .AP..., len:  192, seq: 146{  9.24.104.176   WTR05218
WTR05218      TCP      .A...., len:    0, seq: 146{  9.24.104.176   WTR05218
WTR05218      ICMP     Echo,     From 09.24.104.10{  9.24.104.109   WTR05218
000000000000  STATS    Number of Frames Captured =

+FRAME: Base frame properties
+TOKENRING: Length = 552, Priority Normal (No token) LLC Frame, Routed.
+LLC: UI DSAP=0xAA SSAP=0xAA C
+SNAP: ETYPE = 0x0800
+IP: ID = 0xE01F; Proto = TCP; Len: 522
+TCP: .AP..., len:  482, seq: 146897758-146898239, ack: 163917110, win:32767,

00000110  88 00 00 00 0C 00 00 40 18 F8 12 00 00 00 00 00   ê......@.°......
00000120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000140  04 00 00 00 02 00 00 40 18 FE 44 01 00 00 00 00   .......@.¦D.....
00000150  00 00 00 00 00 00 00 00 58 02 00 00 00 00 00 00   ........X.......
00000160  00 00 00 00 00 00 00 00 00 00 00 00 01 53 00 00   .............S..
00000170  01 00 03 00 70 69 73 74 6F 69 61 53 41 4D 50 4C   ....pistoiaSAMPL
00000180  45 73 77 31 35 30 34 72 E4 04 00 00 4E 30 30 44   Esw1504rS...N00D
00000190  32 38 36 44 30 30 41 30 30 30 42 45 01 00 00 00   286D00A000BE....
000001A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001C0  00 73 61 6D 70 6C 65 31 00 2A 53 51 4C 30 35 30   .sample1.*SQL050
000001D0  30 30 00 64 62 32 62 70 2E 65 78 65 00 24 00 C0   00.db2bp.exe.$.+
000001E0  0E C0 00 82 00 00 00 D4 05 00 00 00 A0 00 00 00   .+.é...+....á...
000001F0  BE 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00   +...............
00000200  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000210  00 00 00 00 00 00 4E 41 52 41 59 41 4E 00 44 01   ......NARAYAN.D.
00000220  FF FF FF FF FF 7F 00 00                                 □..

TCP protocol summary    F#: 4/8    Off: 50 (x32)    L: 20 (x14)
```

Figure 366. Sniffing the DB2 Client/Server Communication

We used the Microsoft Systems Management Server Network Monitor Version 4.00 to capture these frames. This utility was installed on another Windows NT Server system located in the same intranet as the DB2 client and server.

Look at what the mouse arrow is pointing to: you will notice that user ID, database name on the DB2 server, password and database alias on the client are transmitted in clear text.

From a security point of view, this could be a problem. User ID and password are clearly visible, either hard coded in the servlet or dynamically entered by the user on the client machine. We solved this problem by designing an appropriate architecture for our environment, where the Web server machine, which is also the DB2 client, is placed in the DMZ and the DB2 server is located in the secure network. The firewall will prevent intruders from retrieving private information when this is transmitted between the DB2 client and the DB2 server.

Another solution could have been to use Distributed Computing Environment (DCE) Security Services to authenticate users, since DCE provides:

- Centralized administration of users and passwords

- No transmission of clear text passwords and user IDs
- A single sign-on for users

For more information on how to use DCE Security Services to authenticate DB2 users, refer to *IBM DB2 Universal Database Administration Guide Version 5*.

8.4 Scenario Environment Configuration

The flow of the application that we described in 8.1, "Overview" on page 361 involved three machines: client, server and application server. The protocol connection between these machines was controlled by a firewall, so our environment was composed of four machines, as shown in the following figure:

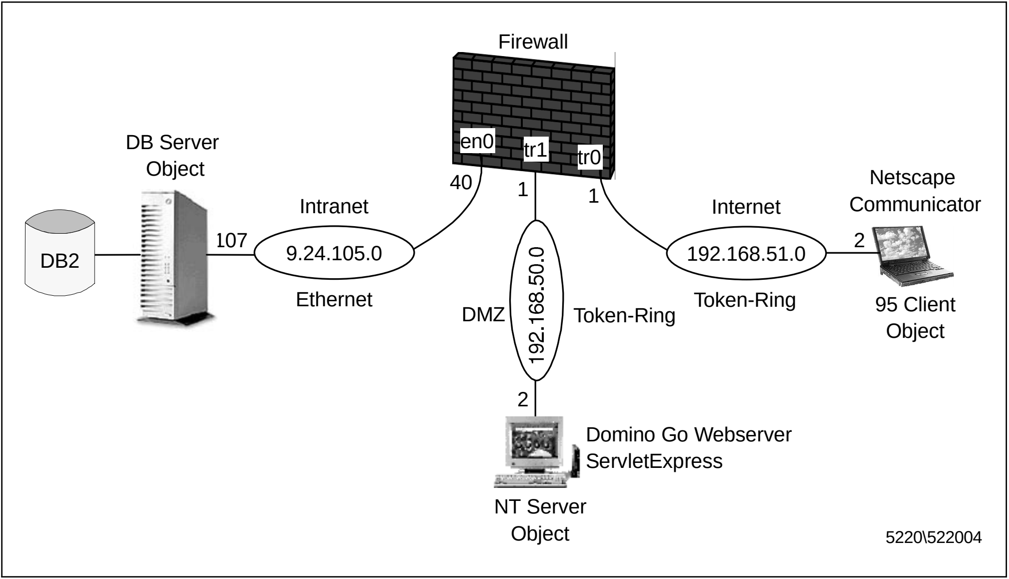


Figure 367. Scenario Environment Configuration

This section describes the particulars of hardware, software and network configuration for the scenario environment.

8.4.1 Hardware Configuration

Here we briefly state the hardware configuration of our test environment through the following table:

Table 8. Hardware Configuration

	Machine Model	**Memory**	**Network Adapters**
Client	IBM ThinkPad 560E	48 MB	Token-ring
Web Server	IBM PC 365	64 MB	Token-ring
Application Server	IBM RS/6000 43P	192 MB	Ethernet
Firewall	IBM RS/6000 370	128 MB	• Token-ring • Token-ring • Ethernet

8.4.2 Software Configuration

The software configuration of our test environment is described in the following table:

Table 9. Software Configuration

	Operating System	Software
Client	Microsoft Windows 95	Netscape Communicator 4.04
Web Server	Microsoft Windows NT Server 4.0	• Sun JDK 1.1.6 • Lotus Domino Go Webserver 4.6.2.2 • IBM ServletExpress 1.0 beta 2.1 • IBM DB2 CAE 5.0
Application Server	IBM AIX 4.3	IBM DB2 UDB 5.0
Firewall	1. IBM AIX 4.2.0 2. IBM AIX 4.2.1	1. IBM Firewall 3.1.1 2. IBM eNetwork Firewall 3.2

As we explained in 7.6, "IBM Firewall 3.1.1 and IBM eNetwork Firewall 3.2 for AIX" on page 350, when the project began, Version 3.1.1 was available for the IBM Firewall and we installed it on the operating system AIX 4.2.0. When Version 3.2 became available, we rebuilt the system with AIX 4.2.1, necessary to run the new version of the firewall, and we installed IBM eNetwork Firewall 3.2.

This is the reason why the previous table indicates two different versions for the AIX operating system and for the IBM Firewall installed on the firewall machine.

8.4.3 Network Configuration

The following table explains the network configuration for our environment:

Table 10. Network Configuration

	Host Name	IP Address	Network Adapter	Subnet Mask
Client		192.168.51.2	Token-ring	255.255.255.0
Web Server		192.168.50.2	Token-ring	255.255.255.0
Application Server	AIXNCF157E	9.24.105.107	Ethernet	255.255.255.0
Firewall	RS600012E	• 9.24.105.40 • 192.168.50.1 • 192.168.51.1	• Ethernet • Token-ring • Token-ring	• 255.255.255.0 • 255.255.255.0 • 255.255.255.0

As you can see in Table 10, we made the Ethernet adapter of the firewall machine the secure network interface, since we assumed that it was connected to the internal secure network. Its two token-ring adapters are non-secure network interfaces, since they connect the firewall machine to the untrusted non-secure network. The following section shows how we set the Ethernet adapter as secure.

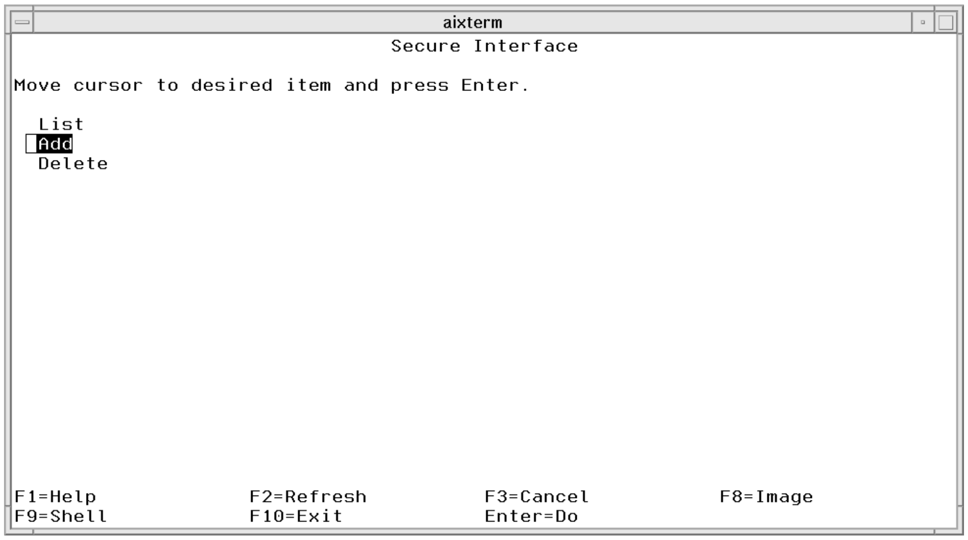
8.4.4 Setting the Secure Network Adapter on the Firewall

Once you have installed the firewall, you need to distinguish the secure network adapters from the non-secure ones. This is very important, because a secure network adapter is for communicating with an Intranet, while a non-secure network adapter is used to communicate with the Internet or with any untrusted network.

In order to identify a network interface as secure, we used the `smit` fast path:

```
smitty fw_set_secure_adapter
```

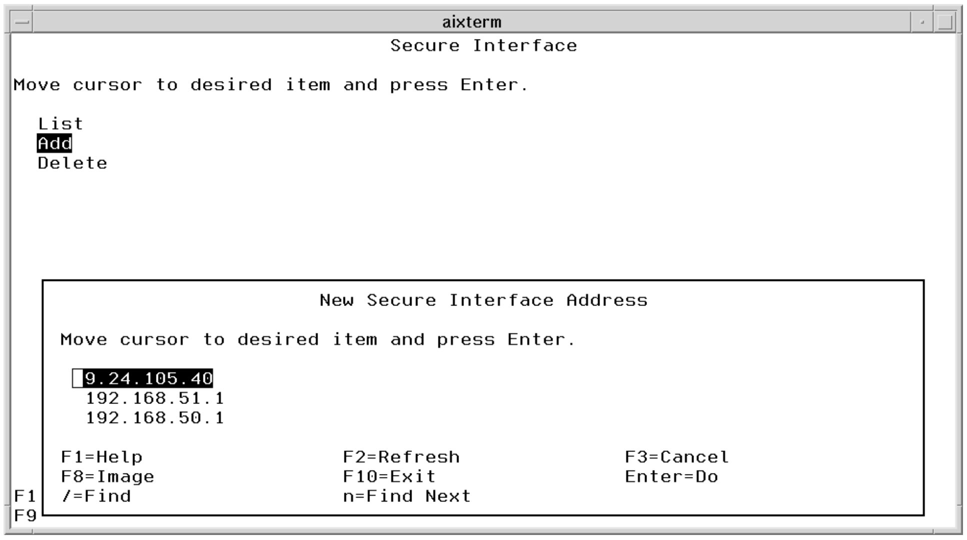
and the Secure Interface window immediately came up:



```
aixterm
                                Secure Interface

Move cursor to desired item and press Enter.

  List
  Add
  Delete

F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 368. How to Add a Secure Network Adapter

We selected **Add** and then we pressed the Enter key of the keyboard. To set an adapter as secure in your environment, you need to specify the qualified IP address corresponding to the network adapter itself. In this scenario, we used only the Ethernet adapter for the firewall as secure. That adapter had IP address 9.24.105.40 (see 8.4.3, “Network Configuration” on page 388), so we selected that value in the following window:



```
                              Secure Interface

Move cursor to desired item and press Enter.

  List
  Add
  Delete



                        New Secure Interface Address

    Move cursor to desired item and press Enter.

      9.24.105.40
      192.168.51.1
      192.168.50.1

    F1=Help                 F2=Refresh              F3=Cancel
    F8=Image                F10=Exit                Enter=Do
F1  /=Find                  n=Find Next
F9
```

Figure 369. How to Select a Secure Network Adapter

This procedure finished successfully and at the end we got a confirmation window informing us that our configuration had been accepted.

8.5 IP Filters Configuration for Three-Tier Applications

This section shows how to use the IP filter firewall technology to enable three-tier applications to pass through the firewall (see Figure 367 on page 387). Later on, we will often speak of connections, network objects, services and rules. To have a general description of each of these concepts, you can refer to 6.1.2, “Expert IP Filters Using IBM eNetwork Firewall 3.2” on page 271.

Before building the IP filters, we had to define the network protocol flow which our three-tier application would use in this scenario (see 8.1, “Overview” on page 361). The following list summarizes objects and connections that took part in our scenario environment:

1. Objects (Type: Host)

   - 95-client

     This object was a mobile IBM ThinkPad computer located in the Internet. A Web browser was installed on this machine. This object played the role of the client tier in this scenario.

   - NT-server

     This object was an IBM Personal Computer having the key role of the middle tier. In this machine we had installed Lotus Domino Go Webserver, with the servlet engine provided by IBM ServletExpress, and IBM DB2 CAE as connector. This object was located in the DMZ.

   - DB-server

     This object was the legacy system where we had installed IBM DB2 UDB as backend server tier. This machine was an IBM RISC/6000 located in the intranet.

2. Connections

   - DMZ to Secure DB2

This connection was defined to permit the DB2 CAE protocol from the NT-server object to flow to the DB-server object in order to grant the DB2 communication.

- Non-secure to DMZ HTTP and SSL

  This connection was defined to permit the HTTP and SSL protocols from 95-client object to NT-server object in order to grant secure Web communication.

See 8.4.3, “Network Configuration” on page 388 for further details.

Notice that we did not need to consider any firewall object in this scenario, because the firewall simply had to route the traffic between the three objects that we had defined, without performing any other action. Had we had to implement a firewall technology different from IP filters, like for example proxy server, then the traffic would have been broken by the firewall and three firewall objects should have been defined, one for each firewall network adapter. We will see several examples of this technique in Chapter 9, “IIOP in Firewall-Protected Network Environments” on page 435.

8.5.1 Objects Addition

We defined and configured the three objects that we have just described using the `fwconfig` program.

After entering the `fwconfig` command, this time we had the possibility to log on as the new firewall administrator user that we had created (the Logon window was shown in Figure 339 on page 353). Then we selected **Network Objects** from the Configuration Client navigation tree:

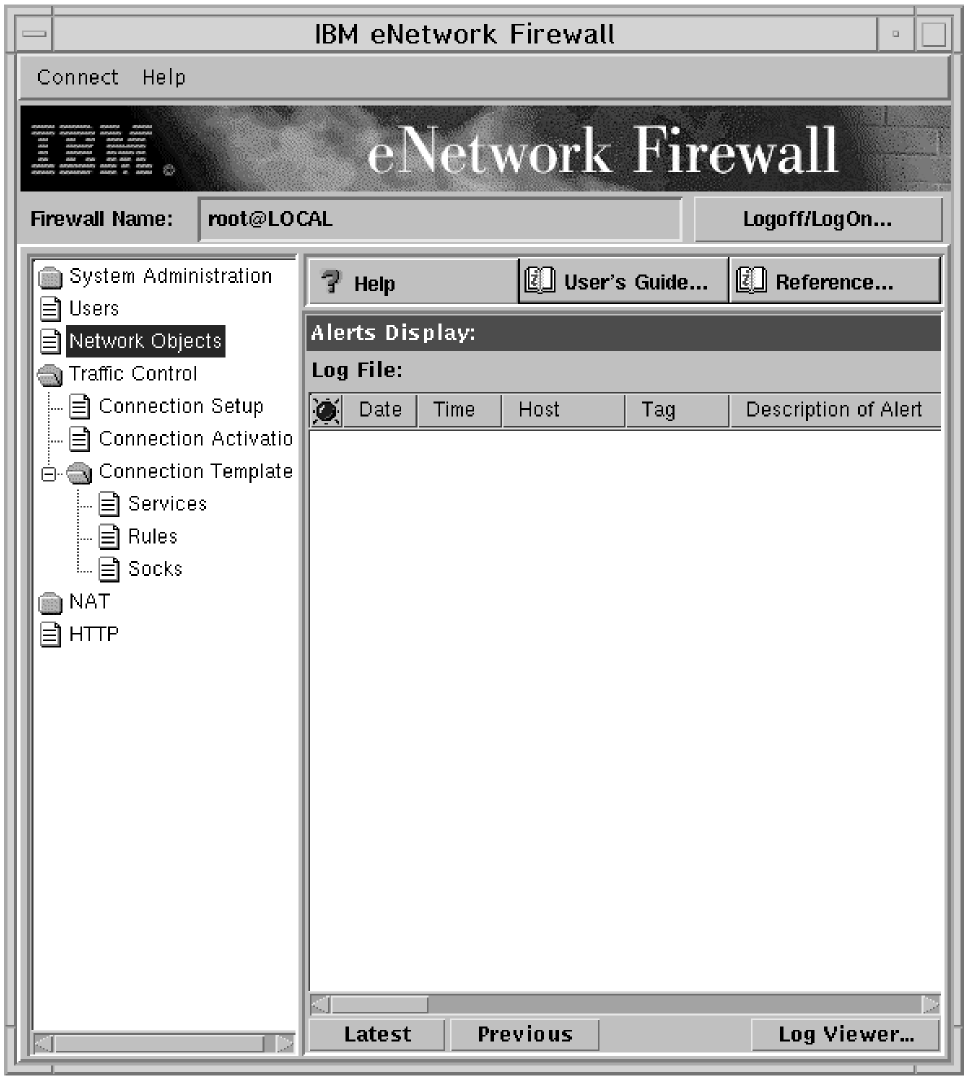
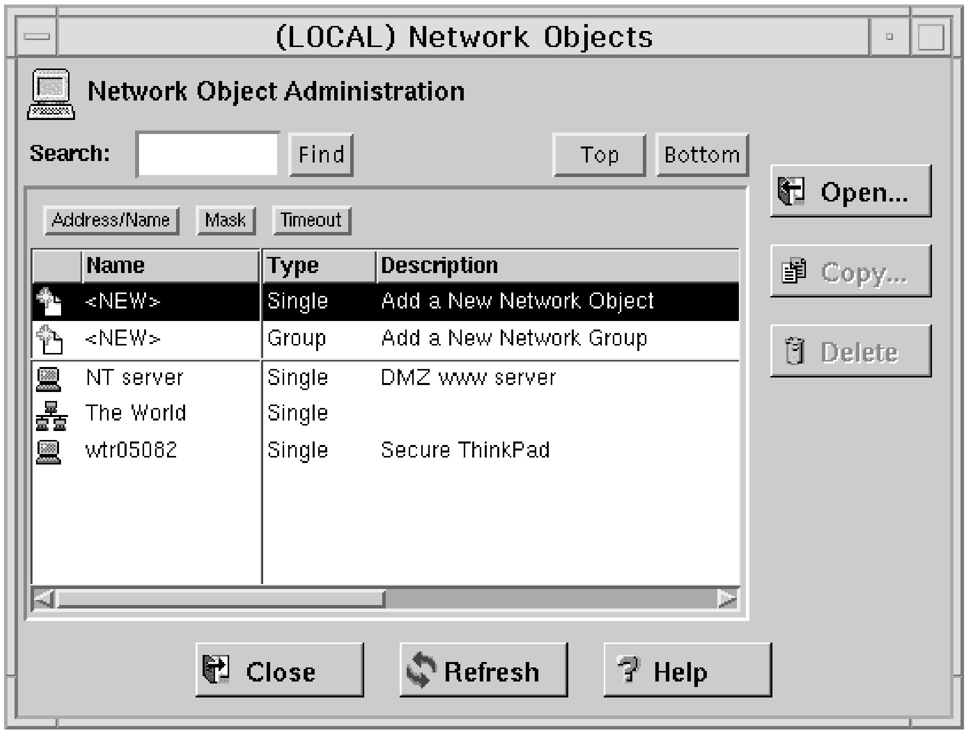


Figure 370. Network Objects in the Configuration Client Navigation Tree

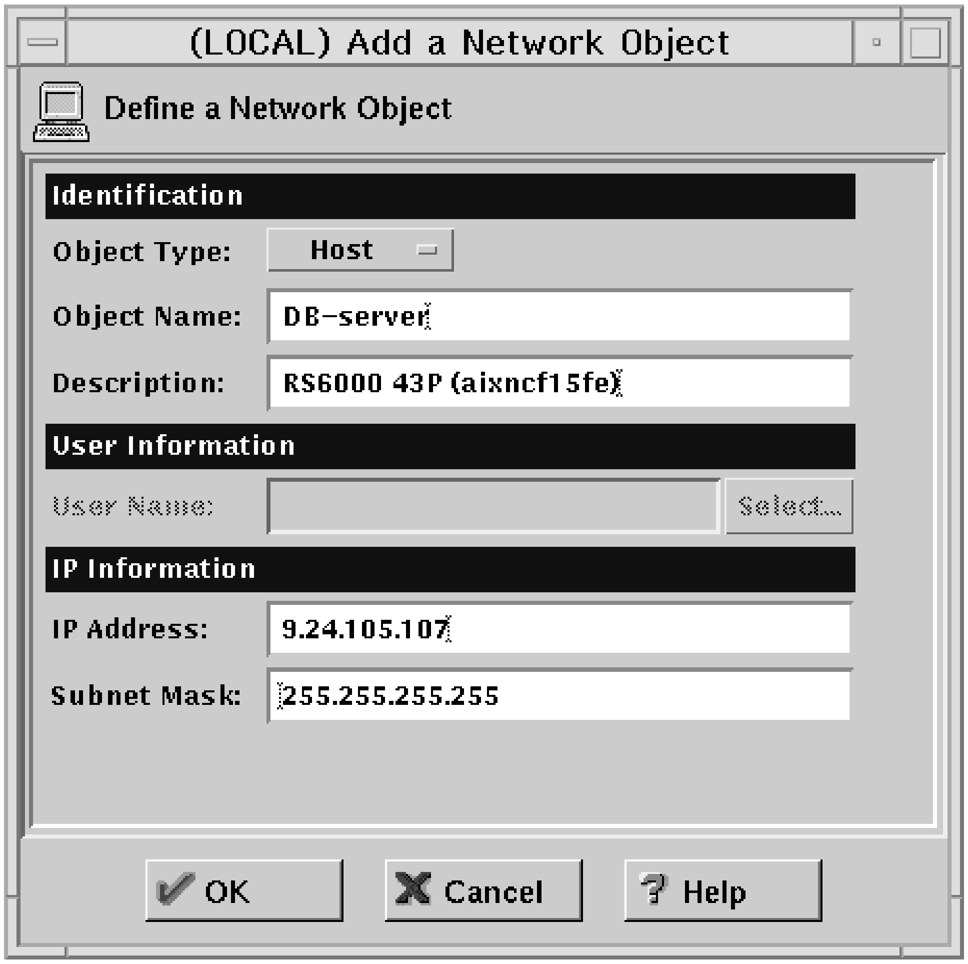
The Network Objects window was brought up. We selected **NEW**, and then we clicked the **Open:::** button:



Name	Type	Description
<NEW>	Single	Add a New Network Object
<NEW>	Group	Add a New Network Group
NT server	Single	DMZ www server
The World	Single	
wtr05082	Single	Secure ThinkPad

Figure 371. Network Objects Window

We had to define the three new objects called 95-client, NT-server and DB-server. We started by defining the DB-server object, as shown in the following screen:


(LOCAL) Add a Network Object

Define a Network Object

Identification

Object Type: Host

Object Name: DB-server

Description: RS6000 43P (aixncf15fe)

User Information

User Name:

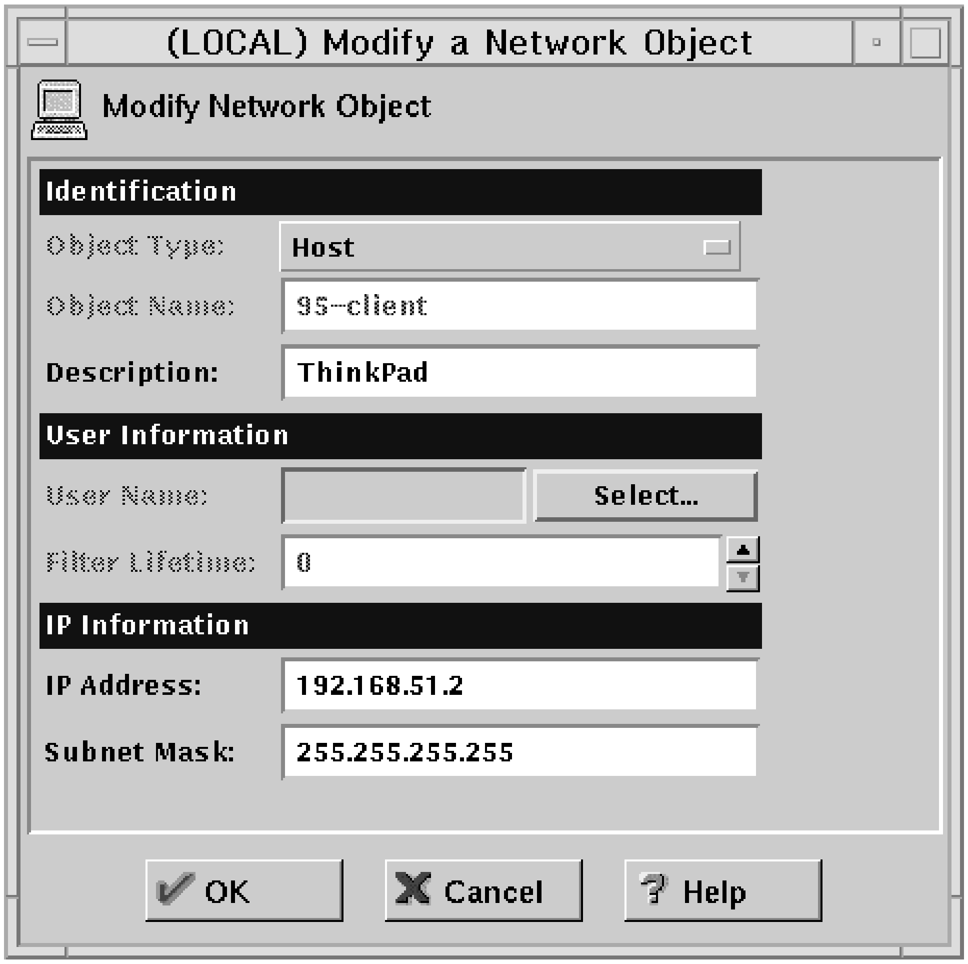
IP Information

IP Address: 9.24.105.107

Subnet Mask: 255.255.255.255

Figure 372. DB-server Object Definition

The other two objects were defined through very similar windows, setting every time the Object Type to **Host** and Subnet Mask to `255.255.255.255` and entering the IP address for that specific object (see 8.4.3, "Network Configuration" on page 388). This is the window that we used to define the 95-client object:



(LOCAL) Modify a Network Object

Modify Network Object

Identification

Object Type:	Host
Object Name:	95-client
Description:	ThinkPad

User Information

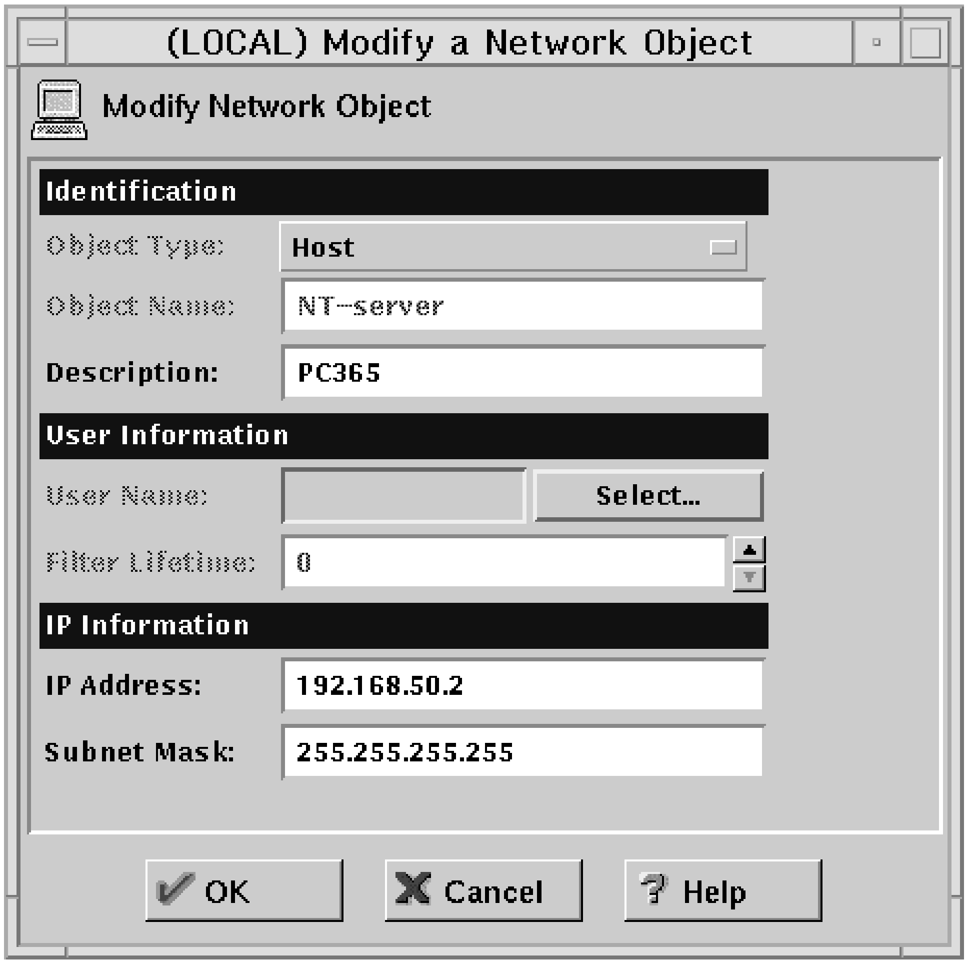
User Name:	
Filter Lifetime:	0

IP Information

IP Address:	192.168.51.2
Subnet Mask:	255.255.255.255

Figure 373. 95-client Object Definition

And Figure 422 on page 456 is the window with which we defined the NT-server object:



(LOCAL) Modify a Network Object

Modify Network Object

Identification

Object Type:	Host
Object Name:	NT-server
Description:	PC365

User Information

User Name:	
Filter Lifetime:	0

IP Information

IP Address:	192.168.50.2
Subnet Mask:	255.255.255.255

Figure 374. NT-server Object Definition

8.5.2 IP Filters Configuration for the DB2 Communication Protocol

We now describe the steps we followed to configure IP filters for the DB2 communication protocol over TCP/IP. IP traffic for this protocol flowed between the DB2 server AIX machine and the Web server NT machine, which in this case played the role of a DB2 client.

IBM Firewall provides a lot of predefined default services and rules that you can use for your purposes. In other words, you do not need to create new services and rules if among the default services and rules you find what you need for your environment. However, we did not find default services and rules matching our needs and for this reason we had to create new services and rules.

8.5.2.1 New Rules Creation

We started by selecting **Rules** from the Configuration Client navigation tree. This item can be accessed only after selecting **Traffic Control** and **Connection Template**, as you can see in the following figure:

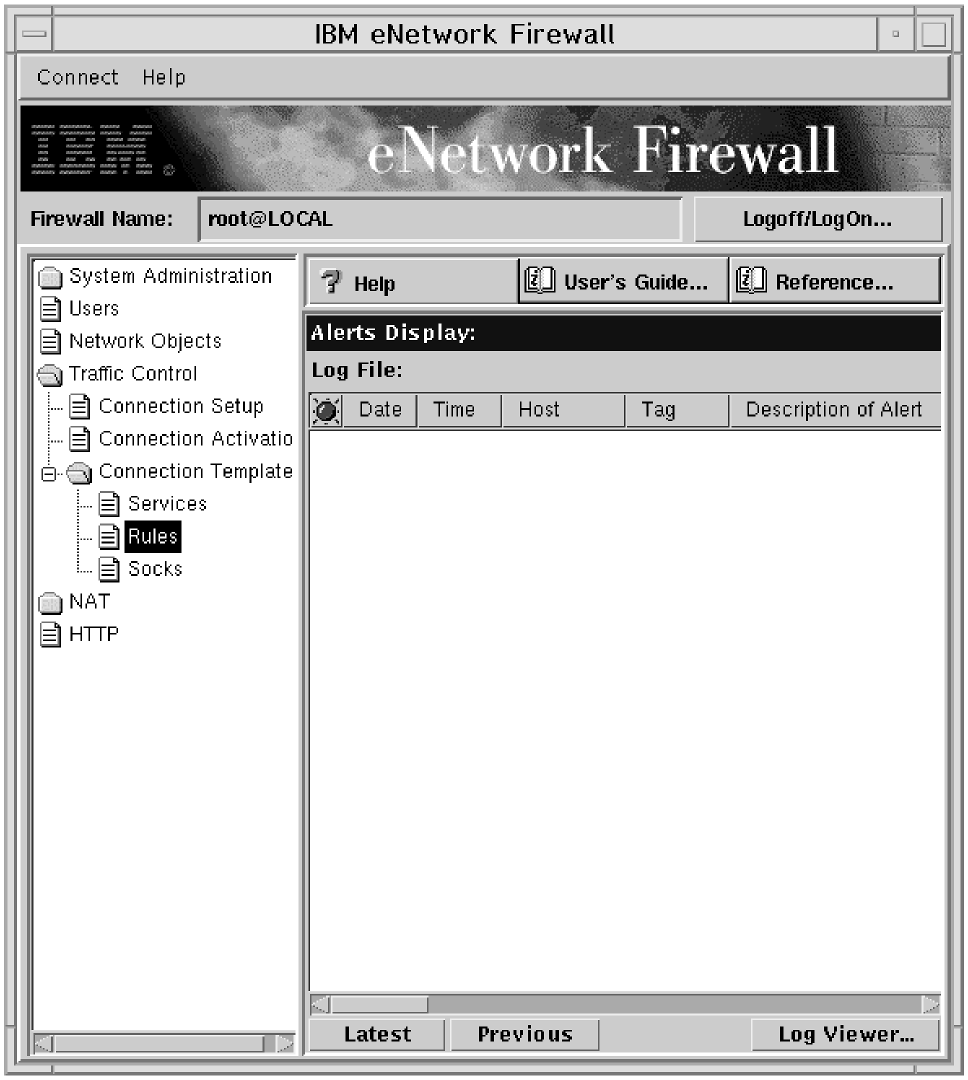
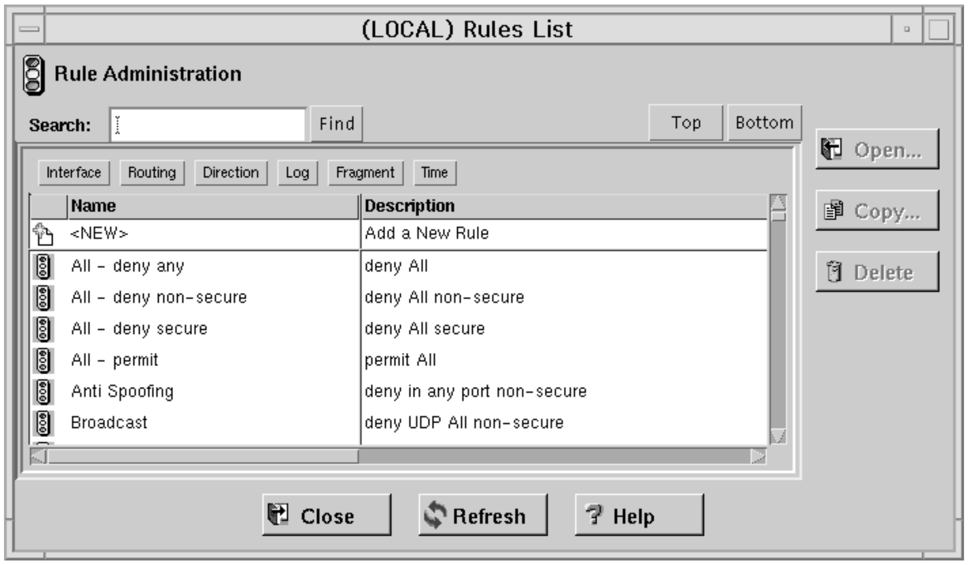


Figure 375. Rules in the Configuration Client Navigation Tree

When we got the Rules List page, we double clicked **NEW**, in order to create new rules, and we pressed the **Open:::** button:



Name	Description
<NEW>	Add a New Rule
All - deny any	deny All
All - deny non-secure	deny All non-secure
All - deny secure	deny All secure
All - permit	permit All
Anti Spoofing	deny in any port non-secure
Broadcast	deny UDP All non-secure

Figure 376. Rules List Window

As we said in 7.5.2, "DB2 CAE Connection Configuration" on page 342, on the DB2 server-side there are two specific port numbers associated with the DB2 instance containing the target database in the DB2 server machine. These port numbers are used for the communication with the DB2 CAE client application. Default values are:

1. 50000 for the main connection port (see Figure 330 on page 346).
2. 50001 for the interrupt port (the interrupt port number is always one number greater than the main connection port number).

We explained in 7.5.2, "DB2 CAE Connection Configuration" on page 342 that it is possible to customize the main connection port number on the server through the DB2 Manager Configuration File. The port number used on the client-side is greater than 1023, in accordance with the convention that port numbers below 1023 are considered privileged and commonly used by applications having root authority. But beside this limitation there is no specific port used on the client. The DB2 client simply asks TCP for a free port that is dynamically assigned and there is no way to customize the client port number.

As we have already explained, we did not find any predefined services and rules that could be used to build the connections we needed. For this reason, we had to define a set of new rules, add these rules to a service and then generate the proper connection using the new service. We started by creating exactly eight new rules. The following list describes these eight new rules:

1. DB2 CAE 1/4 non secure

   This rule was defined to permit the inbound IP routing using tr1 (which was the DMZ network interface) from a TCP port greater than 1023 to the TCP port 50000.

2. DB2 CAE 2/4 non secure

   This rule was defined to permit the outbound IP routing using en0 (which was the secure network interface) from a TCP port greater than 1023 to the TCP port 50000.

3. DB2 CAE 3/4 non secure

   This rule was defined to permit the inbound IP routing using tr1 (which was the DMZ network interface) from a TCP port greater than 1023 to the TCP port 50001.

4. DB2 CAE 4/4 non secure

   This rule was defined to permit the outbound IP routing using en0 (which was the secure network interface) from a TCP port greater than 1023 to the TCP port 50001.

5. DB2 CAE Ack 1/4 non secure

   This rule was defined to permit the outbound IP routing using tr1 (which was the DMZ network interface) from the TCP port 50000 to a TCP port greater than 1023.

6. DB2 CAE Ack 2/4 non secure

   This rule was defined to permit the inbound IP routing using en0 (which was the secure network interface) from the TCP port 50000 to a TCP port greater than 1023.

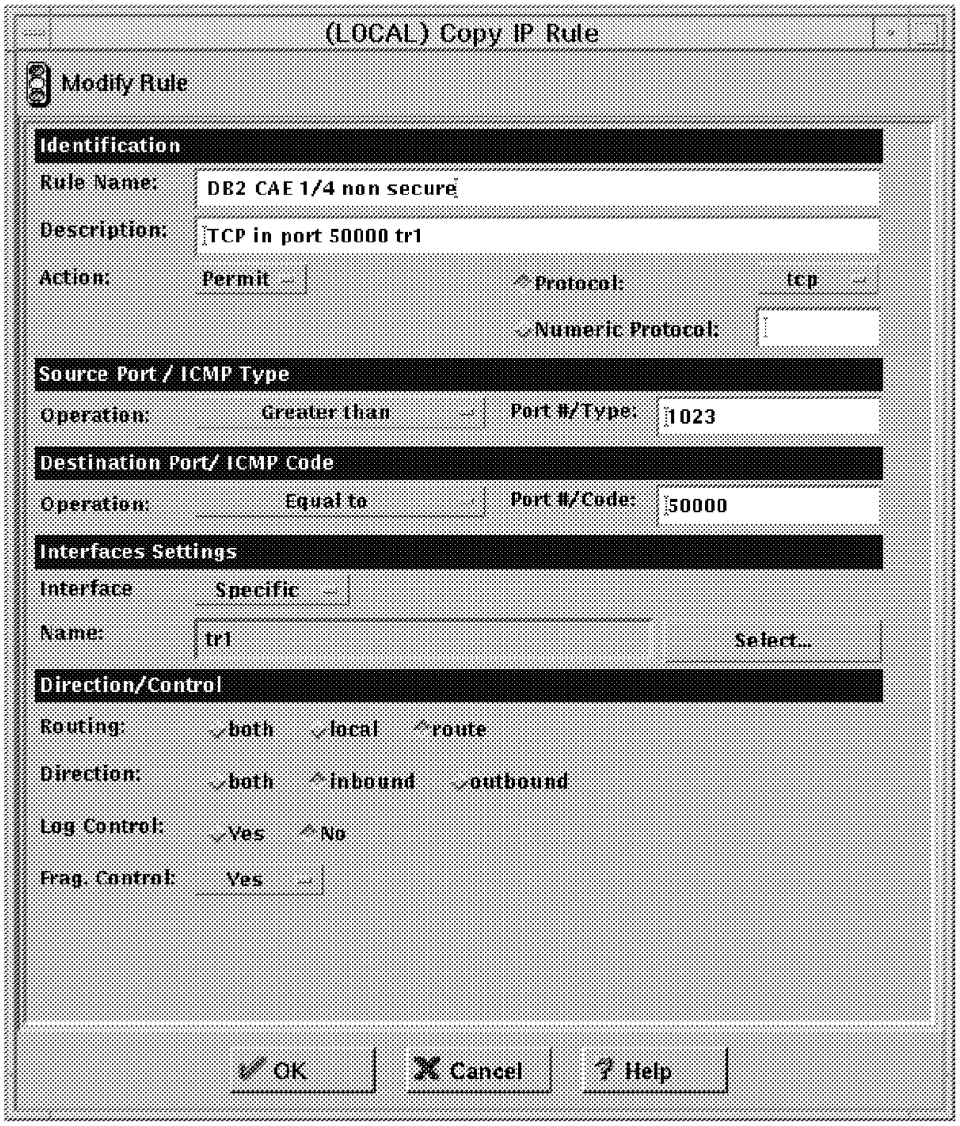
7. DB2 CAE Ack 3/4 non secure

   This rule was defined to permit the outbound IP routing using tr1 (which was the DMZ network interface) from the TCP port 50001 to a TCP port greater than 1023.

8. DB2 CAE Ack 4/4 non secure

   This rule was defined to permit the inbound IP routing using en0 (which was the secure network interface) from the TCP port 50001 to a TCP port greater than 1023.

The following screen captures from Figure 377 on page 400 through Figure 384 on page 407 show the windows we used to define these eight new rules. In case you want to repeat our scenario, you can consider these screens as an example to build your own rules.



Identification

Rule Name: DB2 CAE 1/4 non secure

Description: TCP in port 50000 tr1

Action: Permit

[✓ Protocol:] tcp

Numeric Protocol:

Source Port / ICMP Type

Operation: Greater than

Port #/Type: 1023

Destination Port/ ICMP Code

Operation: Equal to

Port #/Code: 50000

Interfaces Settings

Interface Specific

Name: tr1

Direction/Control

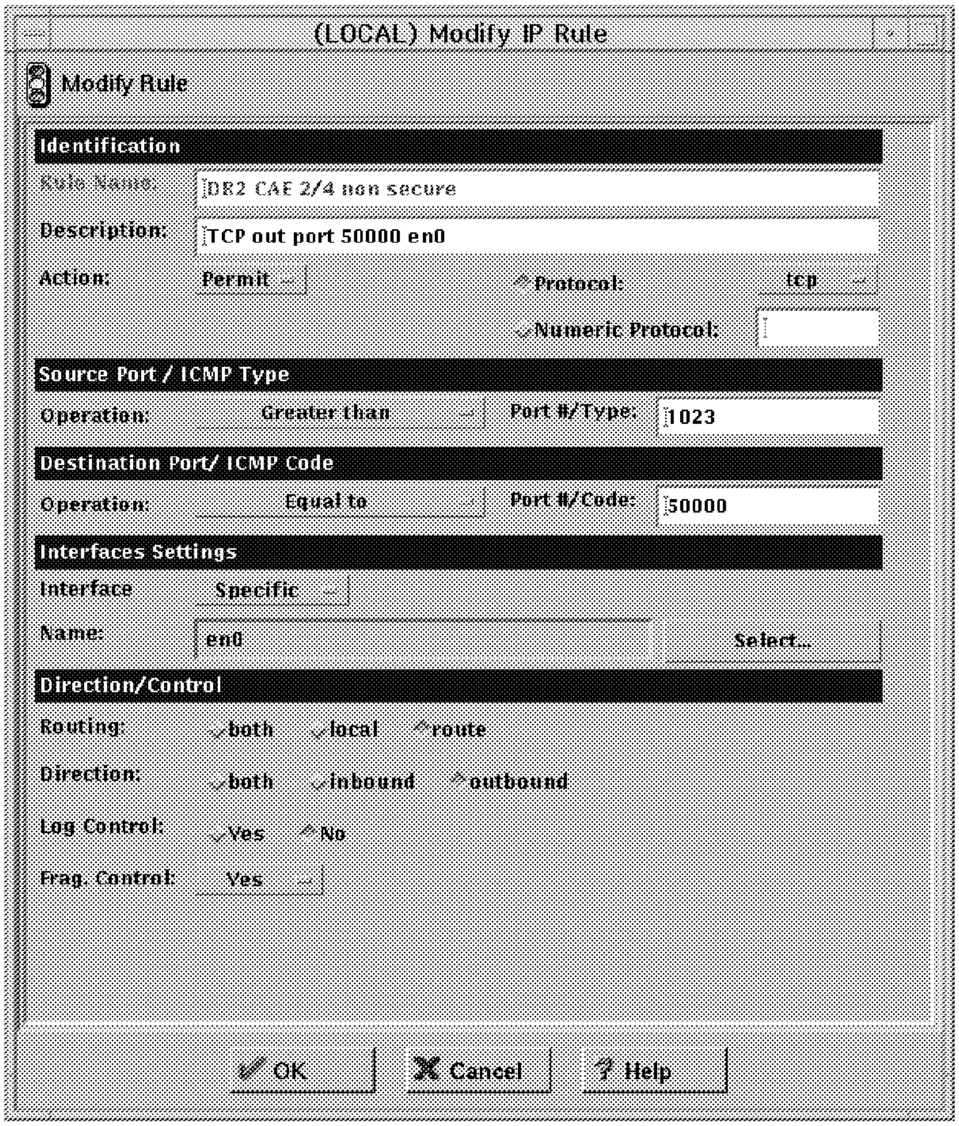
Routing: both local [✓ route]

Direction: both [✓ inbound] outbound

Log Control: Yes [✓ No]

Frag. Control: Yes

Figure 377. DB2 CAE 1/4 non secure Rule


(LOCAL) Modify IP Rule

Modify Rule

Identification

Rule Name: DB2 CAE 2/4 non secure

Description: TCP out port 50000 en0

Action: Permit

[✓ Protocol:] tcp

Numeric Protocol:

Source Port / ICMP Type

Operation: Greater than

Port #/Type: 1023

Destination Port/ ICMP Code

Operation: Equal to

Port #/Code: 50000

Interfaces Settings

Interface: Specific

Name: en0

Direction/Control

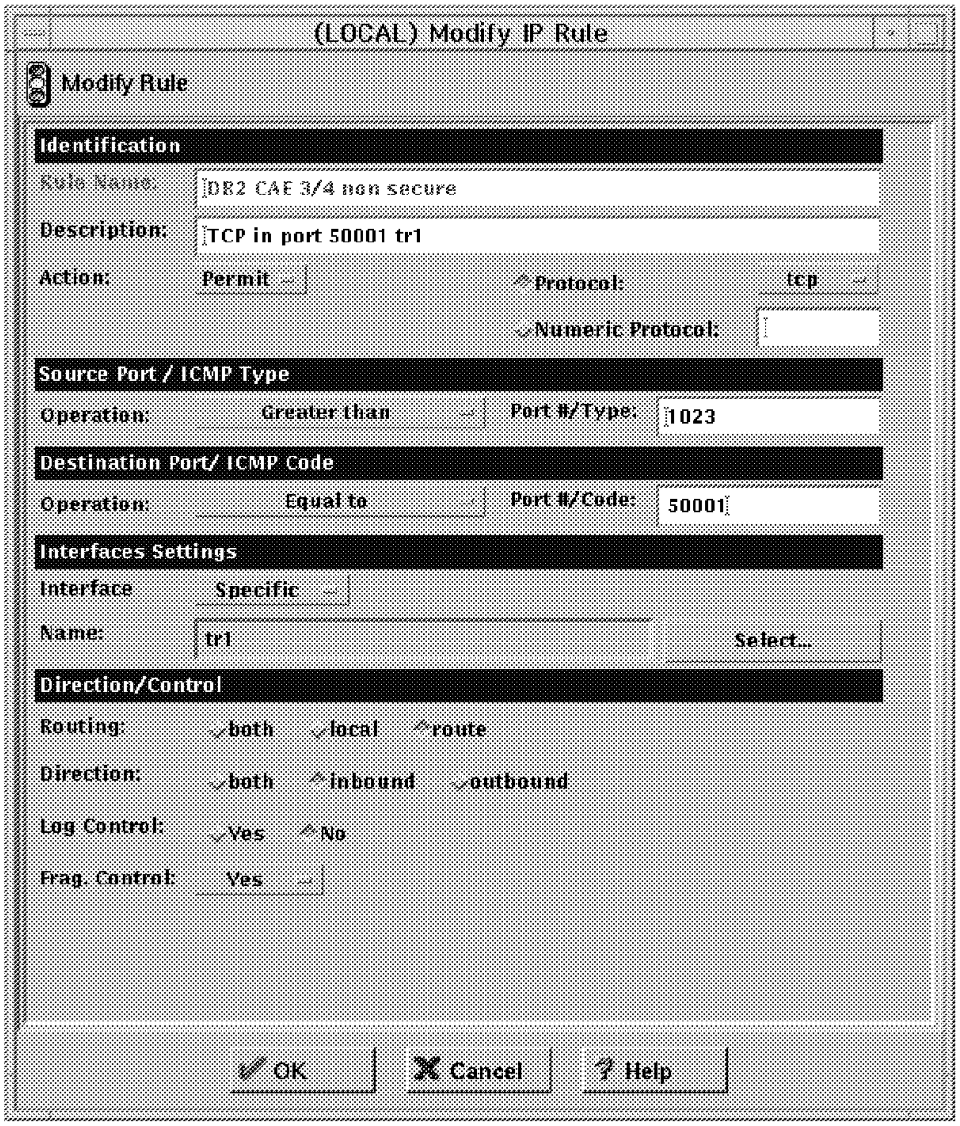
Routing: both local [✓ route]

Direction: both inbound [✓ outbound]

Log Control: Yes [✓ No]

Frag. Control: Yes

Figure 378. DB2 CAE 2/4 non secure Rule



Identification

Rule Name: DB2 CAE 3/4 non secure

Description: TCP in port 50001 tr1

Action: Permit

[✓ Protocol:] tcp

Numeric Protocol:

Source Port / ICMP Type

Operation: Greater than Port #/Type: 1023

Destination Port/ ICMP Code

Operation: Equal to Port #/Code: 50001

Interfaces Settings

Interface Specific

Name: tr1

Direction/Control

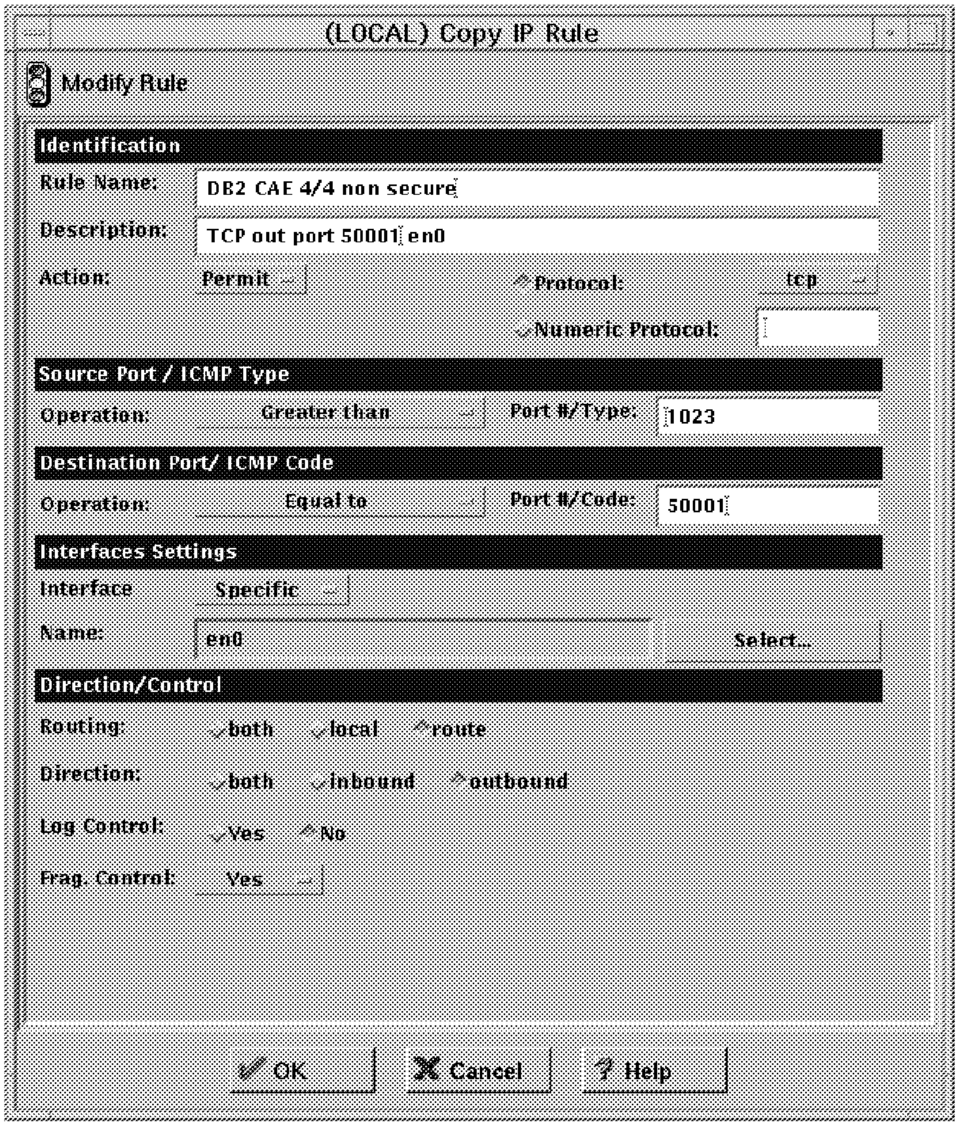
Routing: both local [✓ route]

Direction: both [✓ inbound] outbound

Log Control: Yes [✓ No]

Frag. Control: Yes

Figure 379. DB2 CAE 3/4 non secure Rule



Identification

Rule Name:	DB2 CAE 4/4 non secure
Description:	TCP out port 50001 en0
Action:	Permit
[✓ Protocol:]	tcp
Numeric Protocol:	

Source Port / ICMP Type

Operation:	Greater than
Port #/Type:	1023

Destination Port/ ICMP Code

Operation:	Equal to
Port #/Code:	50001

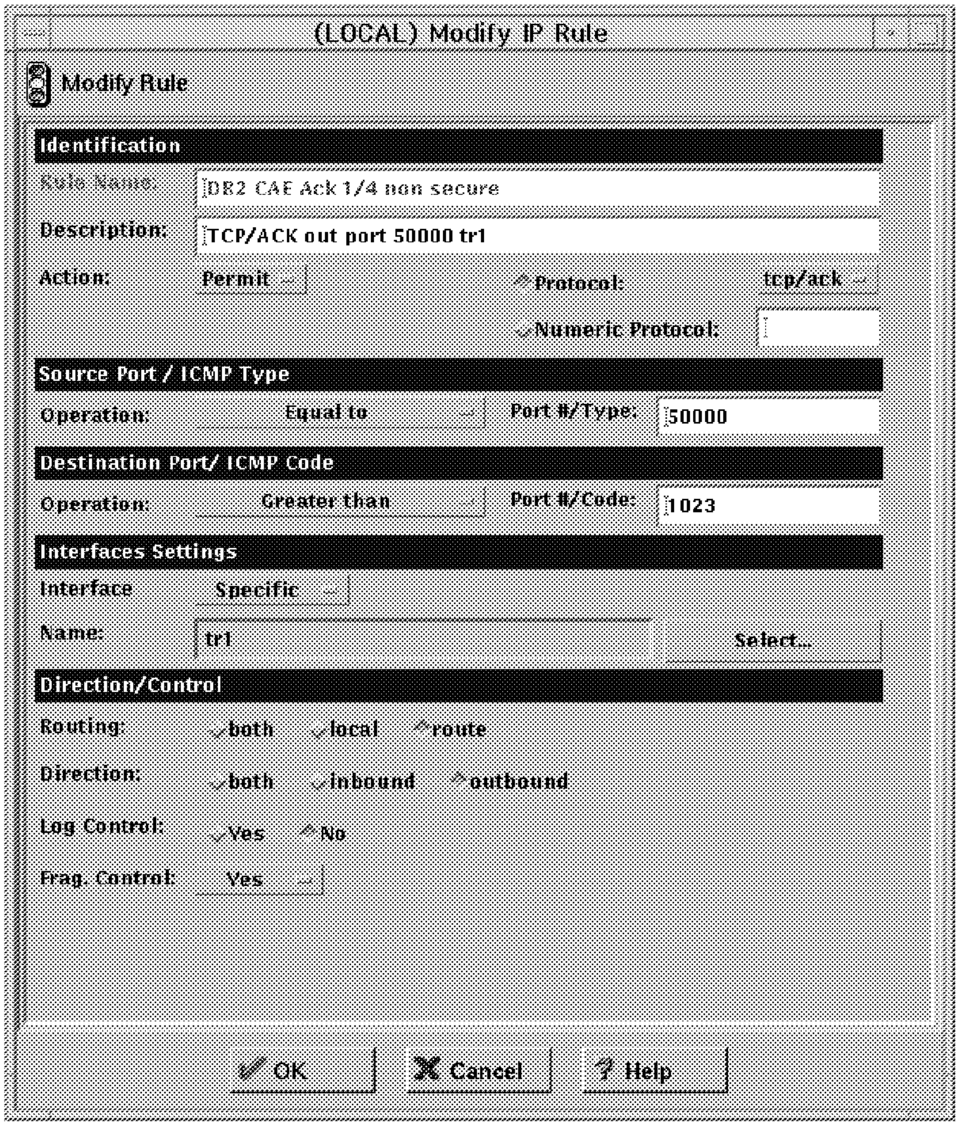
Interfaces Settings

Interface	Specific
Name:	en0

Direction/Control

Routing:	both local [✓ route]
Direction:	both inbound [✓ outbound]
Log Control:	Yes [✓ No]
Frag. Control:	Yes

Figure 380. DB2 CAE 4/4 non secure Rule


(LOCAL) Modify IP Rule

Modify Rule

Identification

Rule Name: DB2 CAE Ack 1/4 non secure

Description: TCP/ACK out port 50000 tr1

Action: Permit

[✓ Protocol:] tcp/ack

Numeric Protocol:

Source Port / ICMP Type

Operation: Equal to

Port #/Type: 50000

Destination Port/ ICMP Code

Operation: Greater than

Port #/Code: 1023

Interfaces Settings

Interface: Specific

Name: tr1

Select...

Direction/Control

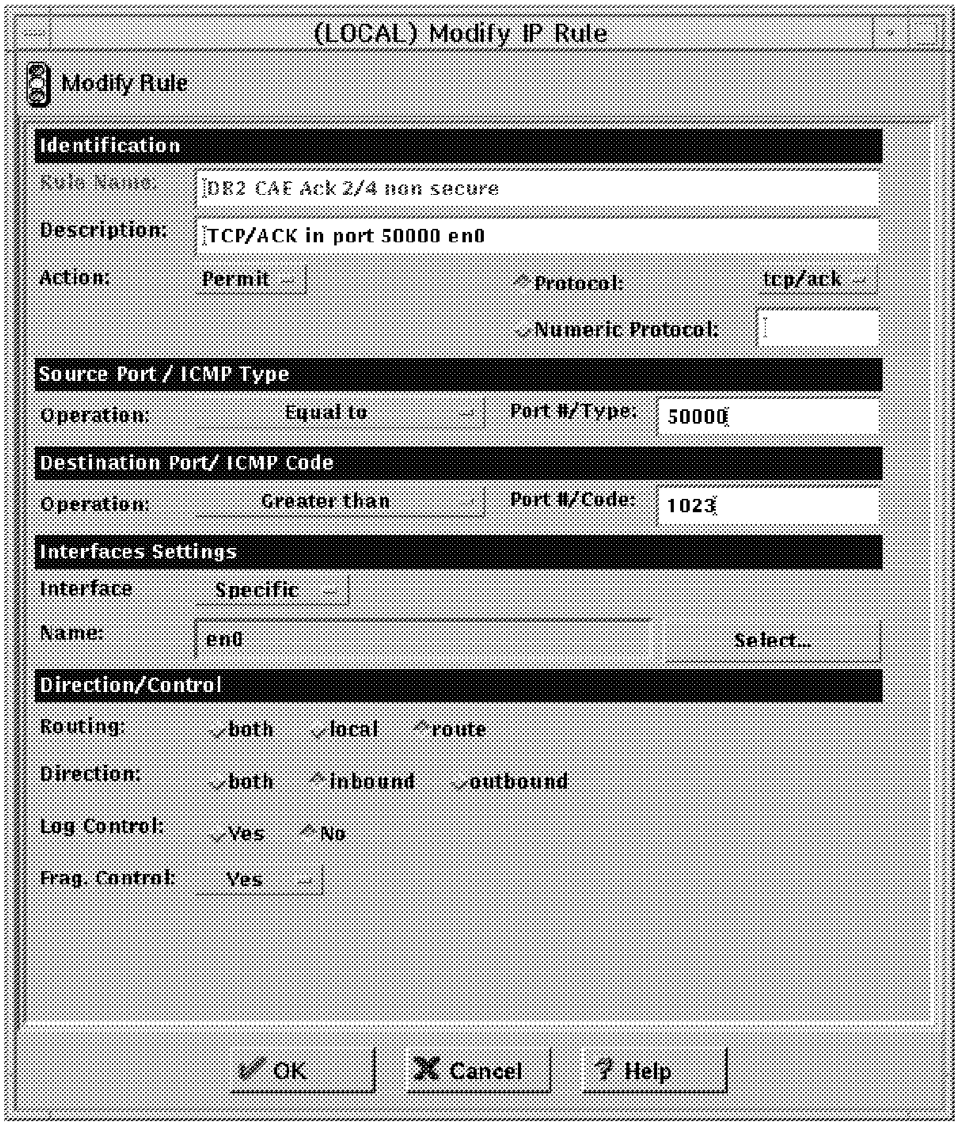
Routing: both local [✓ route]

Direction: both inbound [✓ outbound]

Log Control: Yes [✓ No]

Frag. Control: Yes

Figure 381. DB2 CAE Ack 1/4 non secure Rule


(LOCAL) Modify IP Rule

Modify Rule

Identification

Rule Name:	DB2 CAE Ack 2/4 non secure		
Description:	TCP/ACK in port 50000 en0		
Action:	Permit	[✓ Protocol:]	tcp/ack
		Numeric Protocol:	

Source Port / ICMP Type

Operation:	Equal to	Port #/Type:	50000

Destination Port/ ICMP Code

Operation:	Greater than	Port #/Code:	1023

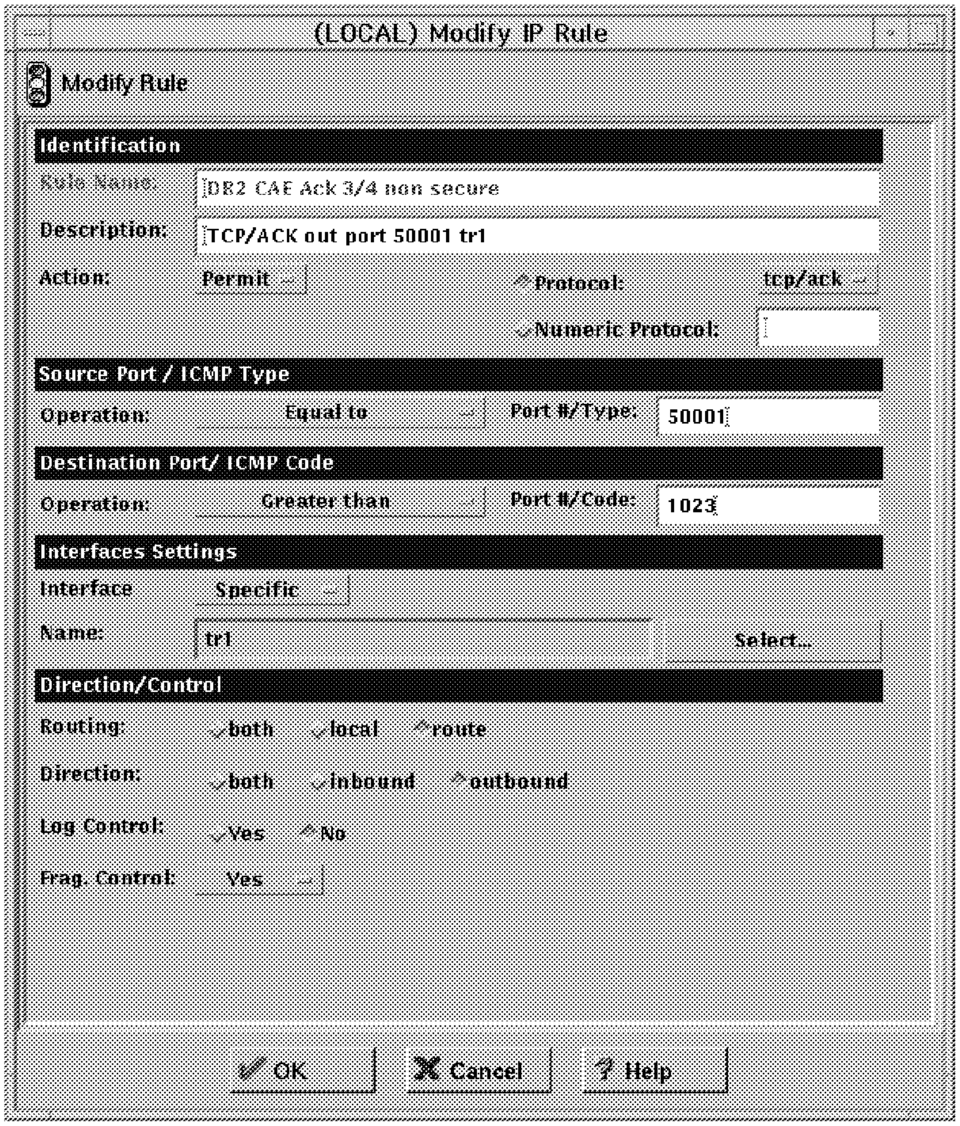
Interfaces Settings

Interface	Specific
Name:	en0

Direction/Control

Routing:	both local [✓ route]
Direction:	both [✓ inbound] outbound
Log Control:	Yes [✓ No]
Frag. Control:	Yes

Figure 382. DB2 CAE Ack 2/4 non secure Rule


(LOCAL) Modify IP Rule

Modify Rule

Identification

Rule Name:	DB2 CAE Ack 3/4 non secure
Description:	TCP/ACK out port 50001 tr1
Action:	Permit
[✓ Protocol:]	tcp/ack
Numeric Protocol:	

Source Port / ICMP Type

Operation:	Equal to
Port #/Type:	50001

Destination Port/ ICMP Code

Operation:	Greater than
Port #/Code:	1023

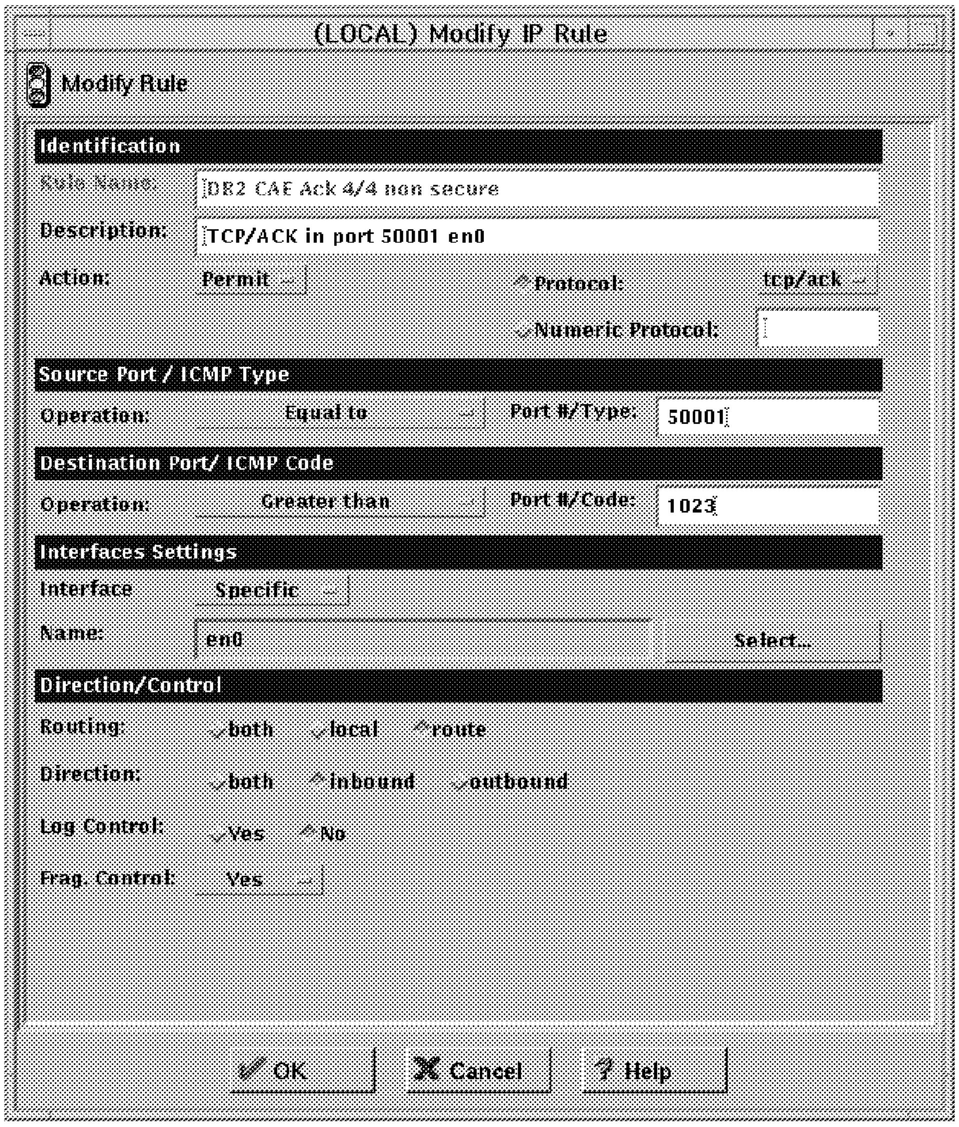
Interfaces Settings

Interface	Specific
Name:	tr1

Direction/Control

Routing:	both local [✓ route]
Direction:	both inbound [✓ outbound]
Log Control:	Yes [✓ No]
Frag. Control:	Yes

Figure 383. DB2 CAE Ack 3/4 non secure Rule



(LOCAL) Modify IP Rule

Modify Rule

Identification

Rule Name: DB2 CAE Ack 4/4 non secure

Description: TCP/ACK in port 50001 en0

Action: Permit

Protocol: tcp/ack

Numeric Protocol:

Source Port / ICMP Type

Operation: Equal to   Port #/Type: 50001

Destination Port/ ICMP Code

Operation: Greater than   Port #/Code: 1023

Interfaces Settings

Interface: Specific

Name: en0

Direction/Control

Routing: both local [✓ route]

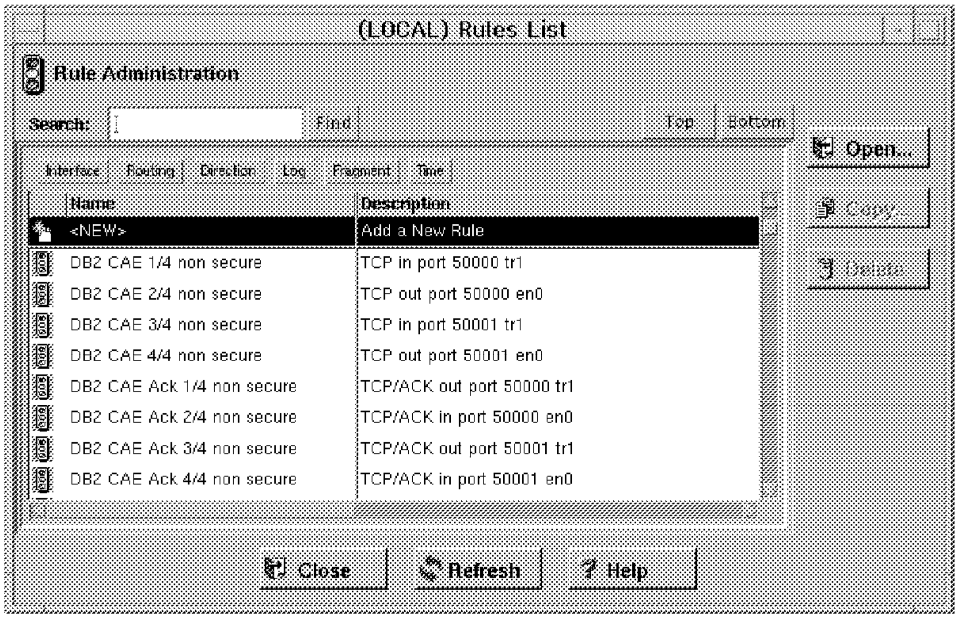
Direction: both [✓ inbound] outbound

Log Control: Yes [✓ No]

Frag. Control: Yes

Figure 384. DB2 CAE Ack 4/4 non secure Rule

As soon as these eight new rules were created, they were immediately registered and displayed by the Rules List window, as we show in the following figure:



Name	Description
<NEW>	Add a New Rule
DB2 CAE 1/4 non secure	TCP in port 50000 tr1
DB2 CAE 2/4 non secure	TCP out port 50000 en0
DB2 CAE 3/4 non secure	TCP in port 50001 tr1
DB2 CAE 4/4 non secure	TCP out port 50001 en0
DB2 CAE Ack 1/4 non secure	TCP/ACK out port 50000 tr1
DB2 CAE Ack 2/4 non secure	TCP/ACK in port 50000 en0
DB2 CAE Ack 3/4 non secure	TCP/ACK out port 50001 tr1
DB2 CAE Ack 4/4 non secure	TCP/ACK in port 50001 en0

Figure 385. DB2 CAE Rules in the Rules List Window

Notice that for the last four rules that we defined, in the Protocol field we selected **tcp/ack** (see Figure 381 on page 404 through Figure 384 on page 407) instead of **tcp**. This means that we permitted only TCP packets having the Acknowledgment (ACK) flag set to flow from the DB2 server machine to the DB2 client machine. In general this choice has security implications. In fact the ACK flag in a TCP session is used by one end of a session to inform the other end that the previous packet was received correctly. For this reason, in a TCP session, only the first packet, which is responsible for establishing the session, does not have the ACK flag set. Since the DB2 server never initiates a connections with the DB2 client, only packets with the ACK flag set should flow from the DB2 server to the DB2 client. The tcp/ack Protocol specification prevents an undesired TCP session from being established, since it blocks the first packet.

8.5.2.2 New Service Creation

This section describes how we added the eight new rules to a new service that we named `CAE direct in`. This service defined what type of IP traffic we wanted to permit between the two network objects NT-server and DB-server.

First of all, we selected **Services** in the Client Configuration navigation tree, displayed in the following screen:

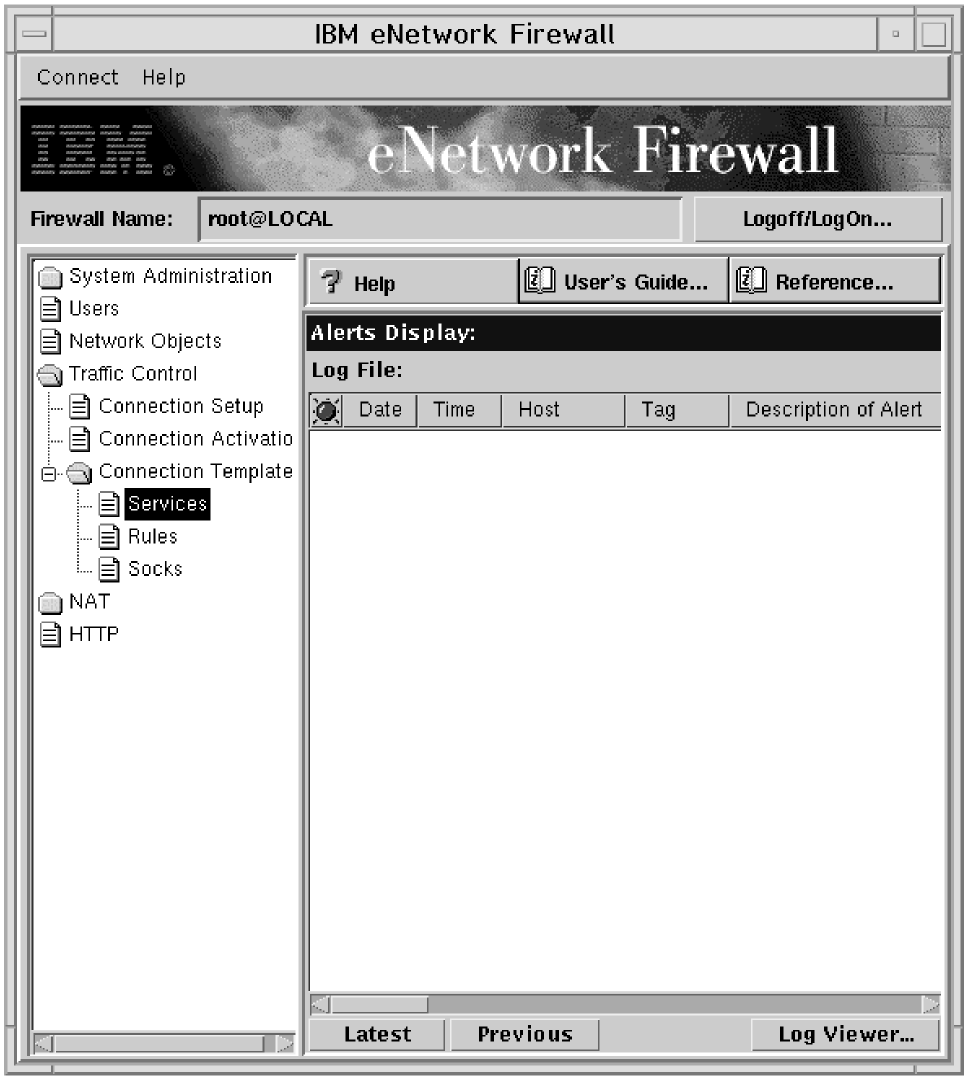


Figure 386. Client Configuration Navigation Tree

Notice that in order to select **Services**, you first need to select **Traffic Control** and **Connection Template**.

Once the Services List page is displayed, you should click **NEW**, in order to add a new service, and then press the **Open:::** button, as shown in the following figure:

The Add Service window will be displayed. We used this window to add the new service. We entered the service name `CAE direct in` and then typed in the Description field `Permit CAE from DMZ directly to DB2`. Then we added the new rules. To add rules during a service definition, you should click the **Select:::** button in the Service Composition section. The Rules List window is brought up and you can pick up the rules that you need for your service definition. The result is shown in the following window:

You can see that we did not modify the sections Service Override Values or Time Controls for the Add Service window. You should have noticed the presence of an arrow beside each rule name. That arrow can go from left to right or from right to left and it is to specify the flow. After selecting a rule, you can change the direction of one arrow to the opposite by clicking the **Flow** button.

1. A left to right arrow indicates to the firewall that when that service is added to a connection, where the source and the destination objects are specified, the rule will be applied to the traffic that flows from the source object to the destination object.
2. A right to left arrow indicates to the firewall that when that service is added to a connection, where the source and the destination objects are specified, the rule will be applied to the traffic that flows from the destination object to the source object.

The connection that we were going to build would have NT-server as its source object and DB-server as its destination object. But for the four rules for which we had selected **tcp/ack** as Protocol specification (see Figure 381 on page 404 through Figure 384 on page 407), it was supposed that the IP traffic would flow from the destination object to the source object. For this reason we marked these four rules with right to left arrows. For the other four rules, for which we had selected **tcp** as Protocol specification (see Figure 377 on page 400 through Figure 380 on page 403), we kept the default left to right arrows, since they would apply to the IP traffic flowing from the source object to the destination object for the connection.

If you make a mistake when you add the rules to the Add Service page, and you want to remove a rule, you can do this by simply selecting the wrong rule and then clicking **Remove**.

The order with which the rules are listed in the Add Service window is also very important from a security point of view, since when IBM Firewall receives a packet, it compares that packet to the rules for that service in the same order with which they have been added, and stops comparing only when the first match is found. After that, it executes the action described in the matching rule. We added the eight rules in the same order with which we had created them (see 8.5.2.1, “New Rules Creation” on page 396). You can change the order with which you added the rules to your service by selecting a rule and clicking **Move Up** or **Move Down**.

8.5.2.3 Connection Configuration

After clicking **OK** in the Add Service window, we needed to add the new service to a new connection that we named `DMZ to Secure DB2`. To do this, we selected **Traffic Control** and then **Connection Setup** in the Configuration Client navigation tree:

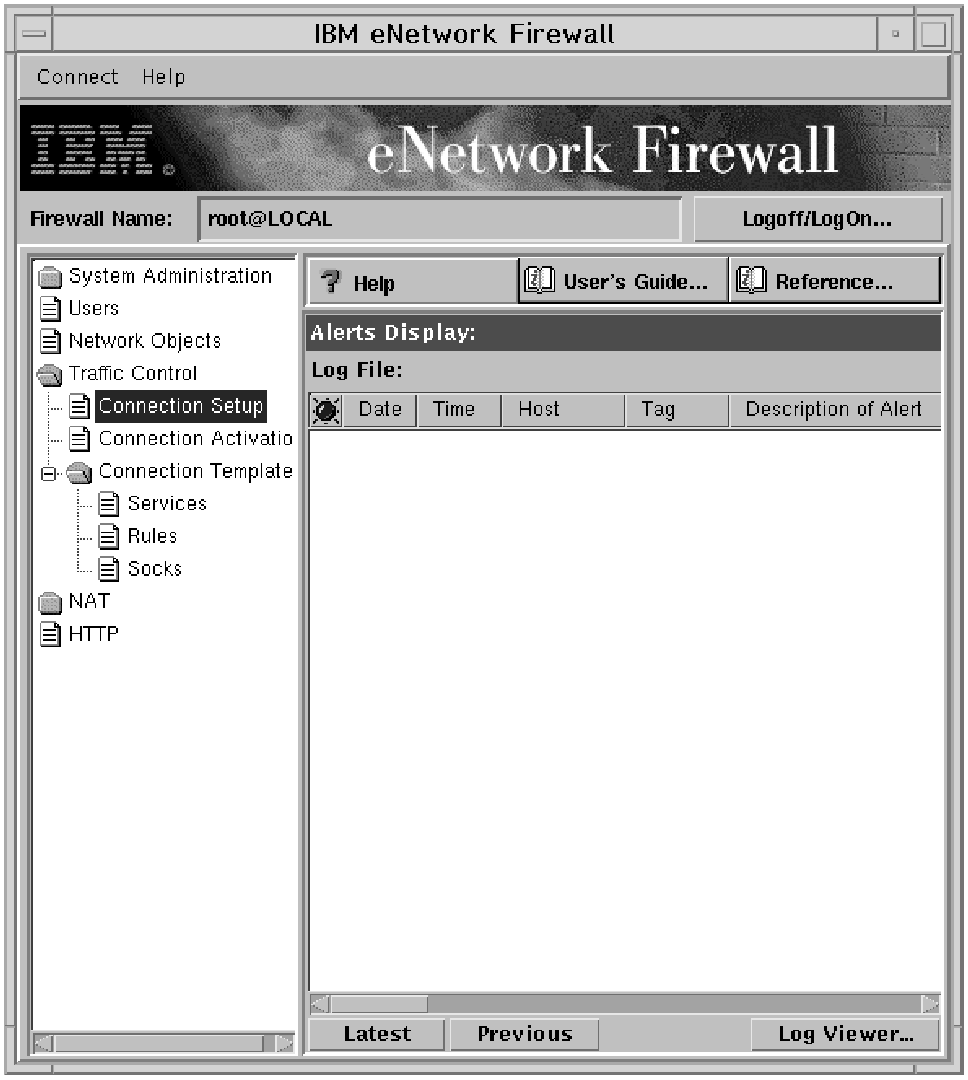
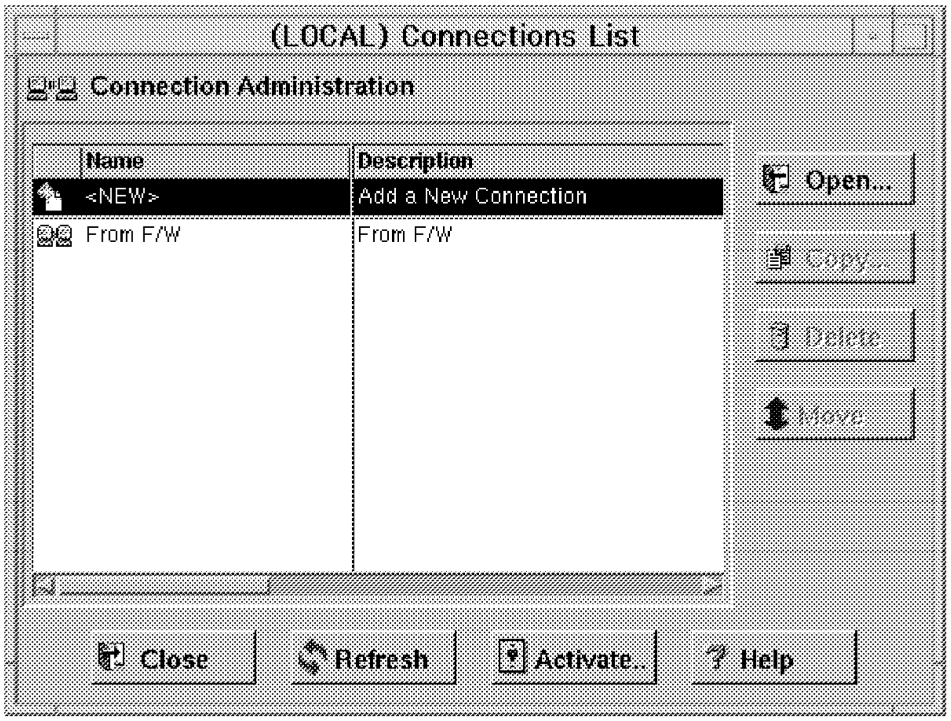


Figure 387. Connection Setup in the Configuration Client Navigation Tree

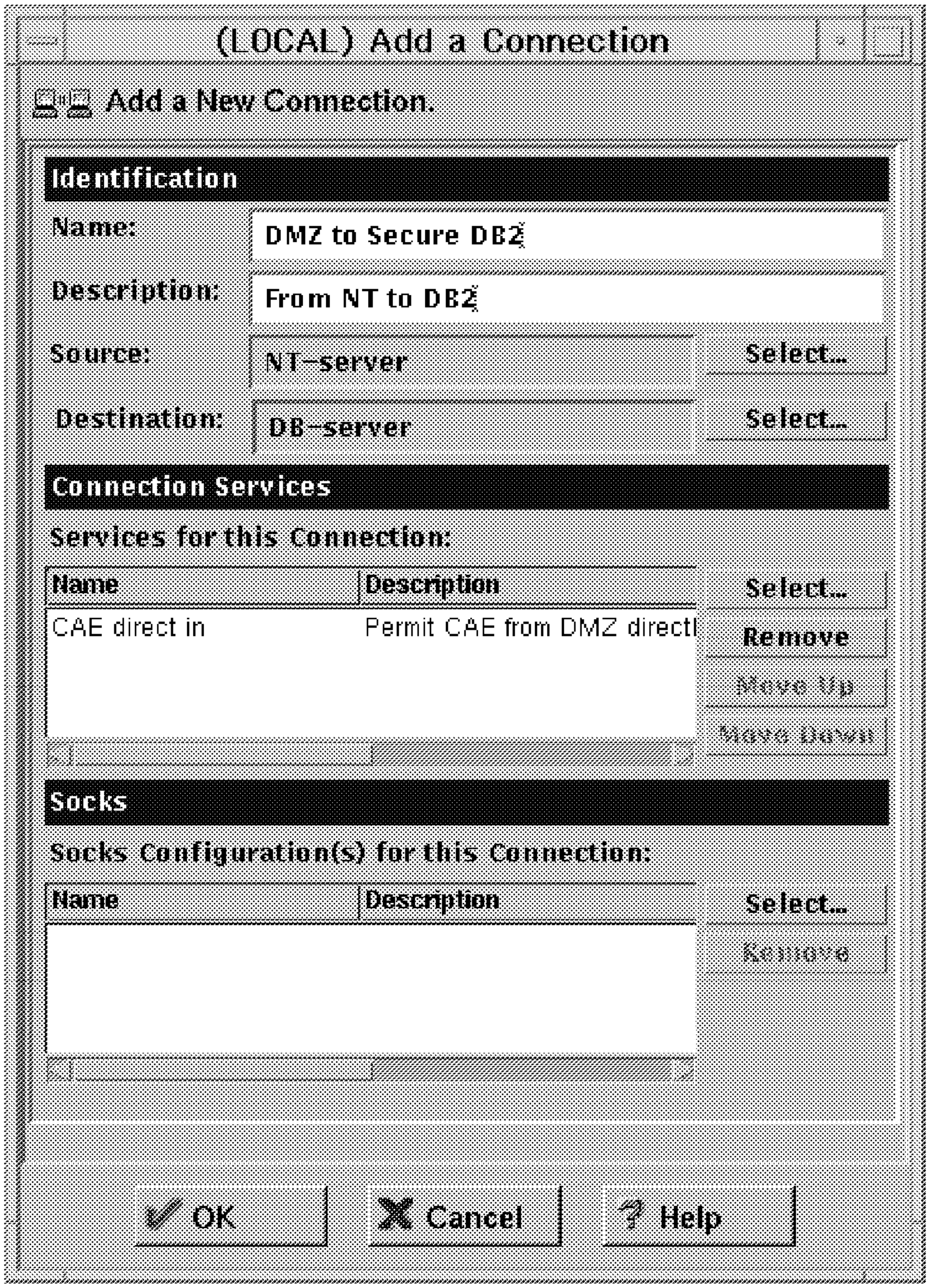
The Connections List window was brought up:



Name	Description
<NEW>	Add a New Connection
From F/W	From F/W

Figure 388. Adding a New Connection through the Connections List Window

We selected **NEW** and then we pressed the **Open:::** button. The Add a Connection window appeared. We set Name to `DMZ to Secure DB2` and Description to `From NT to DB2`. Then we selected **NT-server** as source network object and **DB-server** as destination network object. Also we selected **CAE direct in** as the only service for this connection. The Add a Connection window looked like the following screen:



Name:	DMZ to Secure DB2
Description:	From NT to DB2
Source:	NT-server
Destination:	DB-server

Name	Description
CAE direct in	Permit CAE from DMZ directl

Figure 389. DB2 CAE Connection

Then we simply pressed **OK**. This connection was to permit the IP communication between DB2 client and server to flow through the firewall. In order for this connection to work, we needed to activate it. To do this, we selected **Traffic Control** and then **Connection Activation** in the Configuration Client navigation tree, as shown in the following window:

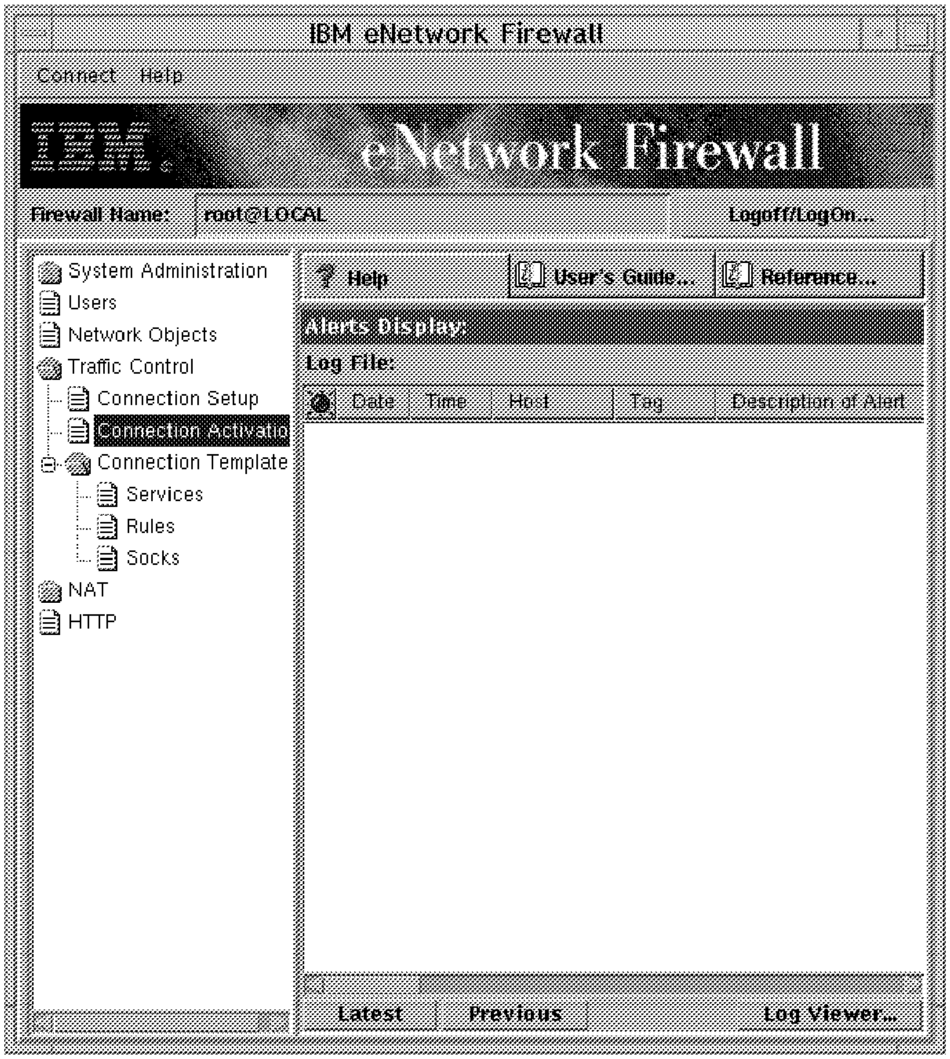
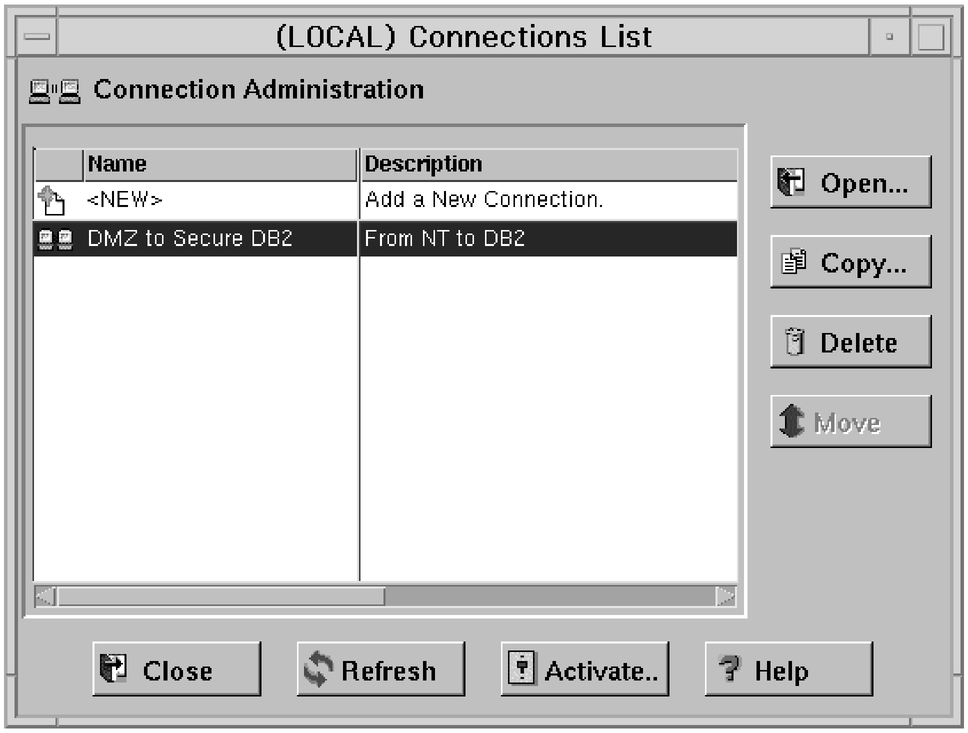


Figure 390. Connection Activation in the Configuration Client Navigation Tree

Once the Connection List window appeared, we could activate the new connection we had built, DMZ to Secure DB2, by clicking on the **Activate...** button.

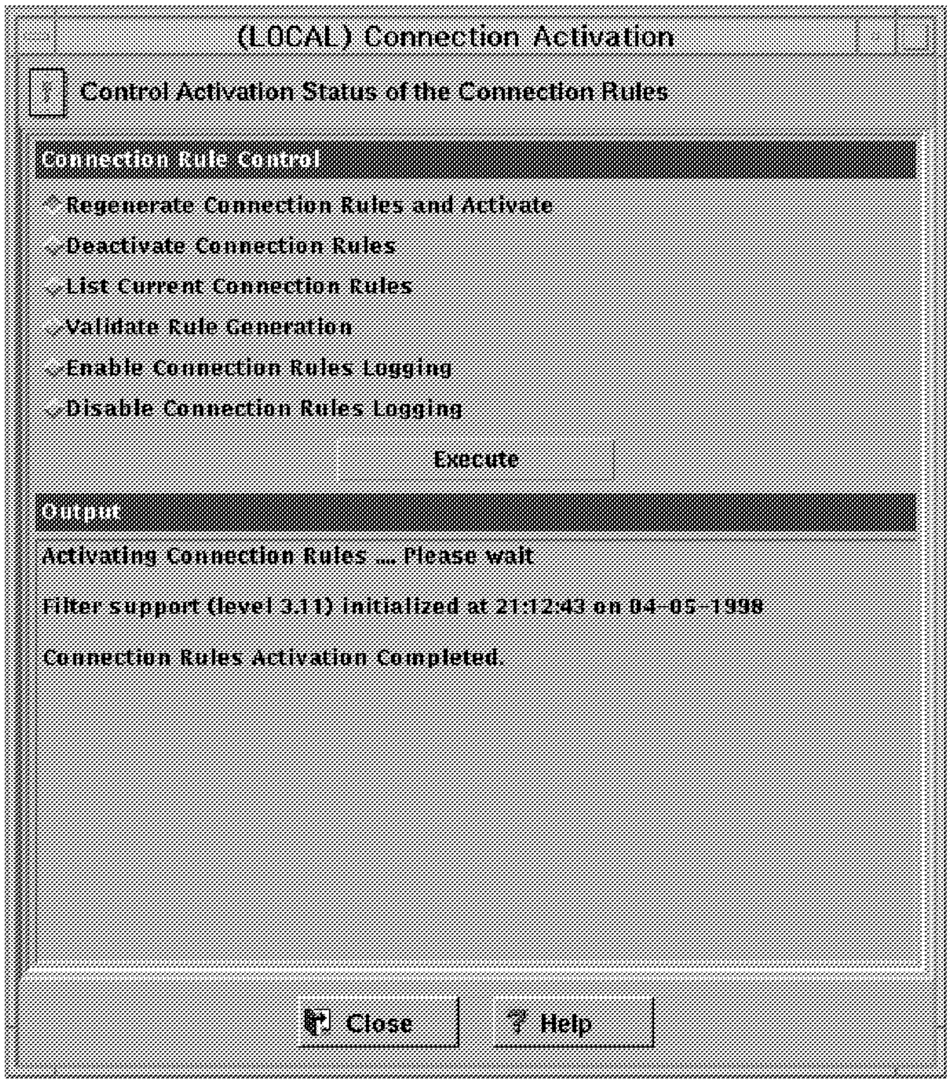


Name	Description
<NEW>	Add a New Connection.
DMZ to Secure DB2	From NT to DB2

Figure 391. Connections List

Remember that the connections that appear in this list when you click **Activate...** will all be activated, not only the highlighted one, so you should have in this window only the connections that you really want to activate. If you had created other connections, perhaps for testing purposes, make sure that you remove them from the Connections List window, by clicking on **Delete**, before the activation takes place. This way you will limit the connections allowed through the firewall and only the necessary connections will be activated.

The Connection Activation window was brought up. Here we selected the item **Regenerate Connection Rules and Activate** and then we clicked **Execute**.


(LOCAL) Connection Activation
Control Activation Status of the Connection Rules
Connection Rule Control
Regenerate Connection Rules and Activate
Deactivate Connection Rules
List Current Connection Rules
Validate Rule Generation
Enable Connection Rules Logging
Disable Connection Rules Logging
Execute
Output
Close
Help

```
Activating Connection Rules .... Please wait

Filter support (level 3.11) initialized at 21:12:43 on 04-05-1998

Connection Rules Activation Completed.
```

Figure 392. Connection Activation Window

8.5.3 IP Filters Configuration for HTTP and SSL

We describe now the steps that we followed to configure IP filters for HTTP and SSL. IP traffic for these two protocols flowed between the Windows 95 machine, where we had installed a Web browser, and the Windows NT Server machine, where we had installed the Web server application. We had configured on the Web server the HTTP port number, setting it to its default value 80, and the SSL port number, setting it to its default value 443 (see Figure 284 on page 307). The HTTP port number used on the client has a dynamic value always greater than 1023, since port numbers below 1024 are in the assigned port range, as we explained in 8.5.2.1, “New Rules Creation” on page 396. Keeping this in mind, we started to create rules and services in order to pass the HTTP and SSL protocols through the firewall.

8.5.3.1 New Rules Creation for HTTP

In order to create new rules, we selected **Rules** from the Configuration Client navigation tree (see Figure 375 on page 397). When we got the Rules List page (see Figure 376 on page 398), we clicked **NEW** and then we pressed the **Open:::** button. We could then create four new rules for the HTTP protocol, that we describe now through the following list:

1. HTTP 1/2 non secure

   This rule was defined to permit the inbound IP routing using tr0 (which was the non-secure network interface) from a TCP port greater than 1023 to the TCP port 80.

2. HTTP 2/2 non secure

   This rule was defined to permit the outbound IP routing using tr1 (which was the DMZ network interface) from the TCP port 80 to a TCP port greater than 1023.

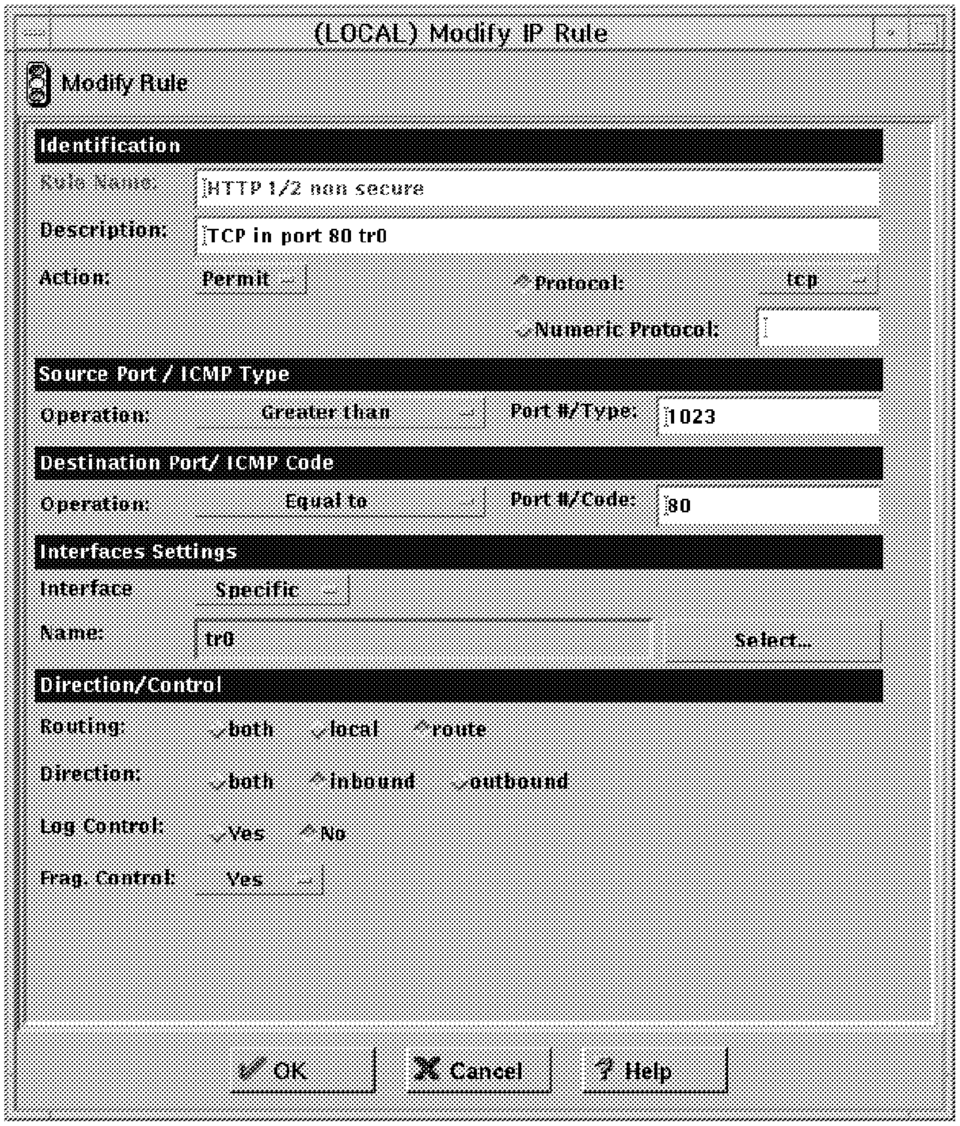
3. HTTP Ack 1/2 non secure

   This rule was defined to permit the inbound IP routing using tr1 (which was the DMZ network interface) from the TCP port 80 to a TCP port greater than 1023.

4. HTTP Ack 2/2 non secure

   This rule was defined to permit the outbound IP routing using tr0 (which was the DMZ network interface) from the TCP port 80 to a TCP port greater than 1023.

For each rule of the above list, we show now the screens that we used for the rule definitions from Figure 393 on page 418 through Figure 396 on page 421. You can consider these screens as an example in case you want to build a scenario similar to ours.


(LOCAL) Modify IP Rule

Modify Rule

Identification

Rule Name: HTTP 1/2 non secure

Description: TCP in port 80 tr0

Action: Permit

Protocol: tcp

Numeric Protocol:

Source Port / ICMP Type

Operation: Greater than

Port #/Type: 1023

Destination Port/ ICMP Code

Operation: Equal to

Port #/Code: 80

Interfaces Settings

Interface: Specific

Name: tr0

Direction/Control

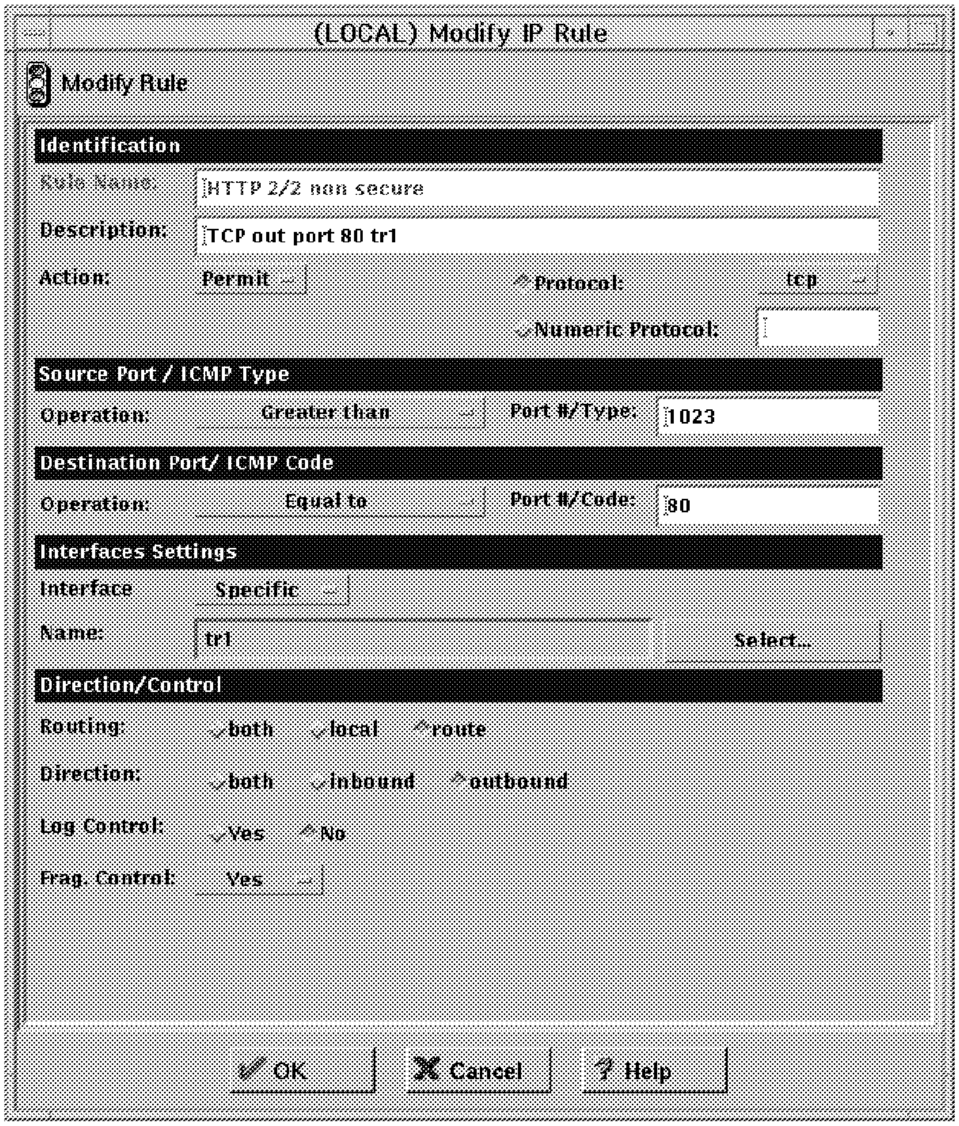
Routing: both local [✓ route]

Direction: both [✓ inbound] outbound

Log Control: Yes [✓ No]

Frag. Control: Yes

Figure 393. HTTP 1/2 non secure Rule


(LOCAL) Modify IP Rule

Modify Rule

Identification

Rule Name: HTTP 2/2 non secure

Description: TCP out port 80 tr1

Action: Permit

[✓ Protocol:] tcp

Numeric Protocol:

Source Port / ICMP Type

Operation: Greater than

Port #/Type: 1023

Destination Port/ ICMP Code

Operation: Equal to

Port #/Code: 80

Interfaces Settings

Interface: Specific

Name: tr1

Direction/Control

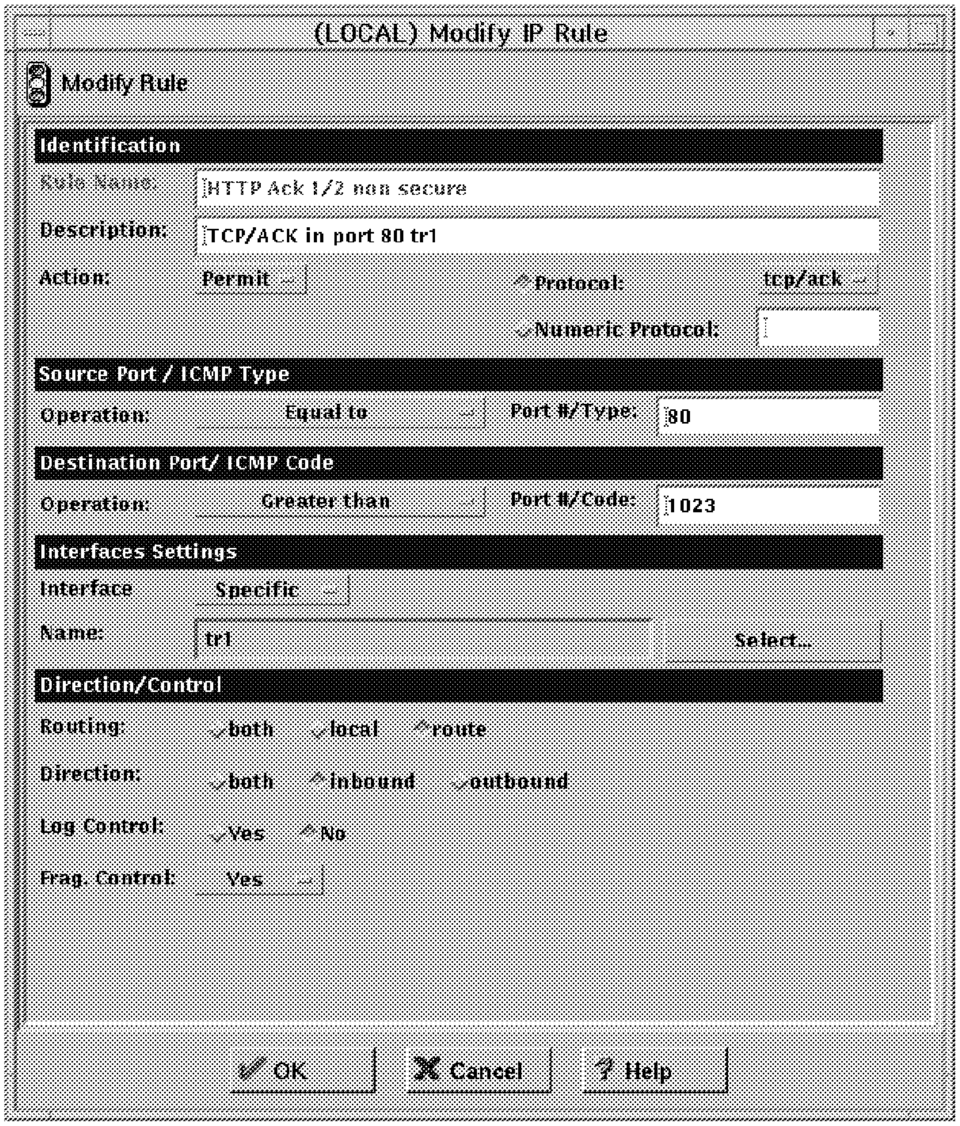
Routing: both local [✓ route]

Direction: both inbound [✓ outbound]

Log Control: Yes [✓ No]

Frag. Control: Yes

Figure 394. HTTP 2/2 non secure Rule



Identification

Rule Name: HTTP Ack 1/2 non secure

Description: TCP/ACK in port 80 tr1

Action: Permit

[✓ Protocol:] tcp/ack

Numeric Protocol:

Source Port / ICMP Type

Operation: Equal to

Port #/Type: 80

Destination Port/ ICMP Code

Operation: Greater than

Port #/Code: 1023

Interfaces Settings

Interface: Specific

Name: tr1

Direction/Control

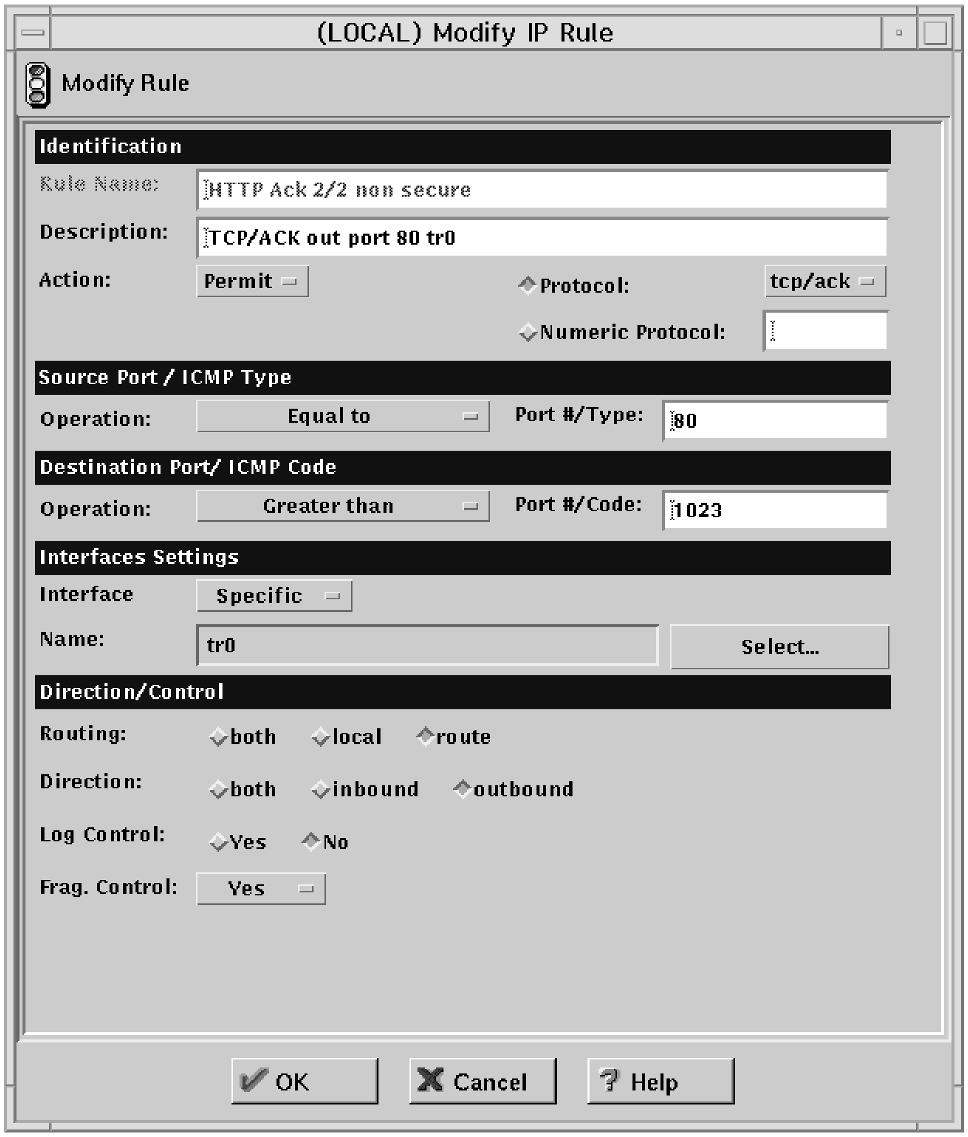
Routing: both local [✓ route]

Direction: both [✓ inbound] outbound

Log Control: Yes [✓ No]

Frag. Control: Yes

Figure 395. HTTP Ack 1/2 non secure Rule


(LOCAL) Modify IP Rule

Modify Rule

Identification

Rule Name: HTTP Ack 2/2 non secure

Description: TCP/ACK out port 80 tr0

Action: Permit

Protocol: tcp/ack

Numeric Protocol:

Source Port / ICMP Type

Operation: Equal to — Port #/Type: 80

Destination Port/ ICMP Code

Operation: Greater than — Port #/Code: 1023

Interfaces Settings

Interface: Specific

Name: tr0

Direction/Control

Routing: both local [✓ route]

Direction: both inbound [✓ outbound]

Log Control: Yes [✓ No]

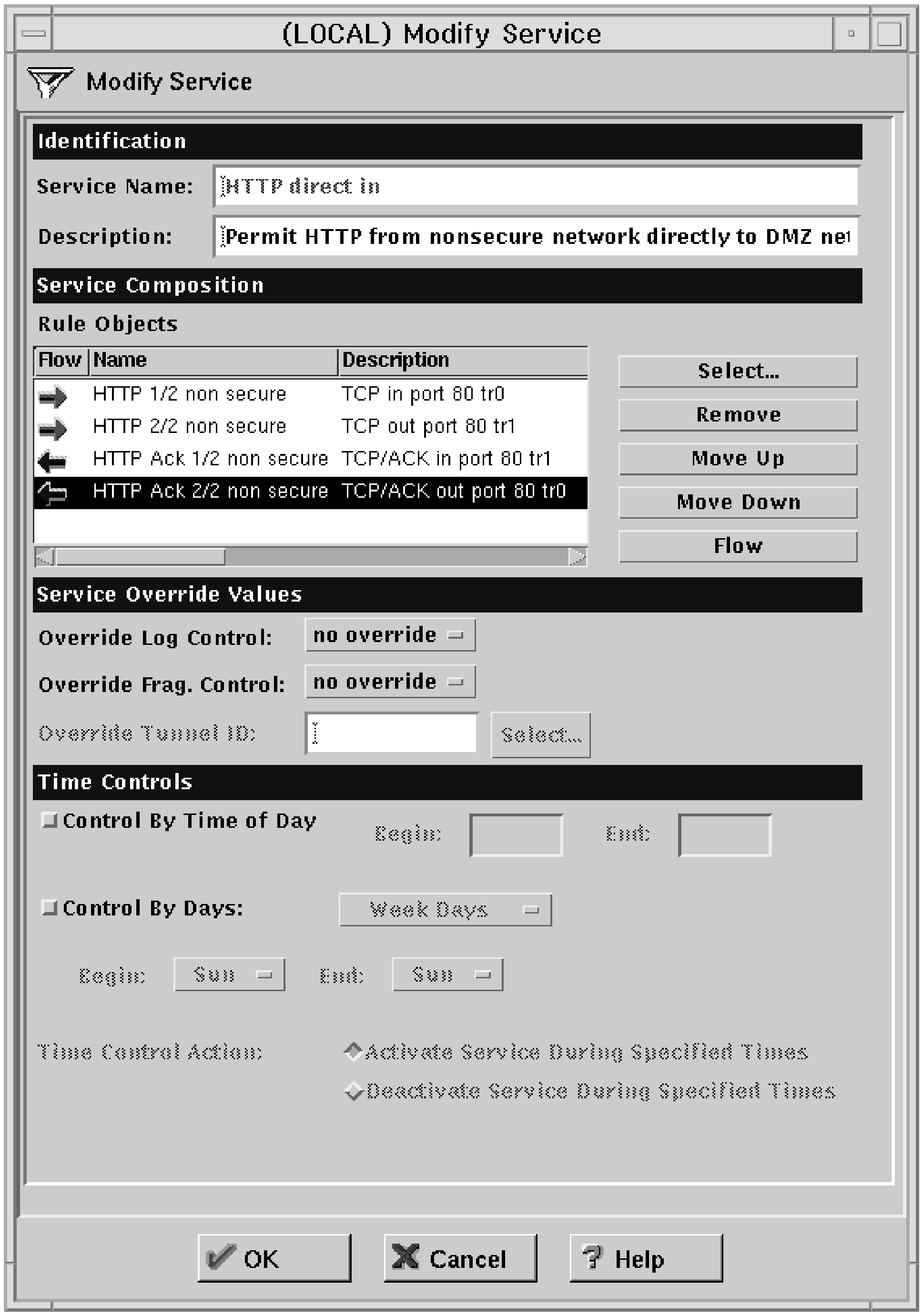
Frag. Control: Yes

Figure 396. HTTP Ack 2/2 non secure Rule

Once these four new rules were created, they became available and were immediately displayed in the Rules List window, as also we saw for the DB2 CAE rules in Figure 385 on page 408. Notice that the last two rules we created had the Protocol specification set to **tcp/ack** (see Figure 395 on page 420 and Figure 396). This is a security feature that we already explained in 8.5.2, “IP Filters Configuration for the DB2 Communication Protocol” on page 396. In this case it is to permit only TCP packets having the ACK flag set to flow from the Web server machine to the client, preventing that unwanted TCP session are established.

8.5.3.2 New Service Creation for HTTP

The next step was to add the four new rules just built for the HTTP protocol to a new service that we called *HTTP direct in*. To do this, we selected the **Services** item from the Configuration Client navigation tree, as shown in Figure 386 on page 409. To define this new service, we followed similar steps as those described in 8.5.2.2, “New Service Creation” on page 408. The result is shown in the following window:



(LOCAL) Modify Service

Modify Service

Identification

Service Name: HTTP direct in

Description: Permit HTTP from nonsecure network directly to DMZ net

Service Composition

Rule Objects

Flow	Name	Description
➡	HTTP 1/2 non secure	TCP in port 80 tr0
➡	HTTP 2/2 non secure	TCP out port 80 tr1
⬅	HTTP Ack 1/2 non secure	TCP/ACK in port 80 tr1
⬅	HTTP Ack 2/2 non secure	TCP/ACK out port 80 tr0

Service Override Values

Override Log Control: no override

Override Frag. Control: no override

Override Tunnel ID:

Time Controls

Control By Time of Day Begin: End:

Control By Days: Week Days

Begin: Sun End: Sun

Time Control Action: Activate Service During Specified Times / Deactivate Service During Specified Times

Figure 397. HTTP direct in Service Definition

Notice that we typed the Service Name and also we provided a description for the service in the Description field. Moreover we added the four rules in the same order as we created them.

The connection that we built had 95-client as source object and NT-server as destination object. For this reason, we had to select in the above window the two rules marked with the tcp/ack protocol specification and change their Flow property by clicking the **Flow** button. Left to right arrows changed immediately into right to left arrows. This operation was necessary, because for those two rules it was supposed that the IP traffic would flow from the destination to the source network object.

Then we clicked **OK** to accept the new service as we had defined it.

8.5.3.3 New Rules Creation for SSL

Similarly, we had to create new rules for the SSL protocol, in order to include them in a new service and then build a new connection for HTTP and SSL between the 95-client and the NT-server objects.

As usual, we needed to select **Rules** from the Configuration Client navigation tree (see Figure 375 on page 397). Then we highlighted **NEW** in the Rules List window (see Figure 376 on page 398) and we pressed the **Open:::** button.

The following list describes the four rules that we built for the SSL protocol:

1. HTTPS 1/2 non secure

   This rule was defined to permit the inbound IP routing using tr0 (which is the non-secure network interface) from a TCP port greater than 1023 to the TCP port 443.

2. HTTPS 2/2 non secure

   This rule was defined to permit the outbound IP routing using tr1 (which was the DMZ network interface) from a TCP port greater than 1023 to the TCP port 443.

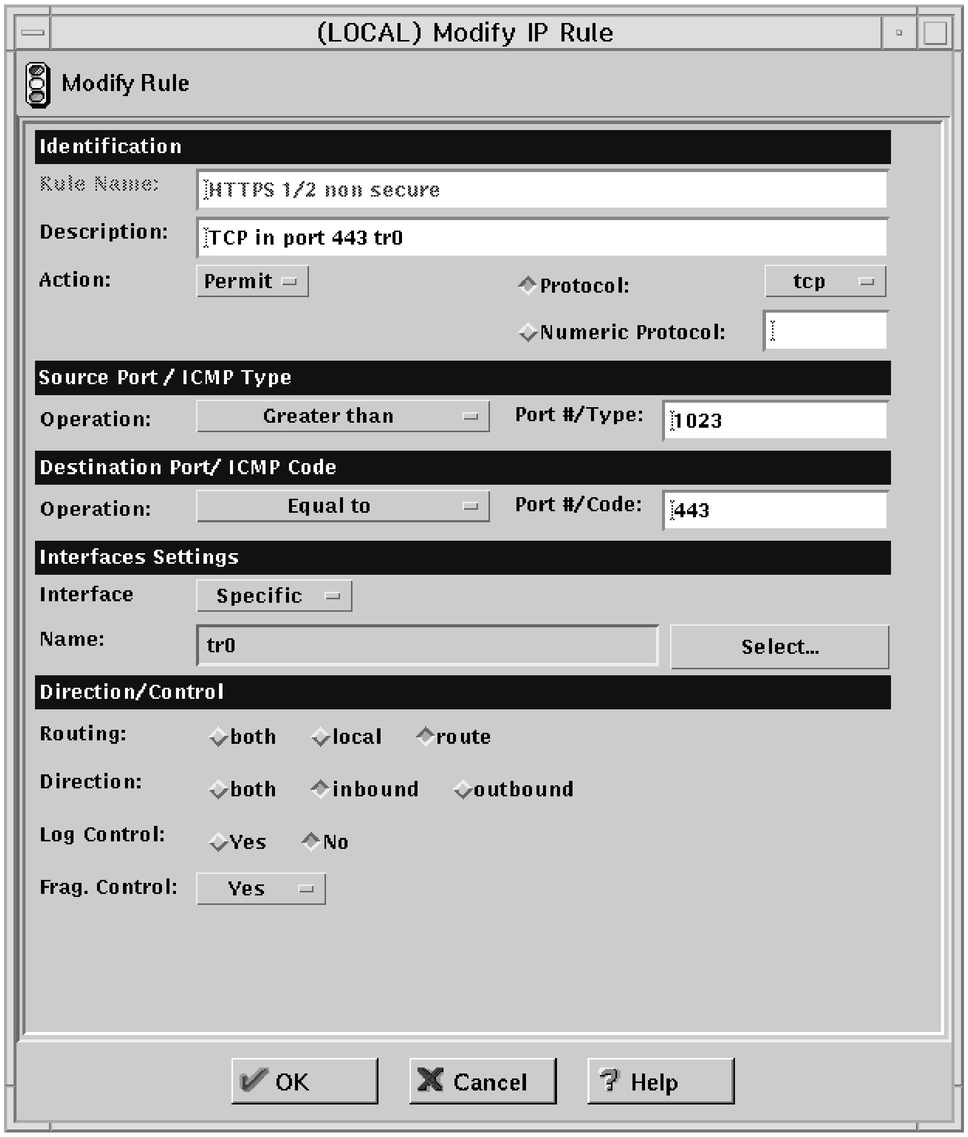
3. HTTPS Ack 1/2 non secure

   This rule was defined to permit the inbound IP routing using tr1 (which was the DMZ network interface) from the TCP port 443 to a TCP port greater than 1023.

4. HTTPS Ack 2/2 non secure

   This rule was defined to permit the outbound IP routing using tr0 (which was he non-secure network interface) from the TCP port 443 to a TCP port greater than 1023.

The screens from Figure 398 on page 424 through Figure 401 on page 427 show how we defined the rules for SSL listed above. As usual, you can consider these rules as an example to build the rules for your own environment.



Identification

Rule Name: HTTPS 1/2 non secure

Description: TCP in port 443 tr0

Action: Permit

[✓ Protocol:] tcp

Numeric Protocol:

Source Port / ICMP Type

Operation: Greater than

Port #/Type: 1023

Destination Port/ ICMP Code

Operation: Equal to

Port #/Code: 443

Interfaces Settings

Interface: Specific

Name: tr0

Direction/Control

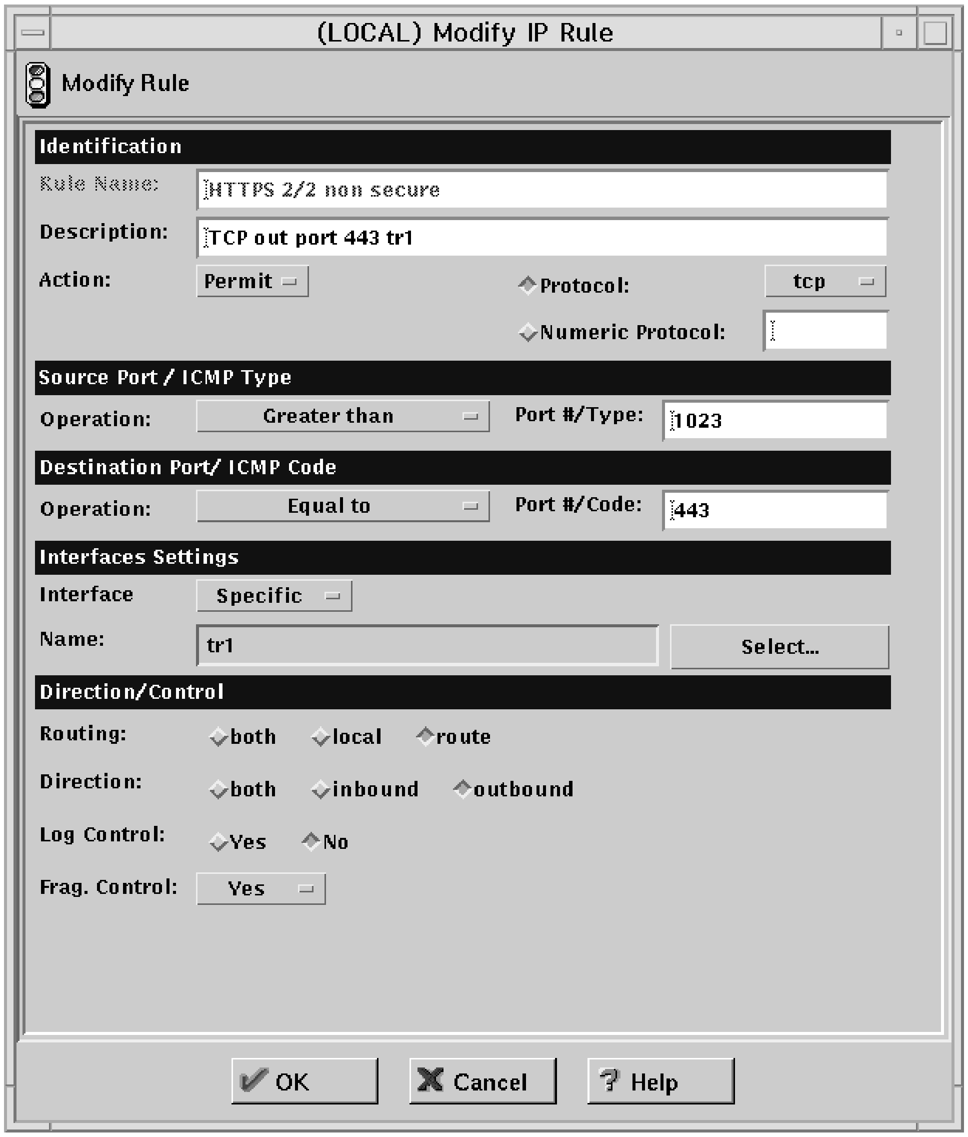
Routing: both local [✓ route]

Direction: both [✓ inbound] outbound

Log Control: Yes [✓ No]

Frag. Control: Yes

Figure 398. HTTPS 1/2 non secure Rule


(LOCAL) Modify IP Rule

Modify Rule

Identification

Rule Name:	HTTPS 2/2 non secure
Description:	TCP out port 443 tr1
Action:	Permit
[✓ Protocol:]	tcp
Numeric Protocol:	

Source Port / ICMP Type

Operation:	Greater than	Port #/Type:	1023

Destination Port/ ICMP Code

Operation:	Equal to	Port #/Code:	443

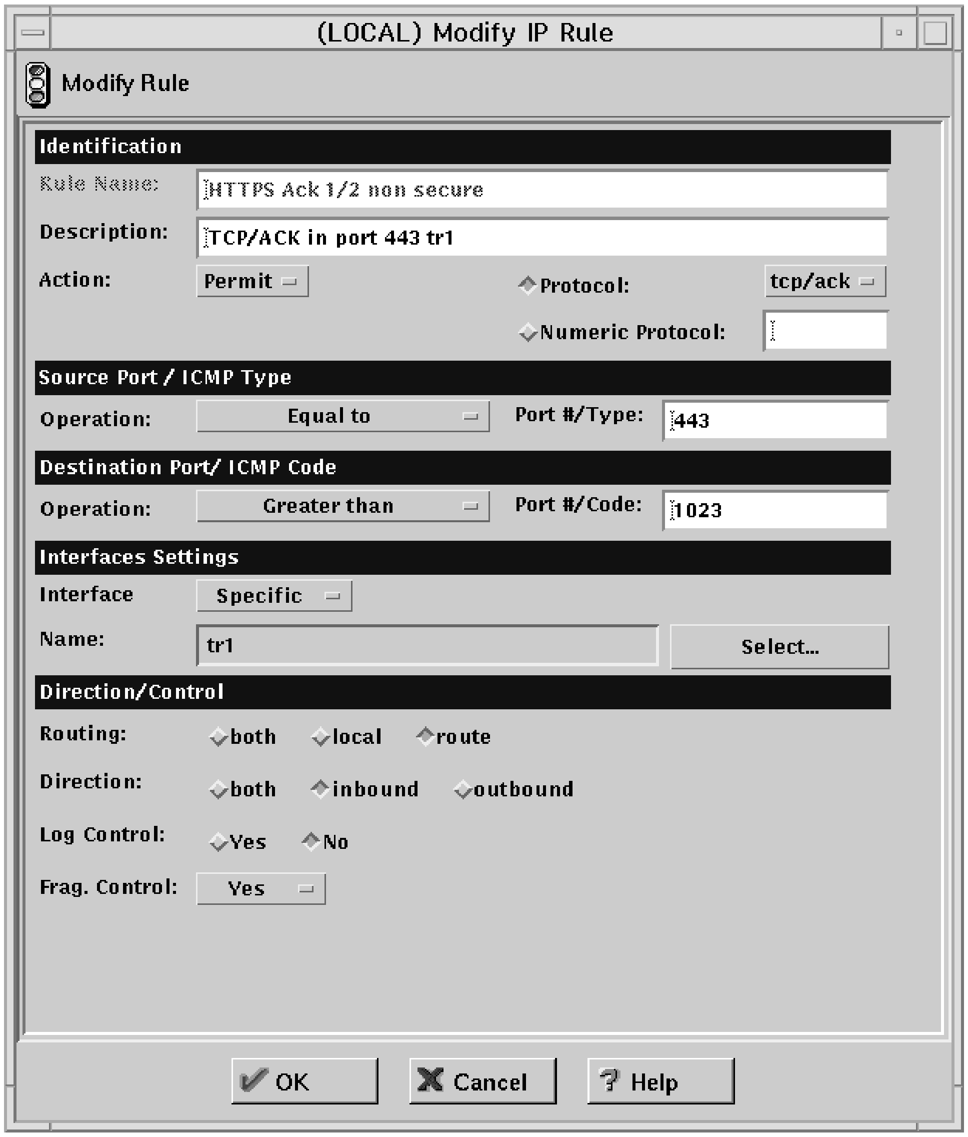
Interfaces Settings

Interface	Specific
Name:	tr1

Direction/Control

Routing:	both local [✓ route]
Direction:	both inbound [✓ outbound]
Log Control:	Yes [✓ No]
Frag. Control:	Yes

Figure 399. HTTPS 2/2 non secure Rule



Identification

Rule Name:	HTTPS Ack 1/2 non secure
Description:	TCP/ACK in port 443 tr1
Action:	Permit
[✓ Protocol:]	tcp/ack
Numeric Protocol:	

Source Port / ICMP Type

Operation:	Equal to
Port #/Type:	443

Destination Port/ ICMP Code

Operation:	Greater than
Port #/Code:	1023

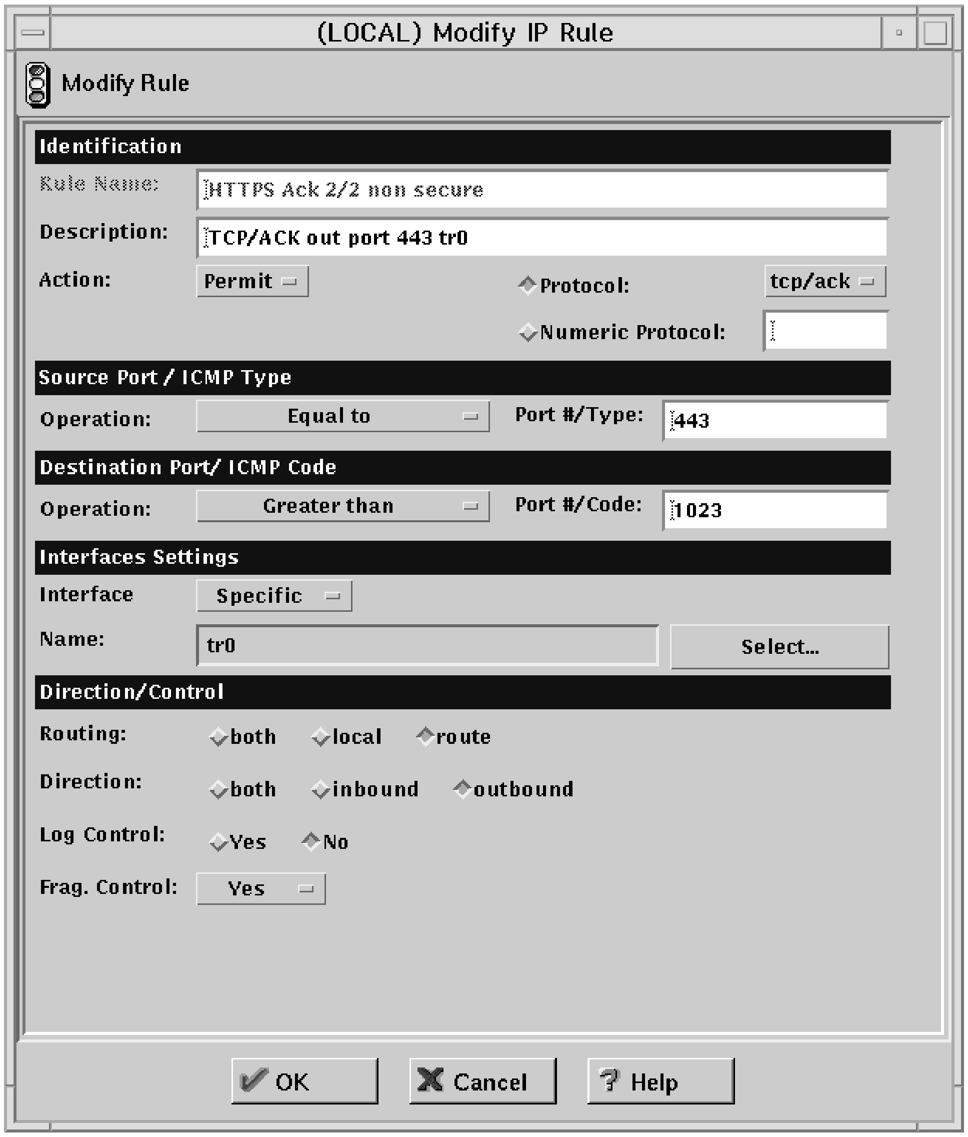
Interfaces Settings

Interface	Specific
Name:	tr1

Direction/Control

Routing:	both local [✓ route]
Direction:	both [✓ inbound] outbound
Log Control:	Yes [✓ No]
Frag. Control:	Yes

Figure 400. HTTPS Ack 1/2 non secure Rule



Identification

Rule Name: HTTPS Ack 2/2 non secure

Description: TCP/ACK out port 443 tr0

Action: Permit

[✓ Protocol:] tcp/ack

Numeric Protocol:

Source Port / ICMP Type

Operation: Equal to Port #/Type: 443

Destination Port/ ICMP Code

Operation: Greater than Port #/Code: 1023

Interfaces Settings

Interface: Specific

Name: tr0 Select...

Direction/Control

Routing: both local [✓ route]

Direction: both inbound [✓ outbound]

Log Control: Yes [✓ No]

Frag. Control: Yes

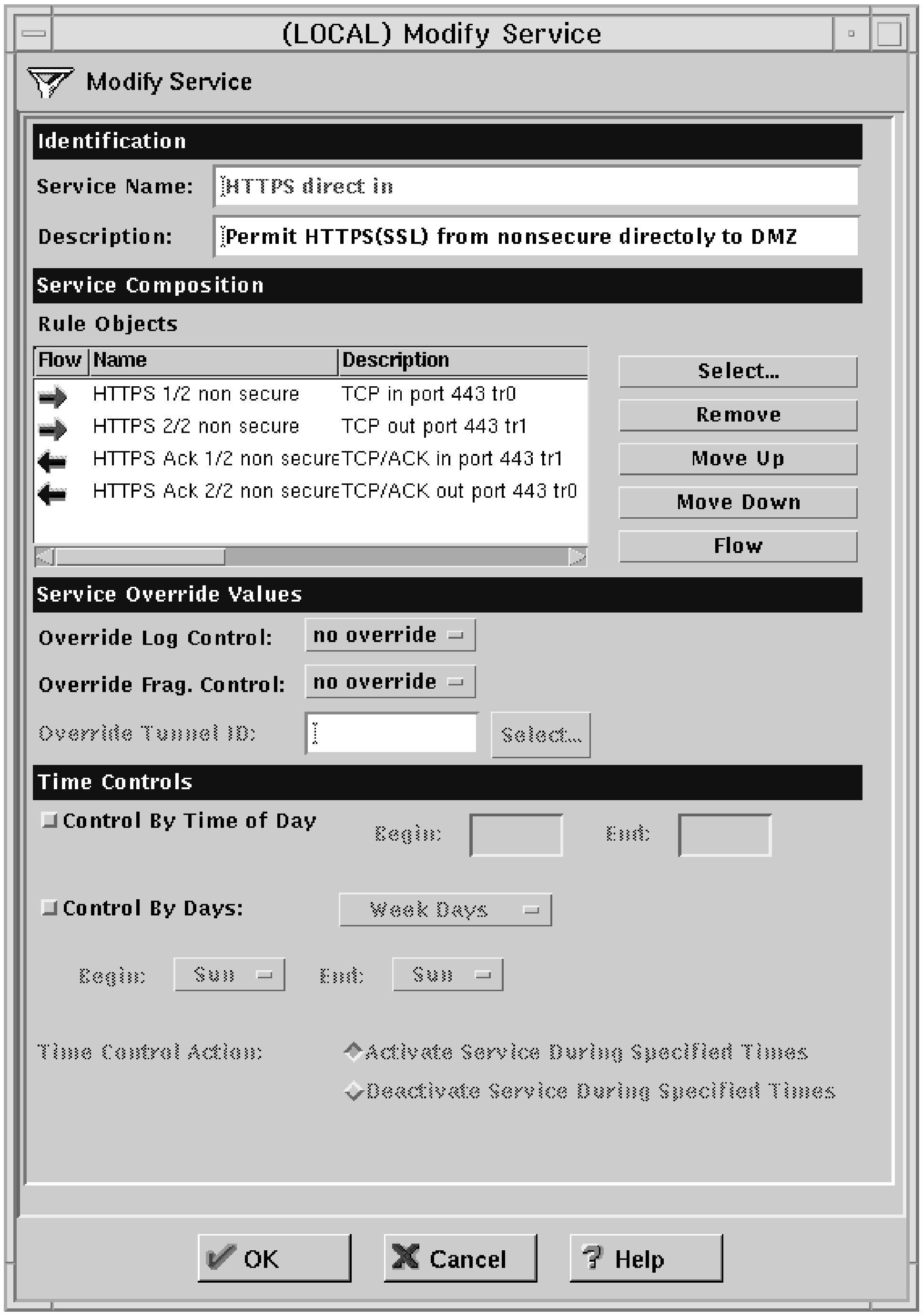
Figure 401. HTTPS Ack 2/2 non secure Rule

As soon as these four new rules were created, they became immediately available and appeared in the Rules List window (see Figure 376 on page 398).

You should have noticed that, building these new rules, we selected as usual **tcp/ack** as the Protocol specification for both the rules for which the IP traffic is supposed to flow from the NT-object to the 95-object (see Figure 400 on page 426 and Figure 401).

8.5.3.4 New Service Creation for SSL

The next step was to add the four new rules just built for SSL to a new service that we called `HTTPS direct in`. First of all, we selected the **Services** item from the Configuration Client navigation tree, as shown in Figure 386 on page 409. To define this new service, we followed similar steps as those described in 8.5.2.2, “New Service Creation” on page 408. The result is shown in the following window:



(LOCAL) Modify Service

Modify Service

Identification

Service Name:	HTTPS direct in
Description:	Permit HTTPS(SSL) from nonsecure directoly to DMZ

Service Composition

Rule Objects

Flow	Name	Description
→	HTTPS 1/2 non secure	TCP in port 443 tr0
→	HTTPS 2/2 non secure	TCP out port 443 tr1
←	HTTPS Ack 1/2 non secure	TCP/ACK in port 443 tr1
←	HTTPS Ack 2/2 non secure	TCP/ACK out port 443 tr0

Service Override Values

Override Log Control:	no override
Override Frag. Control:	no override
Override Tunnel ID:	

Time Controls

Control By Time of Day	Begin: End:
Control By Days:	Week Days
Begin: Sun	End: Sun
Time Control Action:	Activate Service During Specified Times
	Deactivate Service During Specified Times

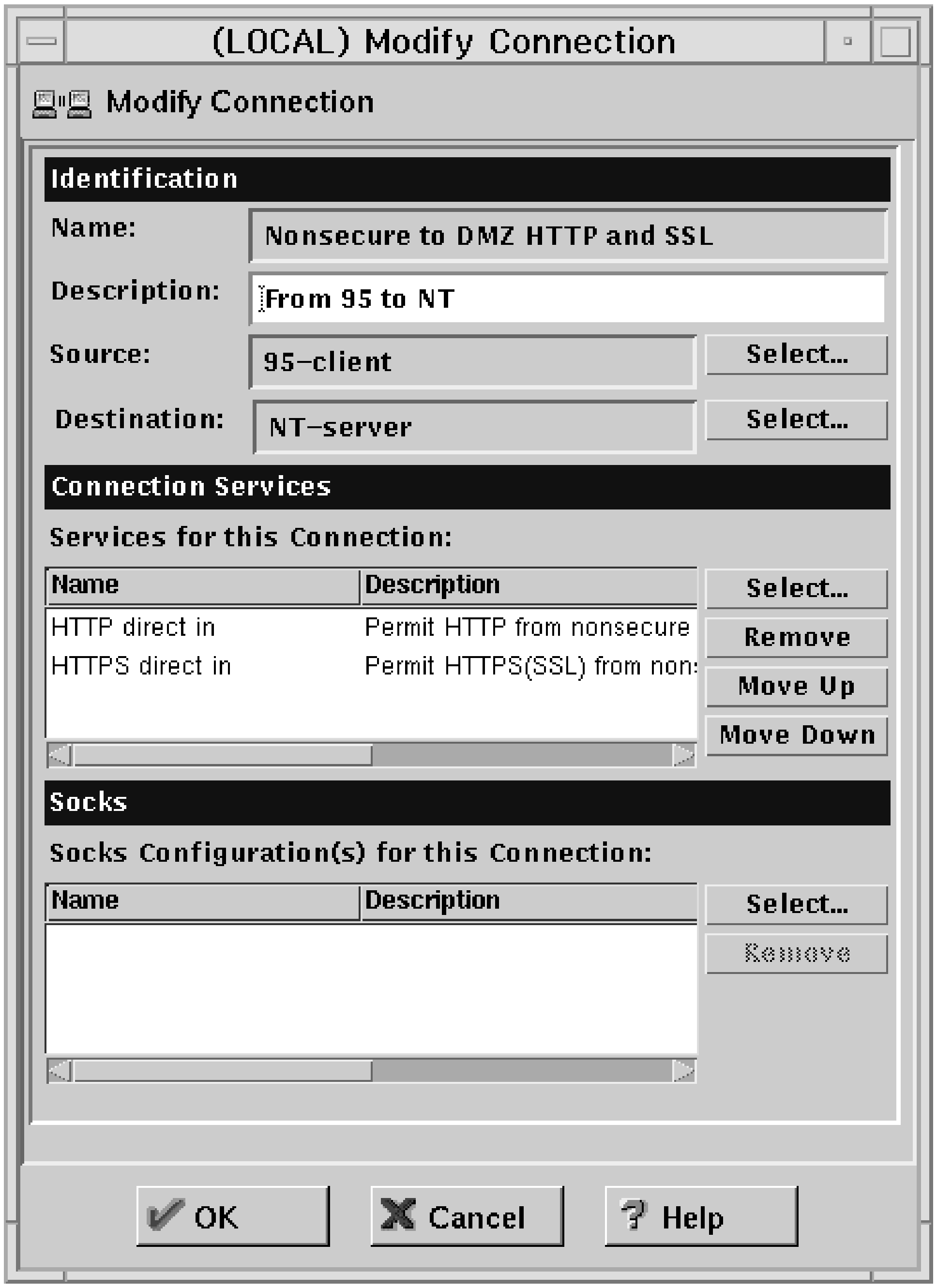
Figure 402. HTTPS direct in Service Definition

As usual, we entered the Service Name and also we filled the Description field with an appropriate description for the new service. The four new rules were added in the same order as they were created.

The connection that we built had 95-client as source object and NT-server as destination object. For this reason, we had to select in the above window the last two rules, which we had marked with the tcp/ack protocol specification, and change their Flow property by clicking the **Flow** button. Left to right arrows turned into right to left arrows. We did this, because for those two rules the IP traffic would flow from the destination to the source network object. After clicking **OK**, the new service became available.

8.5.3.5 Connection Configuration

In this section we describe how we added the two services created for HTTP and SSL to a new connection, that we named `Nonsecure to DMZ HTTP and SSL`. To do this, we selected **Traffic Control** and then **Connection Setup** in the Configuration Client navigation tree (see Figure 387 on page 411). The Add a Connection window appeared. We set Name to `Nonsecure to DMZ HTTP and SSL` and Description to `From 95 to NT`. Then we selected **95-client** as source network object and **NT-server** as destination network object. Also we selected **HTTP direct in** and **HTTPS direct in** as the two services for this connection. Then the Add a Connection window looked like the following screen:


(LOCAL) Modify Connection

Modify Connection

Identification

Name: Nonsecure to DMZ HTTP and SSL

Description: From 95 to NT

Source: 95-client

Destination: NT-server

Connection Services

Services for this Connection:

Name	Description
HTTP direct in	Permit HTTP from nonsecure
HTTPS direct in	Permit HTTPS(SSL) from non:

Socks

Socks Configuration(s) for this Connection:

Name	Description

Figure 403. Nonsecure to DMZ HTTP and SSL Connection

Then we simply pressed **OK**. This connection permits secure communication between the Web client and the Web server to pass through the firewall. Notice that when this connection is activated, both protocols, HTTP and SSL, will be allowed to pass through the firewall. Each time that you configure a firewall to let the SSL protocol pass through, you should also configure it to let HTTP pass through the firewall, since `https://` URLs are often activated as links from `http://` URLs.

In order for this connection to work, we needed to activate it. To do this, we selected **Traffic Control** and then **Connection Activation** in the Configuration Client navigation tree, as shown in Figure 390 on page 414. Once the

Connections List window appeared, we were able to activate the new connection we had built, **Nonsecure to DMZ HTTP and SSL**, in a similar way as we showed in Figure 391 on page 415.

To activate this connection, it was enough to press the **Activate:::** button. The Connection Activation window was brought up (see Figure 392 on page 416). We selected the item **Regenerate Connection Rules and Activate** and then we clicked **Execute**.

In order to run our three-tier application through the firewall, both connections we built, `DMZ to Secure DB2` and `Nonsecure to DMZ HTTP and SSL`, should be activated together, so that these should be the two connections present in the Connections List window when the button **Activate:::** is clicked.

8.6 Demonstration of the Three-Tier Application through the Firewall

We used the sample application named JDBCServlet provided by ServletExpress 1.0. You can find how to configure the JDBCServlet in 8.2.2, “Servlet Registration in ServletExpress” on page 366 and 8.3, “Testing the Application without Using the Firewall” on page 368. Of course, before testing our scenario, we had to set up Lotus Domino Go Webserver to use the SSL protocol. To enforce the security measures, we also enabled SSL client authentication. In order to see how to configure the SSL protocol on Lotus Domino Go Webserver, refer to 3.1.2, “Lotus Domino Go Webserver SSL Setup” on page 71. You can also read 3.1.4, “SSL Client Authentication” on page 84 to see how to enable client authentication. To run the JDBCServlet over SSL, it was enough to point the Web browser of our client machine to the following URL: https://192.168.50.2/ServletExpress/resources/JDBCServlet/JDBCServletForm.html.

We got the query form, which we used to send DB2 requests from the Internet, where we supposed our client machine was located, to the Web server machine, located in the DMZ. The following window shows the query form that we obtained using the SSL protocol and the security measures provided by ServletExpress:

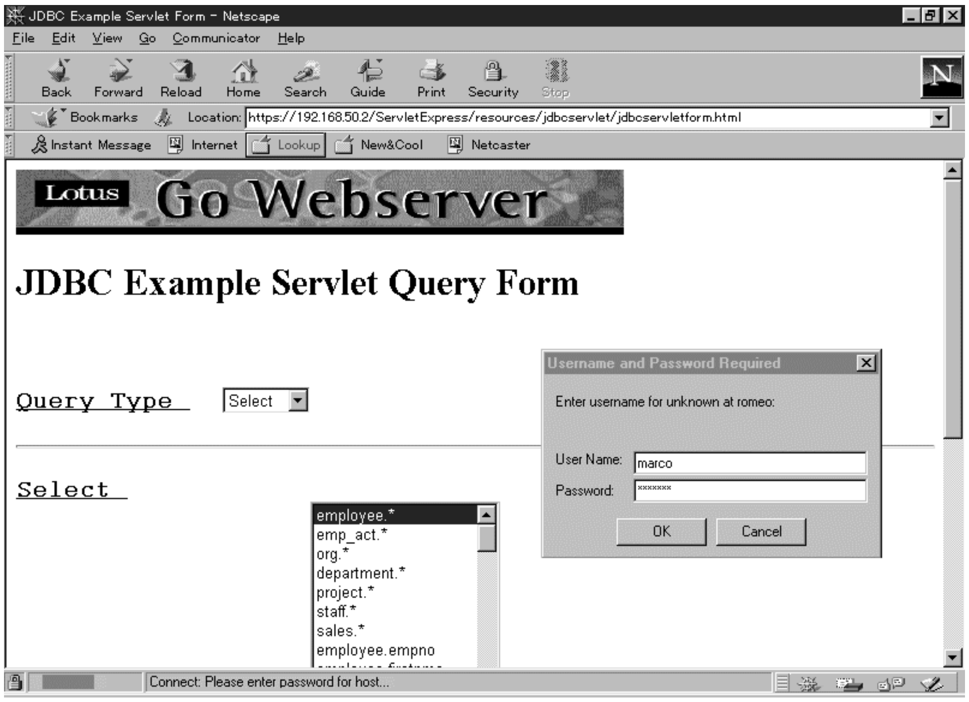
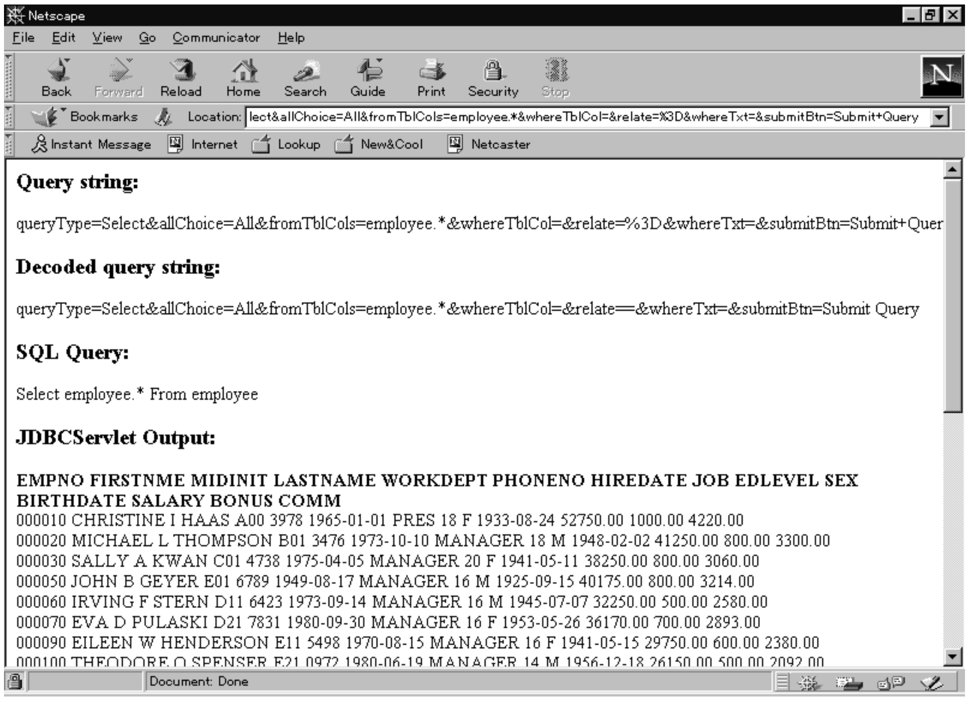

JDBC Example Servlet Form - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: https://192.168.50.2/ServletExpress/resources/jdbcservlet/jdbcservletform.html
Instant Message Internet Lookup New&Cool Netcaster
Lotus Go Webserver
JDBC Example Servlet Query Form
Query Type Select
Select
employee.*
emp_act.*
org.*
department.*
project.*
staff.*
sales.*
employee.empno
Username and Password Required
Enter username for unknown at romeo:
User Name: marco
Password: *******
OK Cancel
Connect: Please enter password for host...

Figure 404. Query Form through the SSL Protocol with ServletExpress Access Restriction

If you compare this picture with Figure 365 on page 384, you will notice that the SSL protocol this time is used since the lock represented at the bottom-left corner of the window is closed now, but it was opened when we had tested the application on HTTP. The closed lock is the security icon that the Netscape browser uses to inform the user that an SSL connection is in place. Notice that we had to enter the user ID and password for the user marco, since access to the JDBCServlet had been restricted to this user through the security measures provided by ServletExpress.

The following window is the result of our query. It demonstrates that our three-tier application using SSL was able to pass through the firewall:



```
Location: lect&allChoice=All&fromTblCols=employee.*&whereTblCol=&relate=%3D&whereTxt=&submitBtn=Submit+Query
```

Query string:

```
queryType=Select&allChoice=All&fromTblCols=employee.*&whereTblCol=&relate=%3D&whereTxt=&submitBtn=Submit+Quer
```

Decoded query string:

```
queryType=Select&allChoice=All&fromTblCols=employee.*&whereTblCol=&relate==&whereTxt=&submitBtn=Submit Query
```

SQL Query:

```
Select employee.* From employee
```

JDBCServlet Output:

```
EMPNO FIRSTNME MIDINIT LASTNAME WORKDEPT PHONENO HIREDATE JOB EDLEVEL SEX
BIRTHDATE SALARY BONUS COMM
000010 CHRISTINE I HAAS A00 3978 1965-01-01 PRES 18 F 1933-08-24 52750.00 1000.00 4220.00
000020 MICHAEL L THOMPSON B01 3476 1973-10-10 MANAGER 18 M 1948-02-02 41250.00 800.00 3300.00
000030 SALLY A KWAN C01 4738 1975-04-05 MANAGER 20 F 1941-05-11 38250.00 800.00 3060.00
000050 JOHN B GEYER E01 6789 1949-08-17 MANAGER 16 M 1925-09-15 40175.00 800.00 3214.00
000060 IRVING F STERN D11 6423 1973-09-14 MANAGER 16 M 1945-07-07 32250.00 500.00 2580.00
000070 EVA D PULASKI D21 7831 1980-09-30 MANAGER 16 F 1953-05-26 36170.00 700.00 2893.00
000090 EILEEN W HENDERSON E11 5498 1970-08-15 MANAGER 16 F 1941-05-15 29750.00 600.00 2380.00
000100 THEODORE Q SPENSER E21 0972 1980-06-19 MANAGER 14 M 1956-12-18 26150.00 500.00 2092.00
```

Document: Done

Figure 405. Results

This time too the lock icon informed us that an SSL connection was in place (see also the difference with Figure 352 on page 371), so that we made sure that all the communication between the client and the server was encrypted by the SSL protocol.

Chapter 9. IIOP in Firewall-Protected Network Environments

In this chapter we will show how to grant security for client/server applications running in an intranet. The IBM Network Computing Framework calls this kind of application a two-tier model. In particular we will focus on applications using the IIOP protocol.

The intranet will be divided by a firewall into two subnetworks:

1. A high-level security subnetwork, corresponding to the client machine
2. A low-level security subnetwork, corresponding to the server machine

The following figure shows a graphical representation of our test environment:

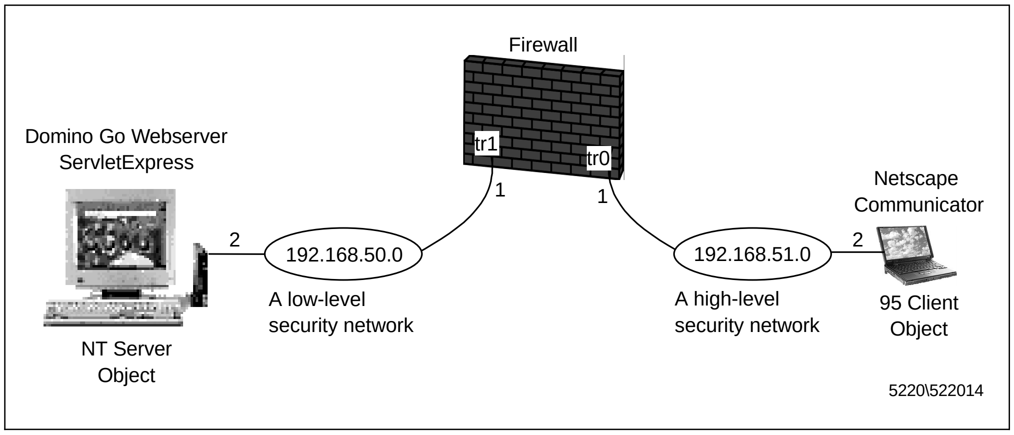


Figure 406. Two-Tier Test Environment

Our application will distribute over the Internet Inter-ORB Protocol (IIOP) using ServletExpress CORBA Support. We will show how to build the firewall within the intranet and what the implications are with IIOP.

Note: The considerations that you read in this chapter are related to our experience with ServletExpress Version 1.0 beta 3. As we mentioned in Chapter 4, "IBM WebSphere Application Server Security" on page 121, the servlet engine and CORBA Support piece provided by ServletExpress are now incorporated in the recently announced IBM WebSphere Application Server Version 1.0. Notice that the directory structure changed for the servlet engine and for the CORBA Support piece. For the rest, you can safely refer to what we have written here about ServletExpress and refer the same experience to WebSphere Application Server Version 1.0. We stayed in touch with the development team all through the writing of this redbook and were informed of one single major change, which we will describe in 9.7.2, "Client/Server Communication through an HTTP Proxy Server" on page 473 and 9.8.2, "Client/Server Communication through a SOCKS Server" on page 482.

It would be good to have a general description of the IIOP protocol and ServletExpress CORBA Support before continuing the description of our scenario.

9.1 What Is IIOP?

IIOP is a communications protocol based on the Common Object Request Broker Architecture (CORBA) specifications provided by the Object Management Group (OMG). You will find further information about the Object Management Group at the OMG home page http://www.omg.org/.

An application is enabled to use distributed objects by using the Object Request Broker (ORB). The ORB transparently forwards remote object requests (for example, a method invocation) to the appropriate server objects, dispatches the requests and returns the results (for example, a method return value).

IIOP is the standard protocol supported by the IBM Network Computing Framework for remote method call support. It allows client communication with Business Object Framework objects on a remote server. By using the CORBA/IIOP protocol, a Java applet running in a client machine can communicate with a servlet running in the Web server and supporting IIOP as well. IIOP is much more efficient than communicating over HTTP, since IIOP may reuse a single connection for multiple requests. That is the reason why the applet and the servlet can exchange method calls in both directions, along the same connection.

9.2 WebSphere Application Server CORBA Support

WebSphere Application Server CORBA Support is a CORBA-compliant ORB and an extensive feature set that enables the development of Web-based Java applications. As we have mentioned several times, we worked with the beta 3 release of WebSphere Application Server, that still carried the name of ServletExpress. For this reason, for the rest of this section, we will continue referring to the CORBA Support piece as ServletExpress CORBA Support.

9.2.1 Configuration

ServletExpress CORBA Support is automatically enabled as soon as you install and configure ServletExpress properly, as indicated in 7.3, “IBM ServletExpress 1.0” on page 311. However, some modifications are required in your computing environment when you are going to develop ServletExpress CORBA applications.

First of all, you should define a system environment variable named JAVA_HOME and set its value to the directory containing bin/javac.exe. If you followed the default installation for the JDK, that directory is usually the root directory of the Java Development Kit. In our Windows NT environment, it was D:\jdk1.1.6, so we opened the System Properties window from the Control Panel, defined a new system environment variable named JAVA_HOME and set its value to `D:\jdk1.1.6`.

Furthermore, during the installation of ServletExpress, two CORBA Support Java archives, named ibmjbrt.jar and ibmjbde.jar, were placed in the directory tree *servexp_root*\web\classes (where *servexp_root* is the ServletExpress installation directory, in our case D:\ServletExpress). The run-time archive ibmjbrt.jar, contains classes for all CORBA Support features, less the compilers. It is the development archive ibmjbde.jar that contains classes for all the compilers. The CLASSPATH system environment variable must now include these two archives, in order for ServletExpress CORBA Support to work correctly. The full path *servexp_root*\web\classes for these two Java archives must be included in the

CLASSPATH system environment variable as well. The AccountsUI applet does not compile if you do not add that directory too.

Considering all the modifications that we have already done to the CLASSPATH variable to enable the servlet engine provided by ServletExpress (see Figure 302 on page 320) the CLASSPATH current value must include the following:

```
.;
D:\jdk1.1.6\lib\classes.zip;
D:\ServletExpress\lib\servexp.jar;
D:\ServletExpress\lib\jst.jar;
D:\ServletExpress\lib\jsdk.jar;
D:\ServletExpress\lib\x509v1.jar;
D:\ServletExpress\web\classes\ibmjbrt.jar;
D:\ServletExpress\web\classes\ibmjbde.jar;
D:\ServletExpress\web\classes;
D:\WWW\CGI-Bin\icsclass.zip;
D:\ServletExpress\servlets;
```

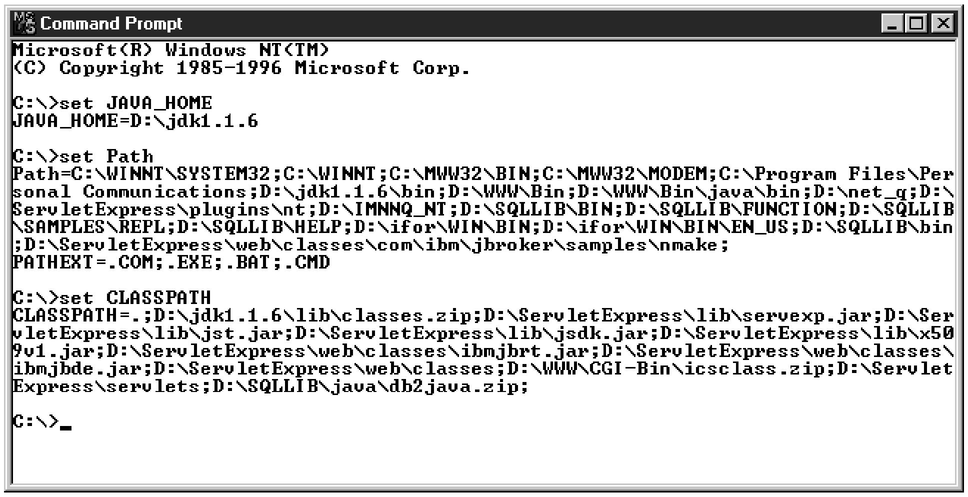
Figure 407. Current Settings for the CLASSPATH Environment Variable

More about the CLASSPATH System Environment Variable

The CORBA Support development team recommends that you include the two Java archives ibmjbrt.jar and ibmjbde.jar, plus the full path *servexp_root*\web\classes for these two files, in the CLASSPATH system environment variable, but only during development (for example, during building of the sample applications) or on development machines. There are cases where leaving those two files and the class directory in the CLASSPATH at all times actually causes problems, and we explain the reason of this in 9.2.2, "Local Applets Security Considerations" on page 438. So both the Java archives ibmjbrt.jar and ibmjbde.jar, plus the classes directory should be removed from the CLASSPATH at deployment time, or on deployment machines.

If you have installed and configured ServletExpress properly, you should have already appended the JDK bin directory (in our case D:\jdk1.1.6\bin) to the Path variable of your computing environment. No new action is required on the Path variable to enable ServletExpress CORBA Support. It is convenient, however, that the full path to the file nmake.exe is added to the value of Path. This program comes with the installation files of ServletExpress CORBA Support and it is very useful for fast access to that executable when you are developing applications. The command line utility `nmake` is automatically installed in the directory *servexp_root*\web\classes\com\ibm\jbroker\samples\nmake, so it is enough that you add this directory to the current value of the Path variable.

By entering the `set` command at an MS-DOS command prompt followed by the name of the environment variable, you can verify whether that variable is set correctly. The following figure summarizes all the system environment settings that were necessary to use the ServletExpress CORBA Support on our Windows NT platform:



```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>set JAVA_HOME
JAVA_HOME=D:\jdk1.1.6

C:\>set Path
Path=C:\WINNT\SYSTEM32;C:\WINNT;C:\MWW32\BIN;C:\MWW32\MODEM;C:\Program Files\Per
sonal Communications;D:\jdk1.1.6\bin;D:\WWW\Bin;D:\WWW\Bin\java\bin;D:\net_q;D:\
ServletExpress\plugins\nt;D:\IMNNQ_NT;D:\SQLLIB\BIN;D:\SQLLIB\FUNCTION;D:\SQLLIB
\SAMPLES\REPL;D:\SQLLIB\HELP;D:\ifor\WIN\BIN;D:\ifor\WIN\BIN\EN_US;D:\SQLLIB\bin
;D:\ServletExpress\web\classes\com\ibm\jbroker\samples\nmake;
PATHEXT=.COM;.EXE;.BAT;.CMD

C:\>set CLASSPATH
CLASSPATH=.;D:\jdk1.1.6\lib\classes.zip;D:\ServletExpress\lib\servexp.jar;D:\Ser
vletExpress\lib\jst.jar;D:\ServletExpress\lib\jsdk.jar;D:\ServletExpress\lib\x50
9v1.jar;D:\ServletExpress\web\classes\ibmjbrt.jar;D:\ServletExpress\web\classes\
ibmjbde.jar;D:\ServletExpress\web\classes;D:\WWW\CGI-Bin\icsclass.zip;D:\Servlet
Express\servlets;D:\SQLLIB\java\db2java.zip;

C:\>_
```

Figure 408. Current Values for CLASSPATH, Path and JAVA_HOME Variables

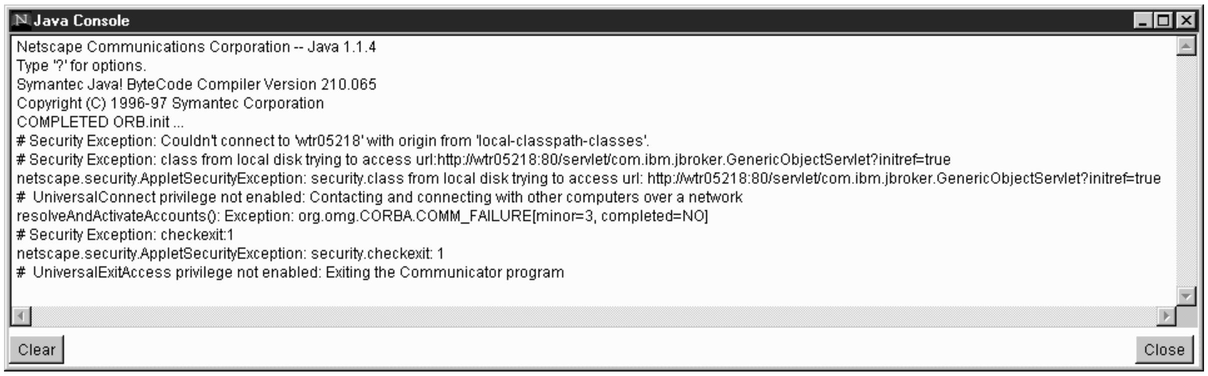
9.2.2 Local Applets Security Considerations

We want to mention in this section a problem we encountered while implementing our two-tier application. Sometimes, for testing purposes, one installs the Web server and ServletExpress on the client machine and then uses the client's browser to connect to the Web server. This is what we did. But similar configurations can easily generate errors. In fact, in order to configure Lotus Domino Go Webserver and ServletExpress to run on a machine, you need to make some changes to the CLASSPATH system environment variable. Each time you modify the CLASSPATH environment variable on a computer that plays also the role of the client, you must be careful because that variable is checked by Netscape Communicator when it needs to load an applet from the network. If an applet with the same file name and the same package name is found in the local CLASSPATH, that local applet is loaded using the file protocol instead of the remote applet. Moreover the SecurityManager implemented by Netscape Communicator does not allow an applet loaded with the file protocol to connect a machine in the network by using an URL connection with the HTTP protocol. For this reason a SecurityException is thrown.

After installing ServletExpress on the client machine, we included the directory D:\ServletExpress\web\classes in the CLASSPATH variable and, exactly under that directory, Netscape's SecurityManager found the applet com.ibm.jbroker.samples.account.AccountsUI, having the same class file name and package name as the applet on the Web server. For this reason, the local applet was downloaded in place of the remote one, but as soon as it tried to connect the Web server the following error message appeared on the bottom of the browser window:

```
Applet com.ibm.jbroker.samples.account.AccountsUI can't start: ERROR
```

and the Java Console, that you can open from the Communicator pull-down menu, immediately registered an unexpected SecurityException:



```
Netscape Communications Corporation -- Java 1.1.4
Type '?' for options.
Symantec Java! ByteCode Compiler Version 210.065
Copyright (C) 1996-97 Symantec Corporation
COMPLETED ORB.init ...
# Security Exception: Couldn't connect to 'wtr05218' with origin from 'local-classpath-classes'.
# Security Exception: class from local disk trying to access url:http://wtr05218:80/servlet/com.ibm.jbroker.GenericObjectServlet?initref=true
netscape.security.AppletSecurityException: security.class from local disk trying to access url: http://wtr05218:80/servlet/com.ibm.jbroker.GenericObjectServlet?initref=true
#  UniversalConnect privilege not enabled: Contacting and connecting with other computers over a network
resolveAndActivateAccounts(): Exception: org.omg.CORBA.COMM_FAILURE[minor=3, completed=NO]
# Security Exception: checkexit:1
netscape.security.AppletSecurityException: security.checkexit: 1
#  UniversalExitAccess privilege not enabled: Exiting the Communicator program
```

Figure 409. Java Console for Netscape Communicator

The SecurityManager reminded us that an applet from local disk cannot access the network through the HTTP protocol. This problem was caused by the CLASSPATH variable configuration, since it included the directory D:\ServletExpress\web\classes too. It is worth knowing that similar errors can be generated, for example when you use for testing purposes the same machine as both client and server. In this case, you should include the above directory in the CLASSPATH variable when you develop your CORBA applications (for example, the AccountsUI applet does not compile if that directory is not included) and remove it when you run that application.

9.2.3 NCF Standard and Enhanced IIOP Modes

IBM Network Computing Framework provides two possibilities to add distribution to Java and Web-centric applications using the IIOP protocol:

1. A standard IIOP mode, which ensures inter-operability between different ORBs through the Internet distributing objects to remote servers. This is accomplished by passing objects by reference.
2. An enhanced IIOP mode, which ensures that a client can use a local copy of a remote object. This is accomplished by passing objects by value rather than by reference. In this way the overhead of object interactions involving remote method invocations is no longer incurred.

9.2.4 IIOP over HTTP

The communication between server and client in ServletExpress CORBA Support can use two connection methods:

1. Direct IIOP connection (independent on the Web server)
2. IIOP over HTTP (through the Web server)

The flow of both is described in the following list:

1. The applet is downloaded from a Web server machine to a Java-enabled Web browser of a client machine by using a typical HTML page.
2. The applet automatically activates a client ORB within itself.
3. The applet sends an HTTP request to the Web server in order to load and run the servlet.
4. The servlet will then activate the server ORB inside itself.
5. The client ORB controls IIOP communication with the server.

a. IIOP communication between client and server uses a long-running socket connection directly from the client to the object server port if ServletExpress CORBA Support can successfully activate a direct IIOP connection. This socket is used for the entire period the client is connected to the server.

b. IIOP communication between client and server uses short-running socket connections to the Web server port (usually port 80) if ServletExpress CORBA Support uses IIOP over HTTP. In this mode, a new connection is used for each IIOP request encoded over HTTP, and that connection is closed at the end of each reply from the server.

Once the IIOP communication is established, object activations and method calls on those objects may occur back and forth using either direct IIOP connection or IIOP over HTTP, depending on what type of connection ServletExpress CORBA Support activated.

Notice that it is the client ORB that decides whether to use a direct IIOP connection or revert to IIOP over HTTP (see 5 on page 439). The client will use the best quality of service available: it tries to make a direct IIOP connection first and, if that doesn't work, it uses IIOP over HTTP.

The following picture represents the flow, as it was described in the above list, when IIOP is carried over HTTP.

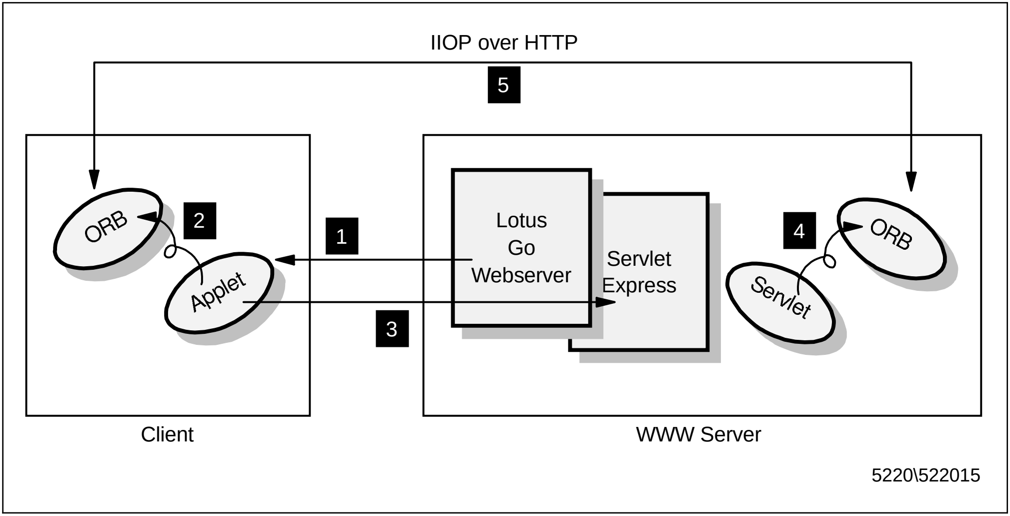
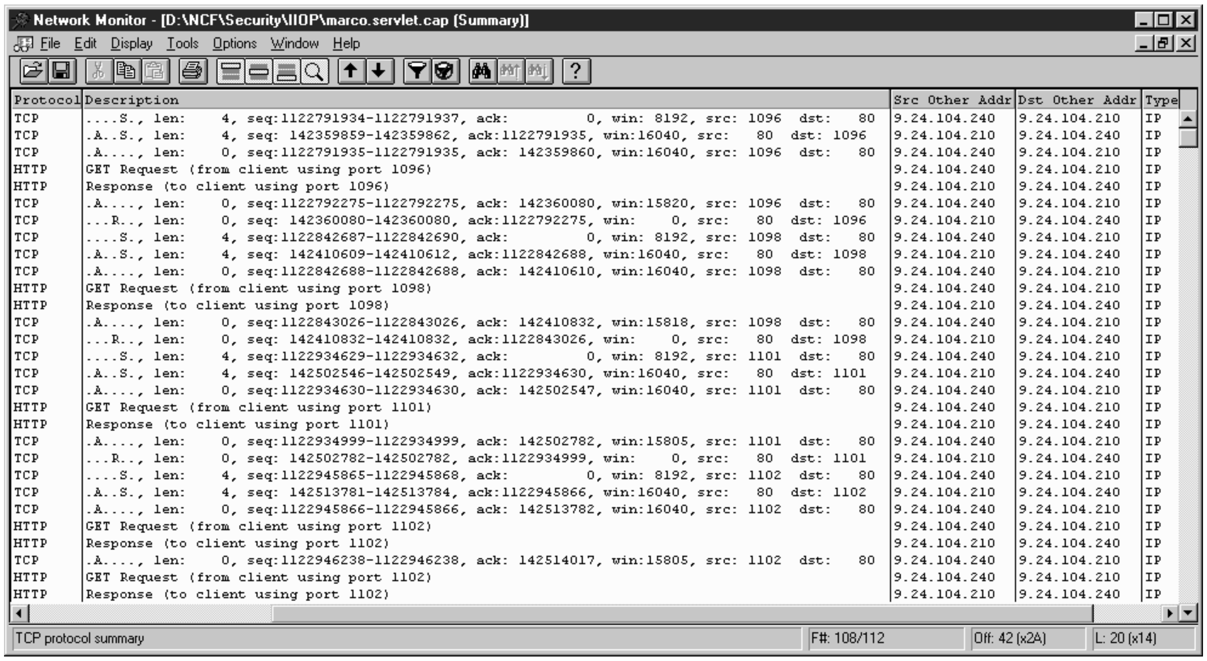


Figure 410. IIOP over HTTP Communication

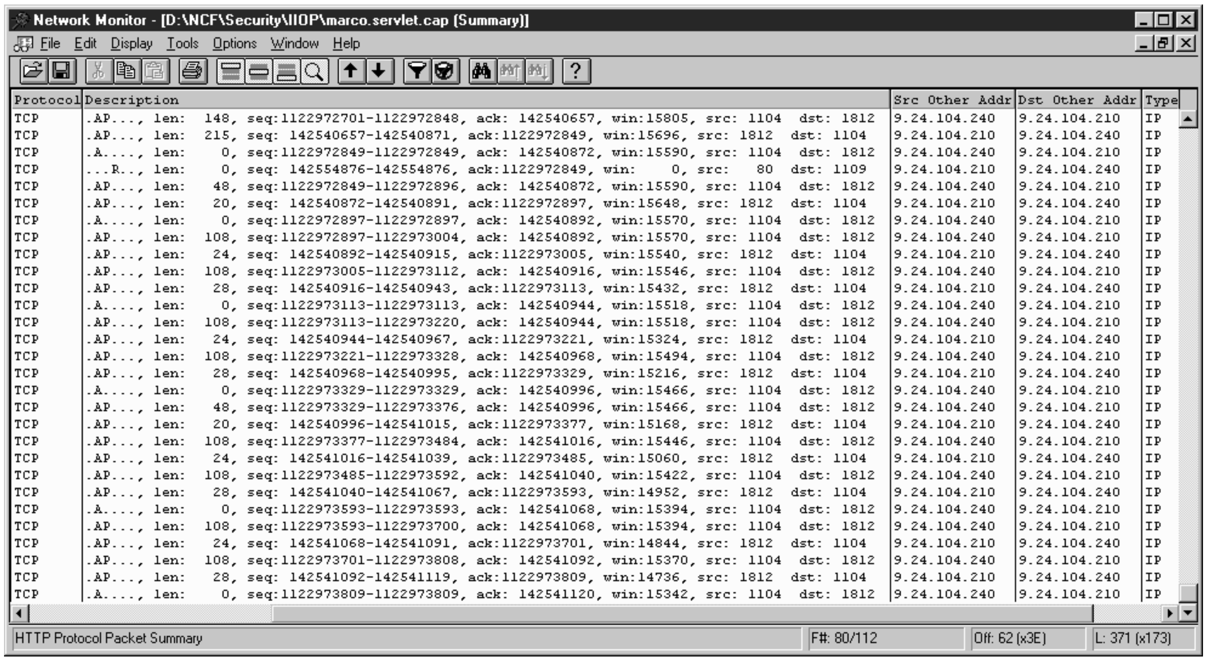
In one of the tests that we did without the firewall, we captured the IP trace on the Web server, that was a Windows NT Server 4.0 machine. The tool that we used to capture the IP trace was Systems Management Server (SMS) Network Monitor Version 4.00, that can be installed on Windows NT Server 4.0. This test was very meaningful. It demonstrated that the client/server communication uses HTTP to download the applet and to activate the servlet (see 1 on page 439 and 3 on page 439). The following window shows that the HTTP port used on the server in this communication is the standard TCP port 80:


Network Monitor - [D:\NCF\Security\IIOP\marco.servlet.cap (Summary)]
File Edit Display Tools Options Window Help

```
Protocol Description                                                                    Src Other Addr Dst Other Addr Type
TCP      ....S., len:    4, seq:1122791934-1122791937, ack:         0, win: 8192, src: 1096  dst:   80 9.24.104.240   9.24.104.210   IP
TCP      .A..S., len:    4, seq: 142359859-142359862, ack:1122791935, win:16040, src:   80  dst: 1096  9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122791935-1122791935, ack: 142359860, win:16040, src: 1096  dst:   80 9.24.104.240   9.24.104.210   IP
HTTP     GET Request (from client using port 1096)                                          9.24.104.240   9.24.104.210   IP
HTTP     Response (to client using port 1096)                                               9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122792275-1122792275, ack: 142360080, win:15820, src: 1096  dst:   80 9.24.104.240   9.24.104.210   IP
TCP      ...R.., len:    0, seq: 142360080-142360080, ack:1122792275, win:    0, src:   80  dst: 1096  9.24.104.210   9.24.104.240   IP
TCP      ....S., len:    4, seq:1122842687-1122842690, ack:         0, win: 8192, src: 1098  dst:   80 9.24.104.240   9.24.104.210   IP
TCP      .A..S., len:    4, seq: 142410609-142410612, ack:1122842688, win:16040, src:   80  dst: 1098  9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122842688-1122842688, ack: 142410610, win:16040, src: 1098  dst:   80 9.24.104.240   9.24.104.210   IP
HTTP     GET Request (from client using port 1098)                                          9.24.104.240   9.24.104.210   IP
HTTP     Response (to client using port 1098)                                               9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122843026-1122843026, ack: 142410832, win:15818, src: 1098  dst:   80 9.24.104.240   9.24.104.210   IP
TCP      ...R.., len:    0, seq: 142410832-142410832, ack:1122843026, win:    0, src:   80  dst: 1098  9.24.104.210   9.24.104.240   IP
TCP      ....S., len:    4, seq:1122934629-1122934632, ack:         0, win: 8192, src: 1101  dst:   80 9.24.104.240   9.24.104.210   IP
TCP      .A..S., len:    4, seq: 142502546-142502549, ack:1122934630, win:16040, src:   80  dst: 1101  9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122934630-1122934630, ack: 142502547, win:16040, src: 1101  dst:   80 9.24.104.240   9.24.104.210   IP
HTTP     GET Request (from client using port 1101)                                          9.24.104.240   9.24.104.210   IP
HTTP     Response (to client using port 1101)                                               9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122934999-1122934999, ack: 142502782, win:15805, src: 1101  dst:   80 9.24.104.240   9.24.104.210   IP
TCP      ...R.., len:    0, seq: 142502782-142502782, ack:1122934999, win:    0, src:   80  dst: 1101  9.24.104.210   9.24.104.240   IP
TCP      ....S., len:    4, seq:1122945865-1122945868, ack:         0, win: 8192, src: 1102  dst:   80 9.24.104.240   9.24.104.210   IP
TCP      .A..S., len:    4, seq: 142513781-142513784, ack:1122945866, win:16040, src:   80  dst: 1102  9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122945866-1122945866, ack: 142513782, win:16040, src: 1102  dst:   80 9.24.104.240   9.24.104.210   IP
HTTP     GET Request (from client using port 1102)                                          9.24.104.240   9.24.104.210   IP
HTTP     Response (to client using port 1102)                                               9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122946238-1122946238, ack: 142514017, win:15805, src: 1102  dst:   80 9.24.104.240   9.24.104.210   IP
HTTP     GET Request (from client using port 1102)                                          9.24.104.240   9.24.104.210   IP
HTTP     Response (to client using port 1102)                                               9.24.104.210   9.24.104.240   IP

TCP protocol summary                                    F#: 108/112      Off: 42 (x2A)      L: 20 (x14)
```

Figure 411. IP Trace for the Initial HTTP Communication

Then a direct IIOP communication is established between the client and the server ORBs over a separate socket (see 5a on page 440). The IP trace shows that the TCP port used on the client is 1104 and the TCP port used on the server is 1812:


Network Monitor - [D:\NCF\Security\IIOP\marco.servlet.cap (Summary)]
File Edit Display Tools Options Window Help

```
Protocol Description                                                                    Src Other Addr Dst Other Addr Type
TCP      .AP...., len:  148, seq:1122972701-1122972848, ack: 142540657, win:15805, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:  215, seq: 142540657-142540871, ack:1122972849, win:15696, src: 1812  dst: 1104  9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122972849-1122972849, ack: 142540872, win:15590, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      ...R.., len:    0, seq: 142554876-142554876, ack:1122972849, win:    0, src:   80  dst: 1109  9.24.104.210   9.24.104.240   IP
TCP      .AP...., len:   48, seq:1122972849-1122972896, ack: 142540872, win:15590, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:   20, seq: 142540872-142540891, ack:1122972897, win:15648, src: 1812  dst: 1104  9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122972897-1122972897, ack: 142540892, win:15570, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:  108, seq:1122972897-1122973004, ack: 142540892, win:15570, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:   24, seq: 142540892-142540915, ack:1122973005, win:15540, src: 1812  dst: 1104  9.24.104.210   9.24.104.240   IP
TCP      .AP...., len:  108, seq:1122973005-1122973112, ack: 142540916, win:15546, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:   28, seq: 142540916-142540943, ack:1122973113, win:15432, src: 1812  dst: 1104  9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122973113-1122973113, ack: 142540944, win:15518, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:  108, seq:1122973113-1122973220, ack: 142540944, win:15518, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:   24, seq: 142540944-142540967, ack:1122973221, win:15324, src: 1812  dst: 1104  9.24.104.210   9.24.104.240   IP
TCP      .AP...., len:  108, seq:1122973221-1122973328, ack: 142540968, win:15494, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:   28, seq: 142540968-142540995, ack:1122973329, win:15216, src: 1812  dst: 1104  9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122973329-1122973329, ack: 142540996, win:15466, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:   48, seq:1122973329-1122973376, ack: 142540996, win:15466, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:   20, seq: 142540996-142541015, ack:1122973377, win:15168, src: 1812  dst: 1104  9.24.104.210   9.24.104.240   IP
TCP      .AP...., len:  108, seq:1122973377-1122973484, ack: 142541016, win:15446, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:   24, seq: 142541016-142541039, ack:1122973485, win:15060, src: 1812  dst: 1104  9.24.104.210   9.24.104.240   IP
TCP      .AP...., len:  108, seq:1122973485-1122973592, ack: 142541040, win:15422, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:   28, seq: 142541040-142541067, ack:1122973593, win:14952, src: 1812  dst: 1104  9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122973593-1122973593, ack: 142541068, win:15394, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:  108, seq:1122973593-1122973700, ack: 142541068, win:15394, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:   24, seq: 142541068-142541091, ack:1122973701, win:14844, src: 1812  dst: 1104  9.24.104.210   9.24.104.240   IP
TCP      .AP...., len:  108, seq:1122973701-1122973808, ack: 142541092, win:15370, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP
TCP      .AP...., len:   28, seq: 142541092-142541119, ack:1122973809, win:14736, src: 1812  dst: 1104  9.24.104.210   9.24.104.240   IP
TCP      .A...., len:    0, seq:1122973809-1122973809, ack: 142541120, win:15342, src: 1104  dst: 1812 9.24.104.240   9.24.104.210   IP

HTTP Protocol Packet Summary                            F#: 80/112       Off: 62 (x3E)      L: 371 (x173)
```

Figure 412. IP Trace for the Direct IIOP Communication

The TCP port values are dynamically assigned with IIOP: different tests demonstrated that different ports are involved. However, we had the opportunity to verify what we stated in 5a on page 440. That is, once activated, a direct IIOP connection uses the same socket for the entire period the client is connected to the server.

The firewall technologies that we used in our tests did not permit a direct IIOP connection and so ServletExpress CORBA Support automatically activated IIOP over HTTP. In that case the IP trace demonstrated that only the HTTP port 80 on the server was involved in the communication with the client. For this reason, in our scenario, we did not need additional modules like the IIOP Proxy Plug-in to adapt our firewall to IIOP, as it was supported by the current version of ServletExpress, since the IIOP protocol was carried over regular HTTP sessions.

9.3 Increasing Security with SSL

In order to guarantee security in our firewall-based scenario and control the HTTP and IIOP over HTTP communication, we used all the firewall technologies listed below, depending on the particular test we had to do:

- IP filters for HTTP and IIOP over HTTP
- HTTP Proxy Server for HTTP and IIOP over HTTP
- SOCKS Server for HTTP and IIOP over HTTP

We also tried to ensure security by encrypting the IIOP over HTTP communication through the SSL protocol, in order to have IIOP over HTTPS. What we were able to get in this case was the possibility to encrypt the HTTP initial communication between the client and the server. So for example the applet that activates the client ORB is downloaded through an encrypted flow. But the WebSphere CORBA Support does not offer yet the possibility to encrypt the IIOP over HTTP communication between the client and the server ORBs.

Why? We were informed by the WebSphere CORBA Support development team that the client ORB, which is responsible for controlling the IIOP communication with the server ORB (see 5 on page 439), activates an IIOP over HTTP communication with the server ORB by using a URLConnection object and specifying `http://` as the protocol. In order to encrypt the IIOP over HTTP communication over SSL, the client ORB should specify `https://` rather than `http://`. This is certainly possible, and so an IIOP over HTTPS communcation can be achieved, but there would need to be a mechanism in place for the server to be able to instruct the client whether `http://` or `https://` should be used. In fact there should always be the possibility to encrypt some connections but not others. This mechanism has not been implemented yet in the current version of WebSphere Application Server. Future releases of the WebSphere CORBA Support will probably offer this feature. Currently, when it uses HTTP tunneling, the client ORB always uses `http://` as the specified protocol when the URLConnection is opened.

A brute force solution to get this feature in place would be to implement an ORB property (passed in as a parameter to the ORB init() method) that forces to always use the `https://` method for all the connections during that use of the ORB. But of course in this way the possibility to switch between HTTP and HTTPS is lost.

SOCKS Server for HTTPS and SSL Tunneling

Once it is possible to encrypt IIOP over HTTP using SSL, it will also be possible to control the resulting IIOP over HTTPS flow through the firewall, implementing firewall technologies such as SSL tunneling and SOCKS Server for HTTPS. See 10.1, "SSL Tunneling Scenario" on page 488 and 10.2, "SOCKS Server for HTTPS Scenario" on page 494 for further details.

9.4 Test Environment Configuration

The two-tier scenario that we implemented involved three machines:

1. The client tier was a Windows 95 machine where Netscape Communicator Version 4.05 was installed as Java-enabled Web browser.
2. The server tier was a Windows NT Server 4.0 machine where Lotus Domino Go Webserver Version 4.6.2.2 was installed. The servlet engine was provided by ServletExpress Version 1.0 beta 3, which included ServletExpress CORBA Support. Java Development Kit 1.1.6 was also installed on this machine.
3. The firewall was an AIX Version 4.2 machine, where IBM Firewall Version 3.1.1 was installed. After IBM eNetwork Firewall Version 3.2 was generally available, we installed it on an AIX Version 4.2.1 machine.

You will find detailed information about how to set up the environment in Chapter 7, "How to Install and Configure a NCF Secure Environment" on page 297. The only difference is that now you do not need to install and configure any database client or server, since a DB2 server machine is not present in this two-tier model environment.

We used two IP subnetworks in our scenario, corresponding to the high-level and low-level security subnetworks of our intranet. Figure 406 on page 435 provides a graphical representation of the hardware, software and network configuration of the NCF secure environment that we built.

The purpose of this scenario is to investigate how to use firewall technologies like IP filters, HTTP proxy server and SOCKS server in the CORBA environment provided by ServletExpress. We built this test environment assuming that the client machine, in the high-level security subnetwork of the intranet, communicates with a CORBA application running on the Web server machine, which is located in the low-level security subnetwork of the intranet.

In this scenario we used the sample application named Account that comes with the installation files of ServletExpress. ServletExpress CORBA Support includes all the files related to the Account application in the directory *servexp_root*\web\classes\com\ibm\jbroker\samples\account, *servexp_root* being the root directory where ServletExpress was installed (in our case it was D:\ServletExpress). To compile and install the Account application properly, you need to follow some simple directions:

1. Open an MS-DOS Command Prompt window and let *servexp_root*\web\classes\com\ibm\jbroker\samples\account be the current directory.
2. Build the Account application.

   a. Edit the file makefile.nt, also stored in the account directory, changing the value for the `ServletExpress_ROOT` variable from the default `C:\ServletExpress` to the actual ServletExpress root directory (for example, in our case, we changed it into `D:\ServletExpress`). Notice that in the account directory you will find another makefile, named makefile.unix, that can be used only in AIX or Sun Solaris environments.

   b. Provided that you have already set CLASSPATH, Path and JAVA_HOME properly, launch the following command on the MS-DOS Command Prompt window you have opened:

      ```
      nmake -f makefile.nt all
      ```

      This command will build the application, because it will force the Java compiler `javac` to compile all the Java files that are stored in the account directory.

3. Install the Account application.

   a. Run the following command:

      ```
      nmake -f makefile.nt install
      ```

      from an MS-DOS Command Prompt window, in order to copy the HTML file in the right location known by the Web server, according to what is specified in the httpd.cnf configuration file for Lotus Domino Go Webserver.

   b. Point your Java-enabled Web browser to http://*ourwebserver*/ServletExpress/resources/CORBA/account.html where *ourwebserver* is the fully qualified name of the Lotus Domino Go Webserver machine.

The Account application maintains two bank account objects in the server, and the clients use these two objetcs to make deposits and withdrawals and check the balance. From the client's point of view these accounts are a checking account and a savings account. The client applet can update either account.

First of all, we tested the application without using the firewall, simply pointing the client's browser to the URL http://192.168.50.2/ServletExpress/resources/CORBA/Account.html. The test was successful and then we configured the firewall to protect the high-level security subnetwork of our intranet.

9.5 Firewall Basic Configuration

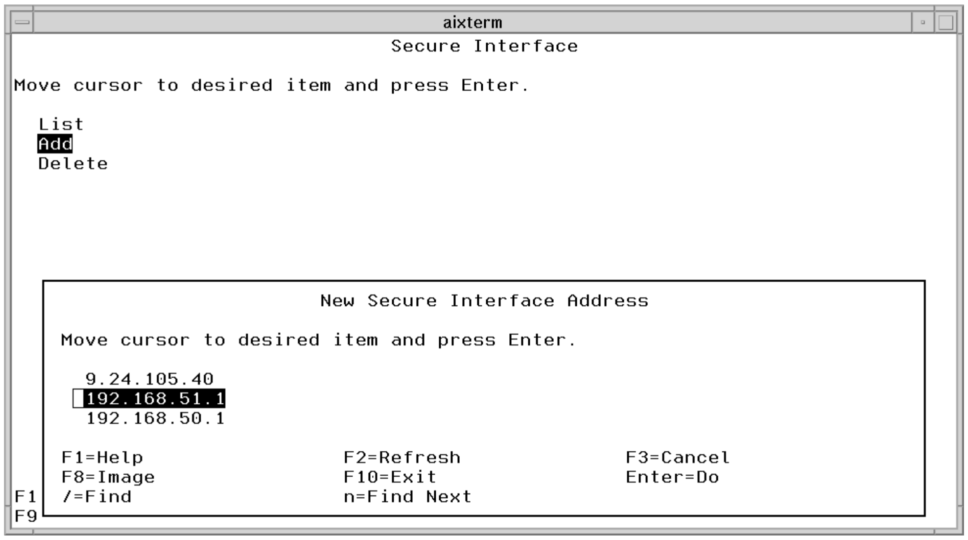
In this section we will show the simple steps necessary to configure the secure network adapter on the firewall and a Domain Name Service in our network environment. The steps we followed to install IBM Firewall 3.1.1 and IBM eNetwork Firewall 3.2 are described in 7.7, “Installation” on page 350. To see how to do the other basic firewall configuration steps, you can refer to 7.8.1, “Remote Configuration” on page 353 and 7.8.2, “Administrator User Addition” on page 354. Notice that the operating system on which we selected to install the firewall was AIX. Things could be different on the Windows NT platform.

9.5.1 Setting the Secure Network Adapter on the Firewall

In order to configure a network interface as a secure network adapter, we entered the following `smit` fast path command:

```
smitty fw_set_secure_adapter
```

on an aixterm window. Then we selected **Add** and pressed **Enter**, as shown in the following screen:



```
                                   aixterm
                              Secure Interface

Move cursor to desired item and press Enter.

  List
  Add
  Delete



    ┌────────────────────────────────────────────────────────────────────────┐
    │                    New Secure Interface Address                        │
    │                                                                        │
    │ Move cursor to desired item and press Enter.                           │
    │                                                                        │
    │   9.24.105.40                                                          │
    │   192.168.51.1                                                         │
    │   192.168.50.1                                                         │
    │                                                                        │
    │ F1=Help                F2=Refresh             F3=Cancel                │
    │ F8=Image               F10=Exit               Enter=Do                 │
F1  │ /=Find                 n=Find Next                                     │
F9  └────────────────────────────────────────────────────────────────────────┘
```

Figure 413. Select the Secure Network Adapter

The New Secure Interface Address section appeared and there we selected **192.168.51.1**, which was the Token-ring adapter that we wanted to configure as secure (see Figure 406 on page 435) and then we pressed **Enter**. This procedure completed successfully and we got a Confirmation message.

9.5.2 Domain Name Service in a Firewall Protected Network Environment

When we performed the scenarios described in this chapters, we also provided our environment with a Domain Name Service (DNS). Even if it is not strictly mandatory that a DNS be installed and configured, its presence in a firewall-protected network environment is very important. In general DNS allows secure users to access all the domain name services of the non-secure network but provides no information to hosts outside the secure network. Moreover the presence of DNS has important consequences in how Java applets can establish network connections through a firewall.

To accomplish its functionality, DNS in a firewall protected network environment requires that three domain name servers are set up:

1. One on the IBM Firewall
2. One inside the secure network
3. One inside the non-secure network

Actually, the role of the domain name server installed on the firewall is named *caching-only nameserver*, because the firewall's domain name server should not contain any database files. For this reason the firewall is in general configured to act like a gateway between the domain name servers for the secure network and that for the non-secure network. The installation process for the IBM eNetwork Firewall will set up the name server successfully for the firewall.

In our environment, we used the firewall machine itself also as domain name server for our secure subnetwork and we used the Windows NT Server machine as domain name server for the low-level security subnetwork. In order to configure a Windows NT Server machine as a domain name server, you have to install and configure Microsoft DNS Server, which is one of the network services for Windows NT.

In general, when a firewall is to separate an intranet from the Internet, using the firewall as domain name server for the secure network can be a dangerous operation, because secure-side information, for example all the database files that would normally reside on the secure server, would get stored in the firewall and so the intranet security would be more exposed to possible attacks. Moreover this configuration can generate workload for the firewall, because the firewall gets involved each time a name must be resolved and not only when a request from the secure side must be forwarded to the non-secure side. In fact if the firewall is the domain name server for the secure network, this means that all the machines of the protected area will point to the firewall as their domain name server. Anyway, considering that in our scenario we were not supposing to interface an intranet to the Internet, but two subnetworks of the same intranet, this configuration did not create any problems.

In order to configure DNS when working with the IBM Firewall, you will have to access and modify the DNS configuration, selecting **System Administration** from the Configuration Client navigation tree. When the view is expanded, you should be able to click **Domain Name Services**, as shown in the following figure:

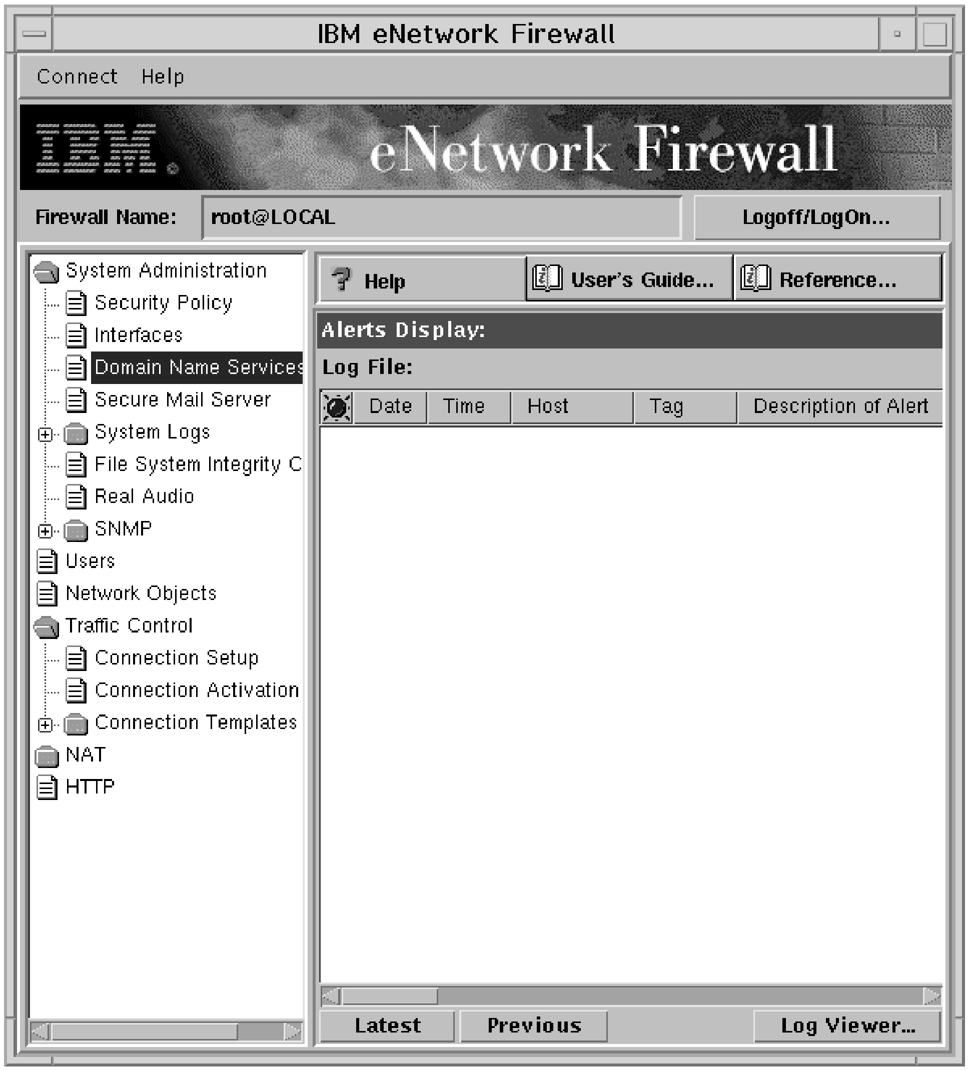


Figure 414. Domain Name Server in the Configuration Client Navigation Tree

The Domain Name Services window is displayed (see Figure 415 on page 448) and you can fill and modify the following three fields:

1. Secure Domain Name

   This field identifies the domain name that the firewall will append to all unqualified host names. We set this field to `test.ibm.com`.

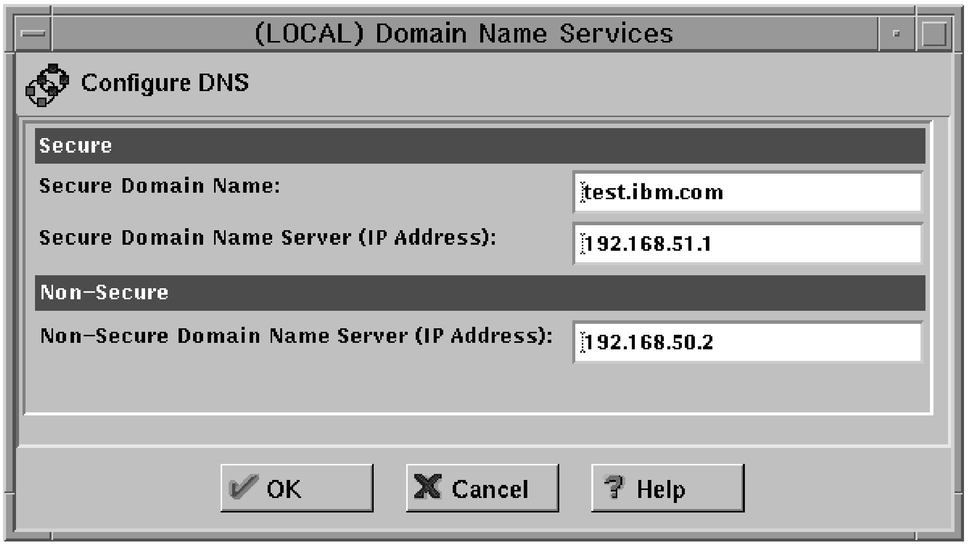
2. Secure Domain Name Server (IP Address)

   This field identifies the IP address of the server that is to resolve names and IP addresses for all the secure hosts. We set this field to `192.168.51.1`, which was the IP address corresponding to the secure network interface on the firewall.

3. Non-Secure Domain Name Server (IP Address)

   This field identifies the IP address of the server that is to resolve names and IP addresses for all the hosts located on the non-secure network. We set this field to `192.168.50.2`, which was the IP address of the Web server machine.

The following figure summarizes our selections:



(LOCAL) Domain Name Services

Configure DNS

Secure

Secure Domain Name: test.ibm.com

Secure Domain Name Server (IP Address): 192.168.51.1

Non-Secure

Non-Secure Domain Name Server (IP Address): 192.168.50.2

Figure 415. Domain Name Services Window

To see how to configure the domain name server on the firewall for the secure network, refer to *IBM eNetwork Firewall for AIX User's Guide*.

If you want to enable your firewall to use the DNS that you have installed and configured, you should select **System Administration** from the Configuration Client navigation tree and then double click **Security Policy**, as shown in the following figure:

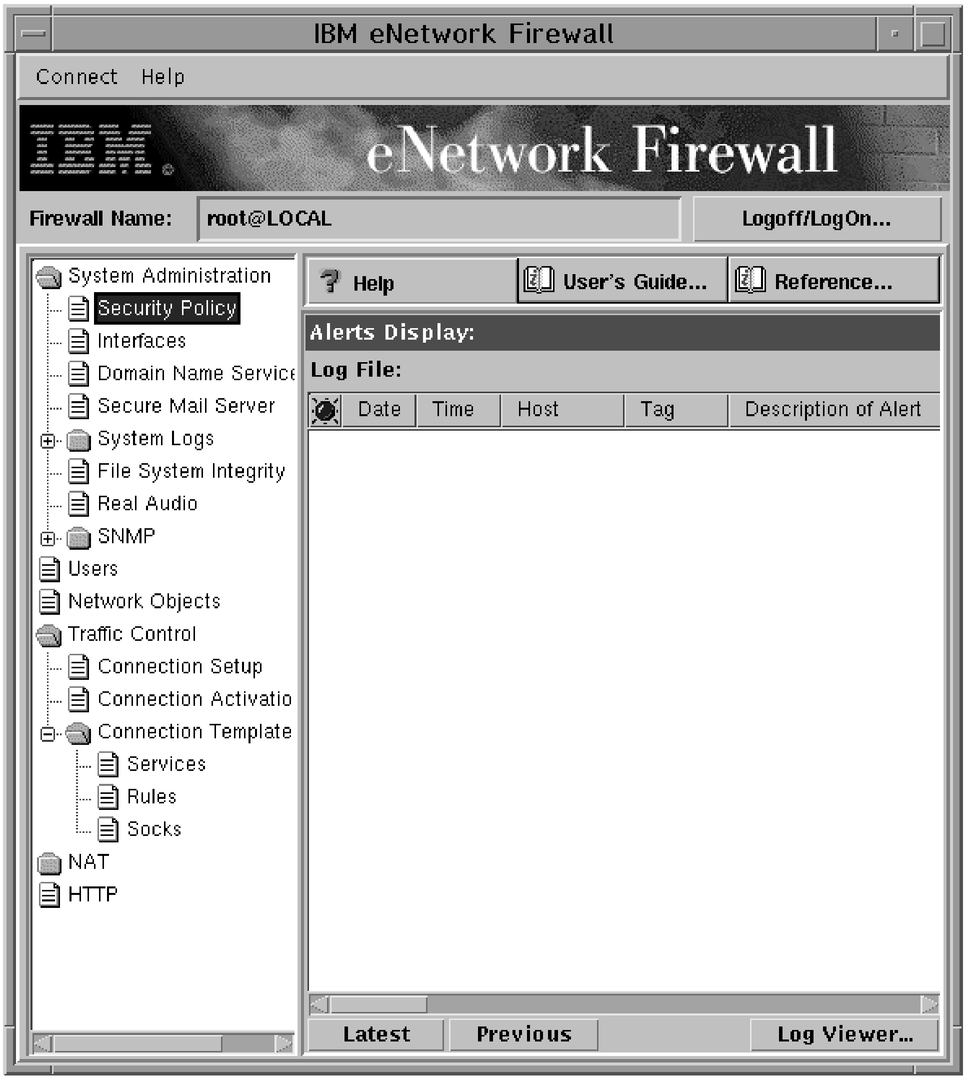
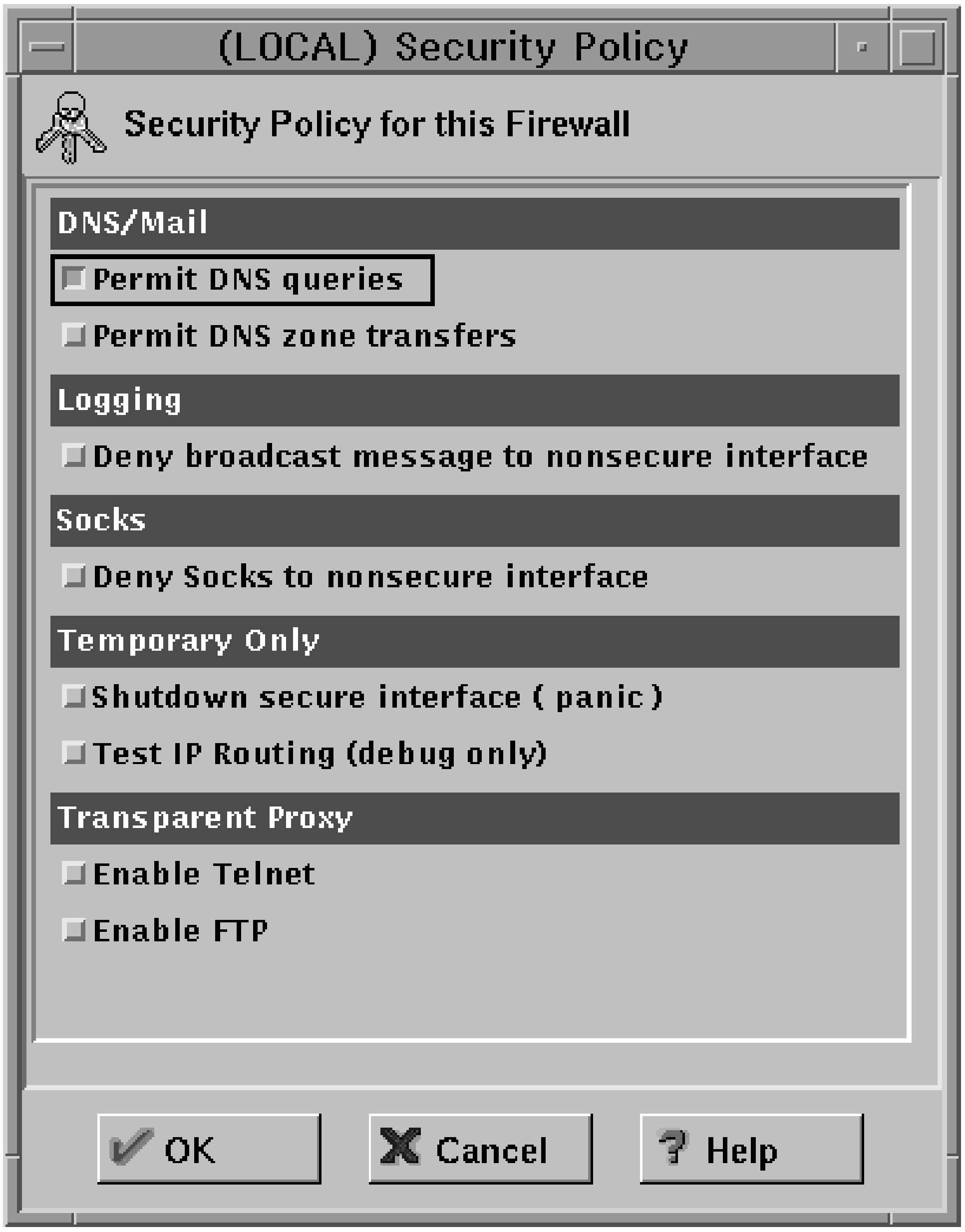


Figure 416. Security Policy in the Configuration Client Navigation Tree

The Security Policy window is brought up. You simply have to mark the box **Permit DNS queries** and then press the **OK** button.



DNS/Mail

- [ ] Permit DNS queries
- [ ] Permit DNS zone transfers

Logging

- [ ] Deny broadcast message to nonsecure interface

Socks

- [ ] Deny Socks to nonsecure interface

Temporary Only

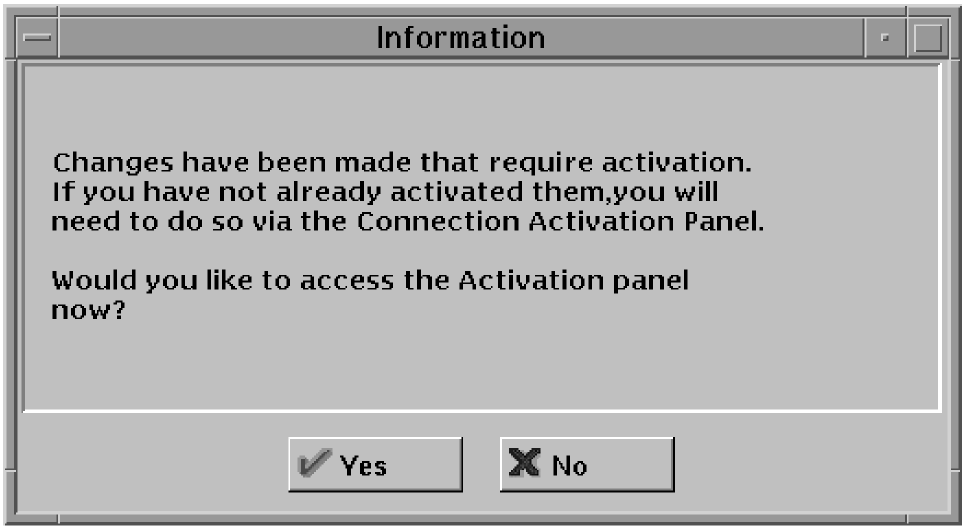
- [ ] Shutdown secure interface (panic)
- [ ] Test IP Routing (debug only)

Transparent Proxy

- [ ] Enable Telnet
- [ ] Enable FTP

Figure 417. How to Enable DNS Queries through the Security Policy Window

Then you are automatically informed by the system that your changes require activation, in order to take effect:


Information

Changes have been made that require activation.
If you have not already activated them,you will
need to do so via the Connection Activation Panel.

Would you like to access the Activation panel
now?

Figure 418. The Information Panel Informs that Activation Is Necessary

If you click **Yes**, a Connection Activation window (see Figure 428 on page 462) will be brought up and you simply have to check the box **Regenerate Connection Rules and Activate** and click **Execute**. You will be able to disable this function later, if you want, unselecting the option **Permit DNS queries** in the Security Policy window (see Figure 417 on page 450).

9.6 IP Filters for HTTP and IIOP over HTTP Scenario

This section focuses on how to use the IP filter technology to enable IIOP over HTTP to pass through the firewall. We performed this test using both the Versions 3.1.1 and 3.2 for IBM Firewall.

IP filter technology allows you to control the traffic through the firewall. The decision whether to permit or deny a packet is based on options like protocol types, port numbers and IP addresses. In general a firewall is configured to act also as a proxy or SOCKS server. In this case it performs the necessary work on behalf of the secure user, and the non-secure host will never know that the secure host exists. In this first implementation, we did not use a proxy or SOCKS gateway; therefore, the firewall had to route the necessary traffic. If the firewall simply routes the traffic, then the secure host and the non-secure host will speak directly to each other, unless Network Address Translation (NAT) is used (for further information, read the Request for Comments RFC1918 at http://info.internet.isi.edu/in-notes/rfc/files/rfc1918.txt). We did not use NAT in our environment, since we assumed that the client and the server were part of the same intranet, even if the server was located in a low-level security subnetwork and the client in a high-level security subnetwork. But if you assume that the client is located in your intranet and the server in the Internet, then you should use NAT; otherwise, the secure host's IP address will be exposed to the network. You can find more information about IP filters in 6.1.2, “Expert IP Filters Using IBM eNetwork Firewall 3.2” on page 271.

In 8.5, “IP Filters Configuration for Three-Tier Applications” on page 390 we saw that sometimes you have to define new rules and services, when you want to configure IP filters and the predefined rules and services that IBM Firewall provides do not match your needs. In this chapter we will see how the configuration is easier and faster when predefined rules and services can be applied.

Notice that predefined rules and services allow a secure adapter on the firewall to have access to a non-secure adapter. For this reason we could configure the adapter corresponding to the client machine as secure and use all predefined rules and services provided by IBM Firewall. This configuration would match also our choice to consider the client machine as located in the high-level security subnetwork of our intranet.

9.6.1 How to Configure IP Filters for HTTP and IIOP over HTTP

IP filters are tools capable of permitting or denying IP packets from passing through the firewall. In 6.1.2, "Expert IP Filters Using IBM eNetwork Firewall 3.2" on page 271, we discussed connections, network objects, services and rules.

In this section we will describe how we implemented the IP filter firewall technology for HTTP and IIOP over HTTP in our scenario environment. First of all, we had to define two new objects and one new connection for our scenario:

1. Objects (Type: Host)
   - 95-client

     This object was a Windows 95 IBM ThinkPad located in the secure subnetwork of our intranet. Netscape Communicator 4.05 was installed on this machine as Java-enabled Web browser. This object played the role of the client in this scenario. The applet and the client side ORB ran on this machine.
   - NT-server

     This object was a Windows NT Server 4.0 IBM Personal Computer, located in the low-level security subnetwork. Lotus Domino Go Webserver 4.6.2.2 was installed on this machine and ServletExpress 1.0 beta 3 provided the servlet engine. This object played the key role of the Web server in this scenario. The sample servlet and the server-side ORB ran on this machine.
2. Connection
   - for 80 port

     We created a new connection in order to pass the HTTP protocol from the 95-client object to the NT-server object through the firewall.

We used the `fwconfig` command to configure the firewall (see 7.8.1, "Remote Configuration" on page 353). We logged on as indicated in 7.8.1, "Remote Configuration" on page 353 and then we selected **Network Objects** from the Configuration Client navigation tree, as shown in the following figure:

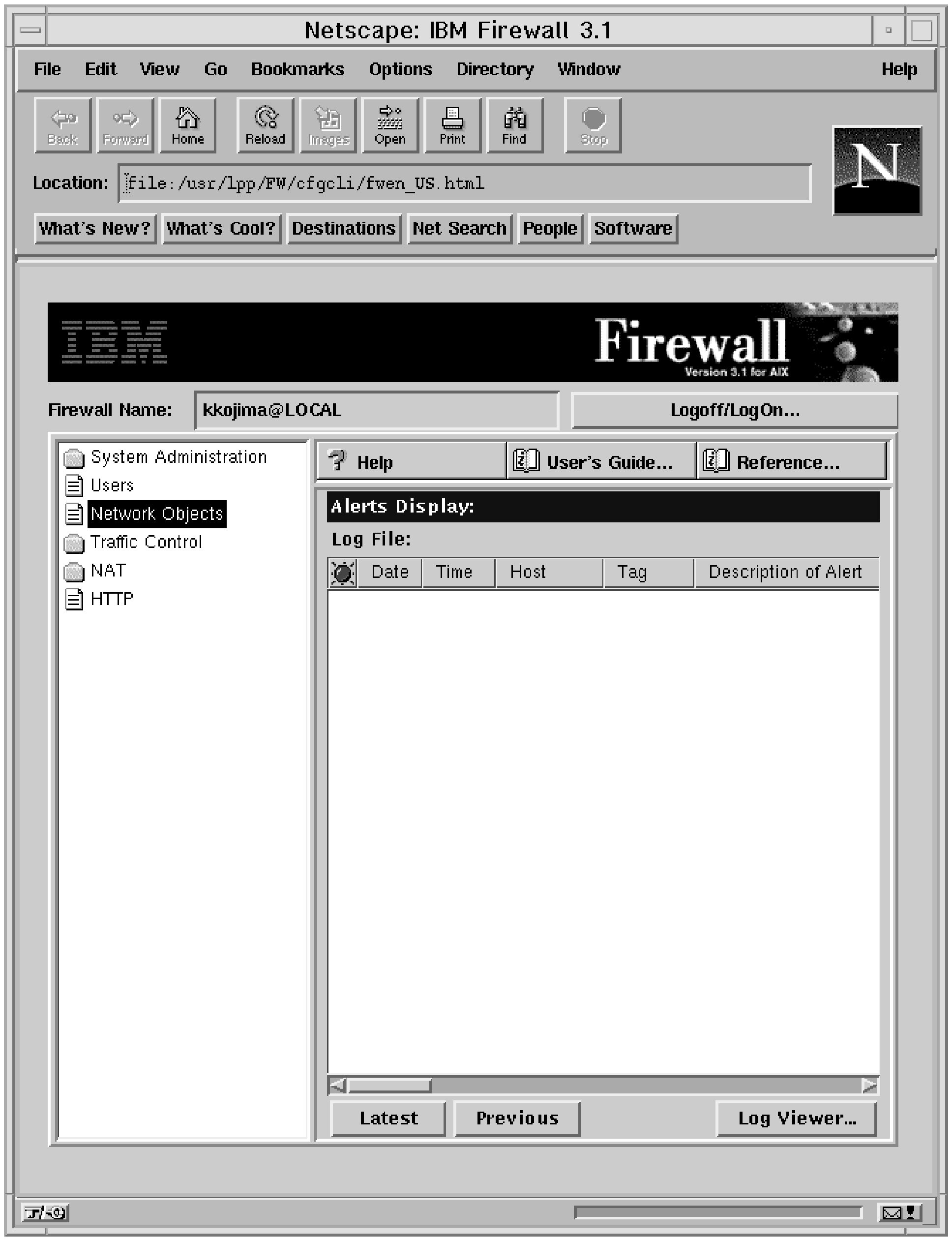
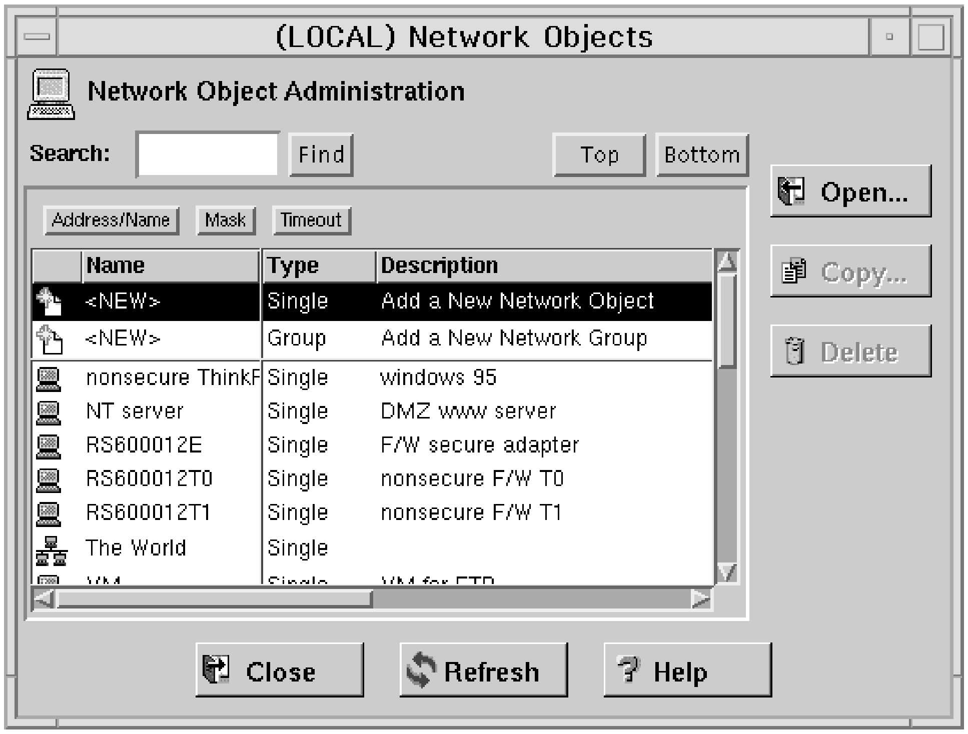

Netscape: IBM Firewall 3.1
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: file:/usr/lpp/FW/cfgcli/fwen_US.html
What's New? What's Cool? Destinations Net Search People Software
IBM Firewall Version 3.1 for AIX
Firewall Name: kkojima@LOCAL
Logoff/LogOn...
System Administration
Users
Network Objects
Traffic Control
NAT
HTTP
Help
User's Guide...
Reference...
Alerts Display:
Log File:
Date Time Host Tag Description of Alert
Latest Previous Log Viewer...

Figure 419. Network Objects in the Configuration Client Navigation Tree

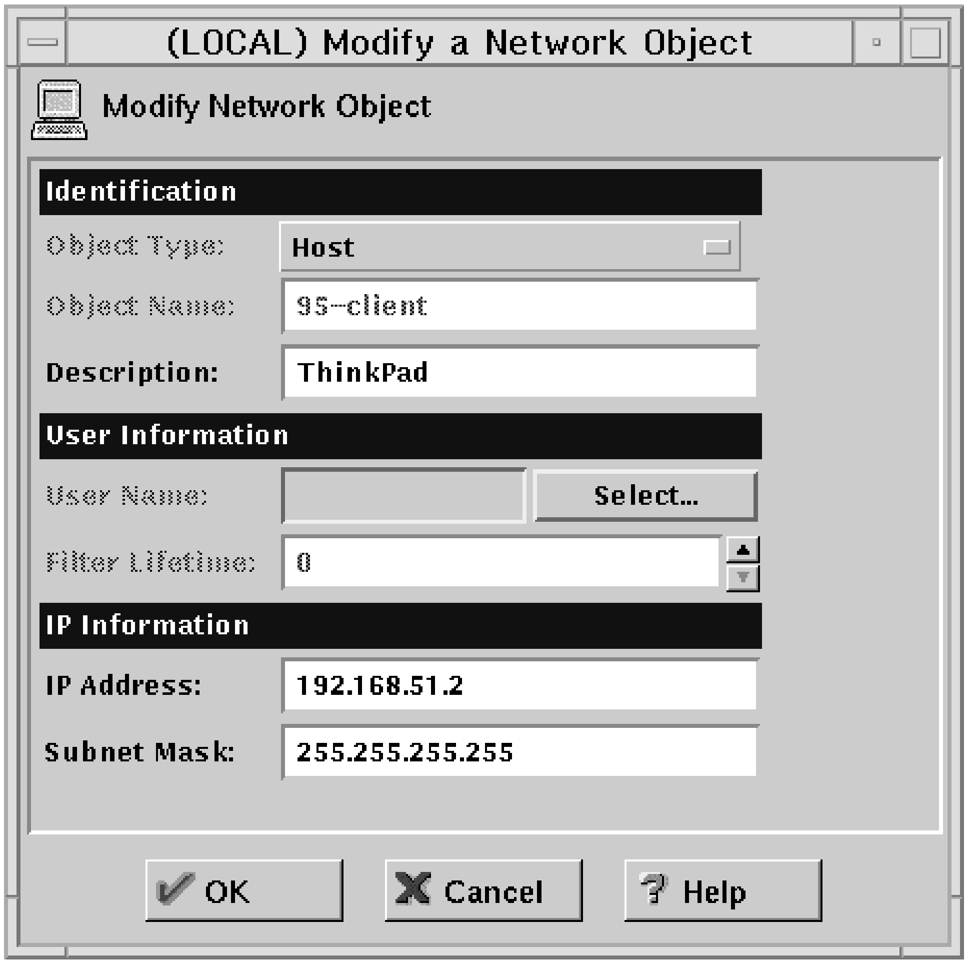
The Network Objects window was brought up. We selected **NEW** and then clicked the **Open...** button, in order to add new network objects, as you can see in the following screen:



Name	Type	Description
<NEW>	Single	Add a New Network Object
<NEW>	Group	Add a New Network Group
nonsecure ThinkP	Single	windows 95
NT server	Single	DMZ www server
RS600012E	Single	F/W secure adapter
RS600012T0	Single	nonsecure F/W T0
RS600012T1	Single	nonsecure F/W T1
The World	Single	
VM	Single	VM for FTP

Figure 420. Network Objects Window

The window that you get is to define new network objects and their properties. We defined two network objects named 95-client and NT-server. The type of these objects was Host and we also provided a brief description for the two new objects, the IP address and subnet mask. The following two pictures show how we added the two objects 95-client and NT-server:


(LOCAL) Modify a Network Object

Modify Network Object

Identification

Object Type:	Host
Object Name:	95-client
Description:	ThinkPad

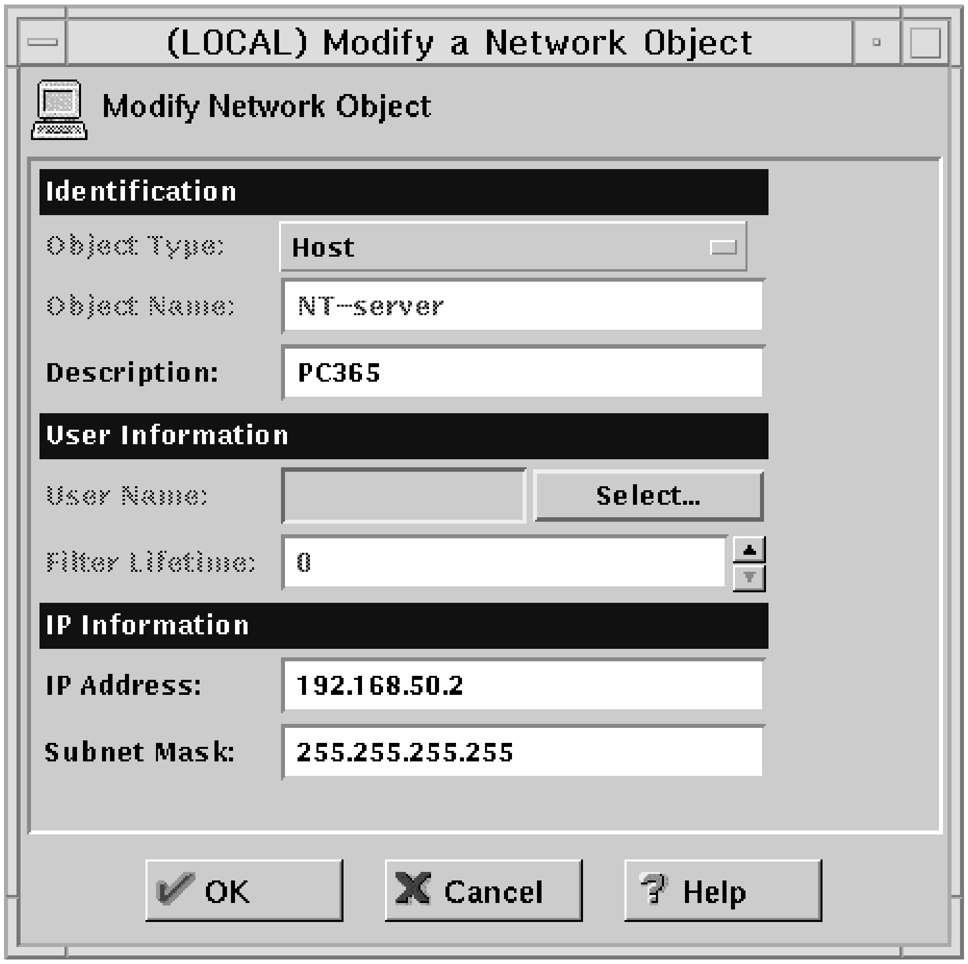
User Information

User Name:	
Filter Lifetime:	0

IP Information

IP Address:	192.168.51.2
Subnet Mask:	255.255.255.255

Figure 421. 95-client Object Definition


(LOCAL) Modify a Network Object

Modify Network Object

Identification

Object Type: Host

Object Name: NT-server

Description: PC365

User Information

User Name:

Filter Lifetime: 0

IP Information

IP Address: 192.168.50.2

Subnet Mask: 255.255.255.255

Figure 422. NT-server Object

To finish, we pressed the **OK** button to register the two new objects. Notice that the network interfaces on the firewall did not need to be registered as network objects, since the flow is direct between the client and the server when the firewall simply routes the traffic, as happens in this case.

We went back to the Configuration Client navigation tree. We then selected **Connection Setup** in order to configure the new connection between the two new network objects that we had defined. Notice that you can access the Connection Setup item only after selecting **Traffic Control**. The following figure shows this situation:

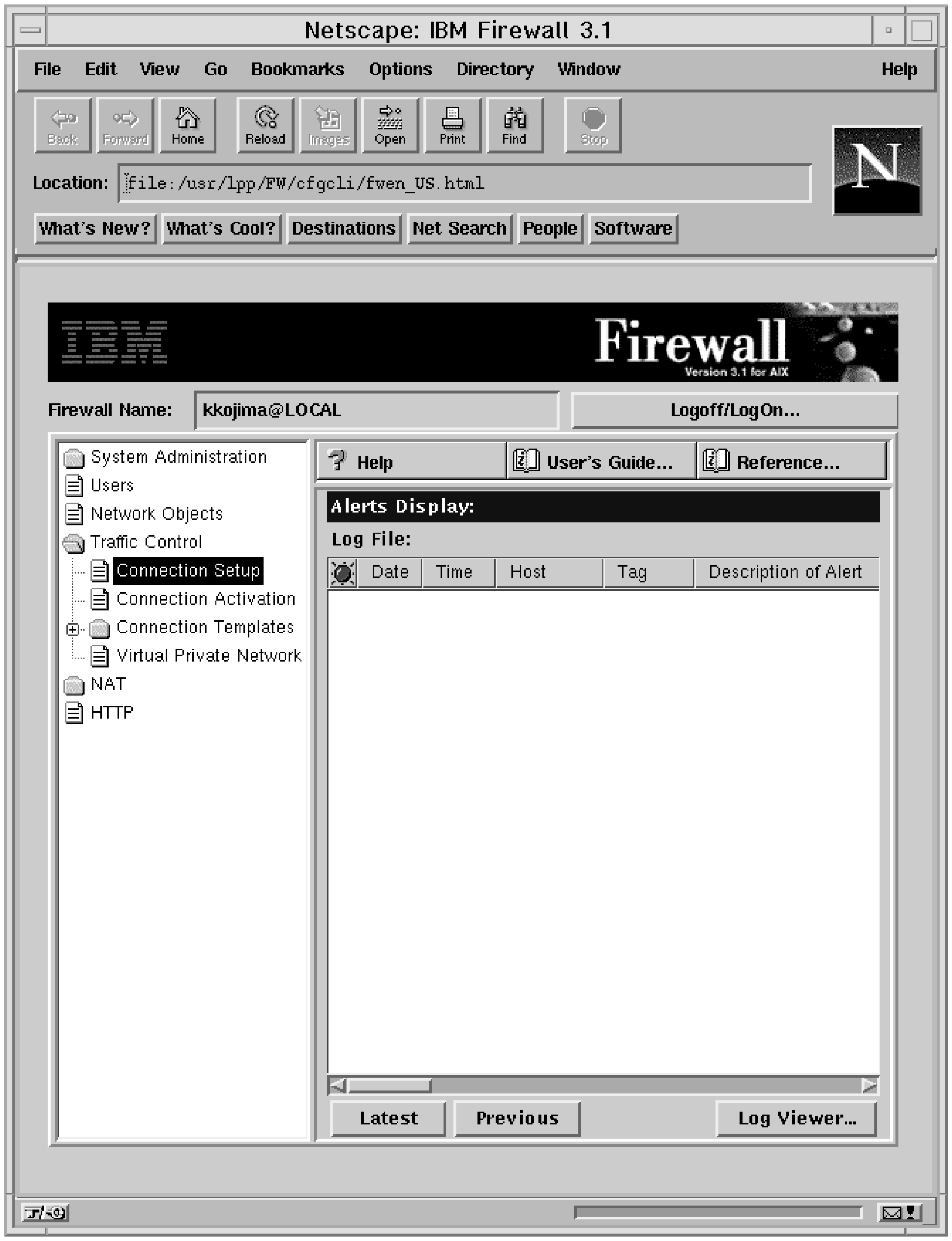
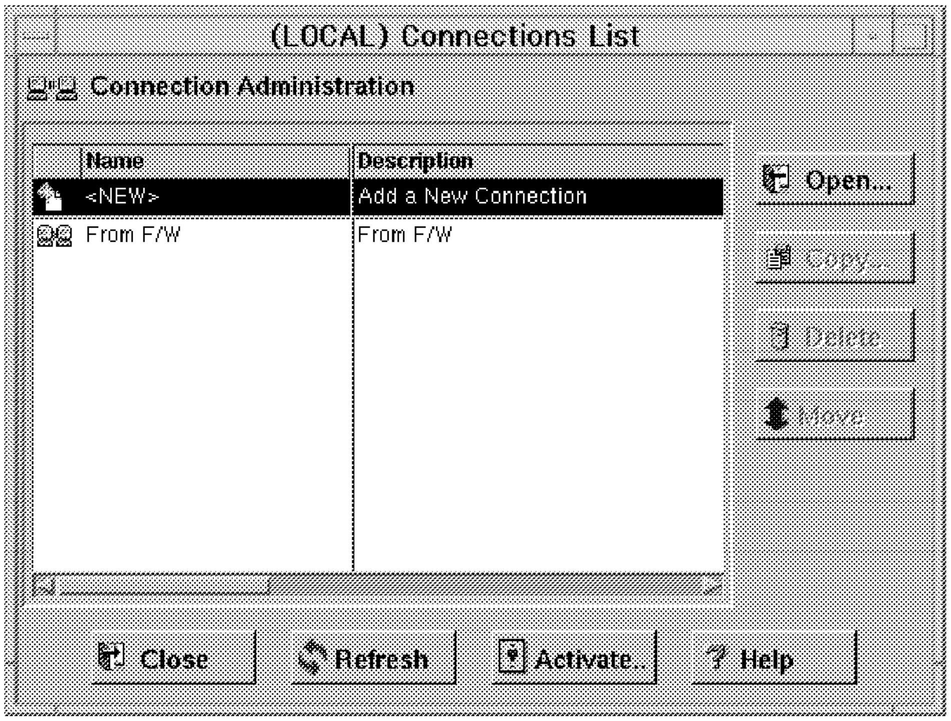

Netscape: IBM Firewall 3.1
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: file:/usr/lpp/FW/cfgcli/fwen_US.html
What's New? What's Cool? Destinations Net Search People Software
IBM Firewall Version 3.1 for AIX
Firewall Name: kkojima@LOCAL
Logoff/LogOn...
System Administration
Users
Network Objects
Traffic Control
Connection Setup
Connection Activation
Connection Templates
Virtual Private Network
NAT
HTTP
Help
User's Guide...
Reference...
Alerts Display:
Log File:
Date Time Host Tag Description of Alert
Latest
Previous
Log Viewer...

Figure 423. Connection Setup in the Configuration Client Navigation Tree

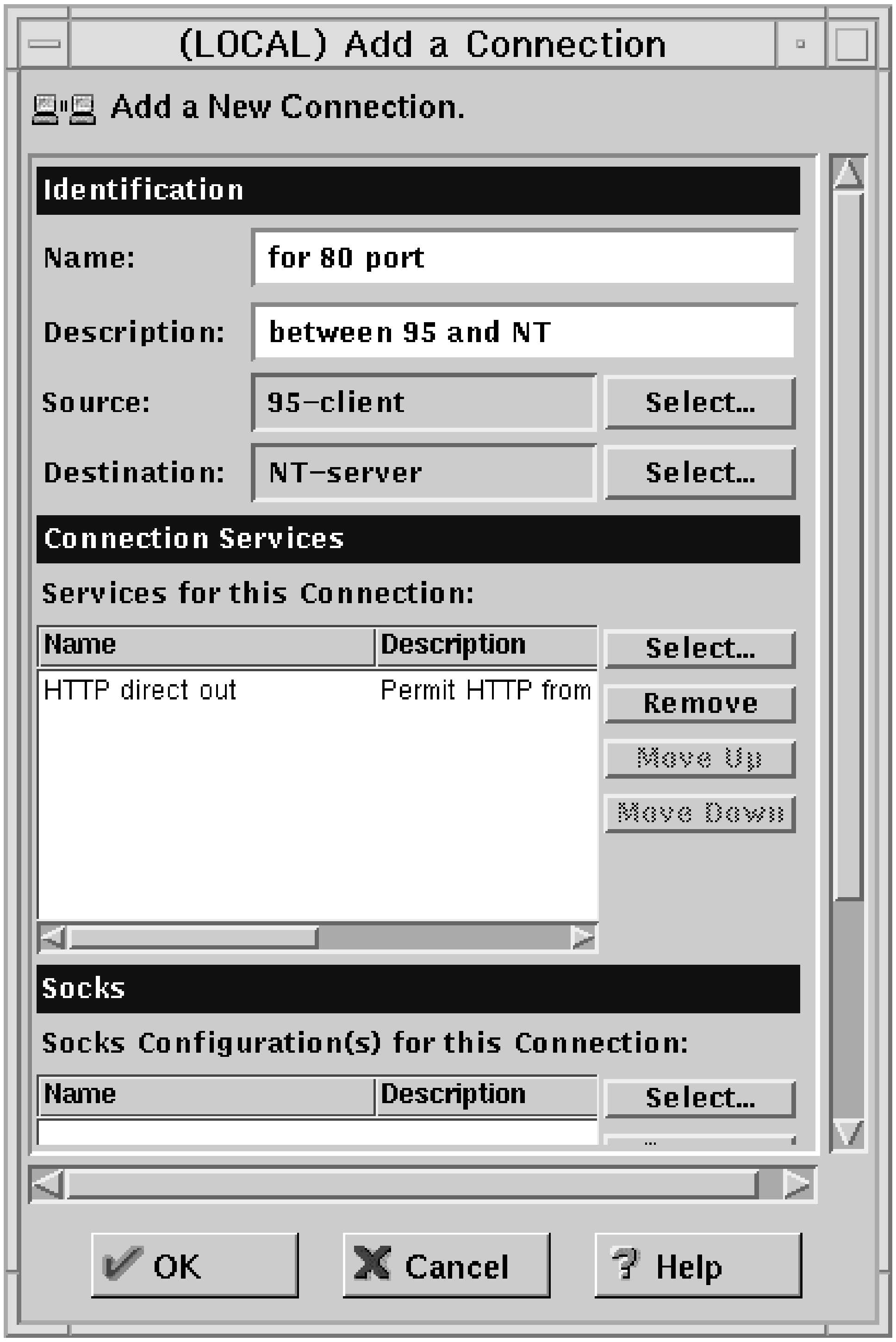
The Connections List window appeared. We selected **NEW** and pressed the **Open...** button, in order to create a new connection, as shown in the following picture:



Name	Description
<NEW>	Add a New Connection
From F/W	From F/W

Figure 424. How to Add a New Connection

As soon as we pressed the **Open...** button, the Add a Connection window was brought up and we filled all its fields with the appropriate values:


(LOCAL) Add a Connection

Add a New Connection.

Identification

Name:	for 80 port
Description:	between 95 and NT
Source:	95-client
Destination:	NT-server

Connection Services

Services for this Connection:

Name	Description
HTTP direct out	Permit HTTP from

Socks

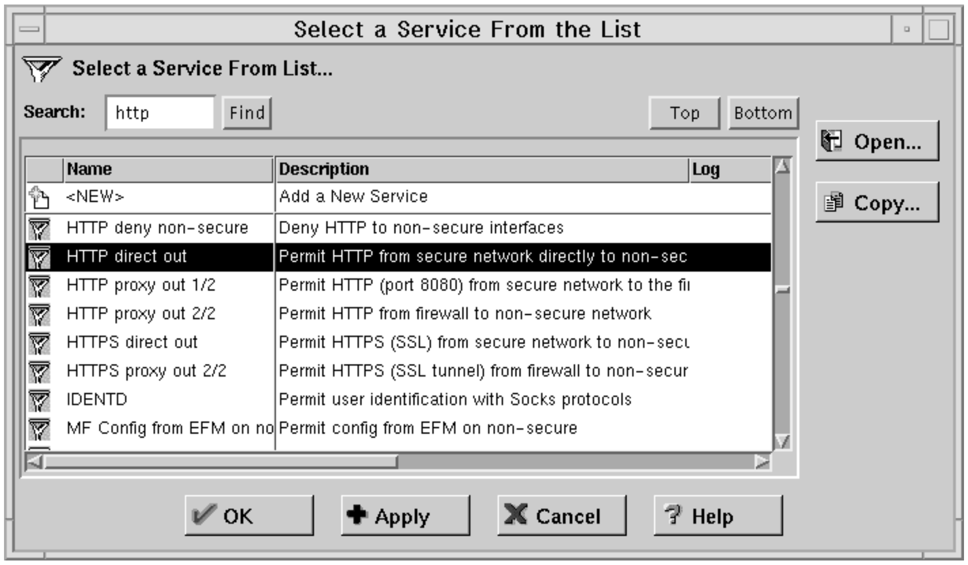
Socks Configuration(s) for this Connection:

Name	Description

Figure 425. Adding for 80 port Connection

As you can see from the above figure, we set Name to `for 80 port` and Description to `between 95 and NT`. Then we had to select the source and the destination objects. For both these items, it was enough to click on the **Select...** button and make the appropriate choices in the Network Objects list that was brought up (see Figure 420 on page 454). We selected **95-client** and **NT-server**, the two Network Objects that we had created.

Also we had to select the services for this connection, by pushing the **Select...** button for the Connection Services frame. You should create a new service only if the service that you want is not included in the services list provided by IBM Firewall for the connection you are going to add. In this scenario we simply needed the service which permits the HTTP protocol to pass between the client object and the server object. Because ServletExpress CORBA Support can also carry IIOP over HTTP sessions to make ORBs communicate, there is no need for the firewall to consider the IIOP protocol and no new service had to be created. For this reason, we could select **HTTP direct out**, which is the predefined service that comes standard with the IBM Firewall and is already configured to allow the HTTP protocol. We highlighted **HTTP direct out** from the services list provided by IBM Firewall and then pressed **OK**, as shown in the following screen:



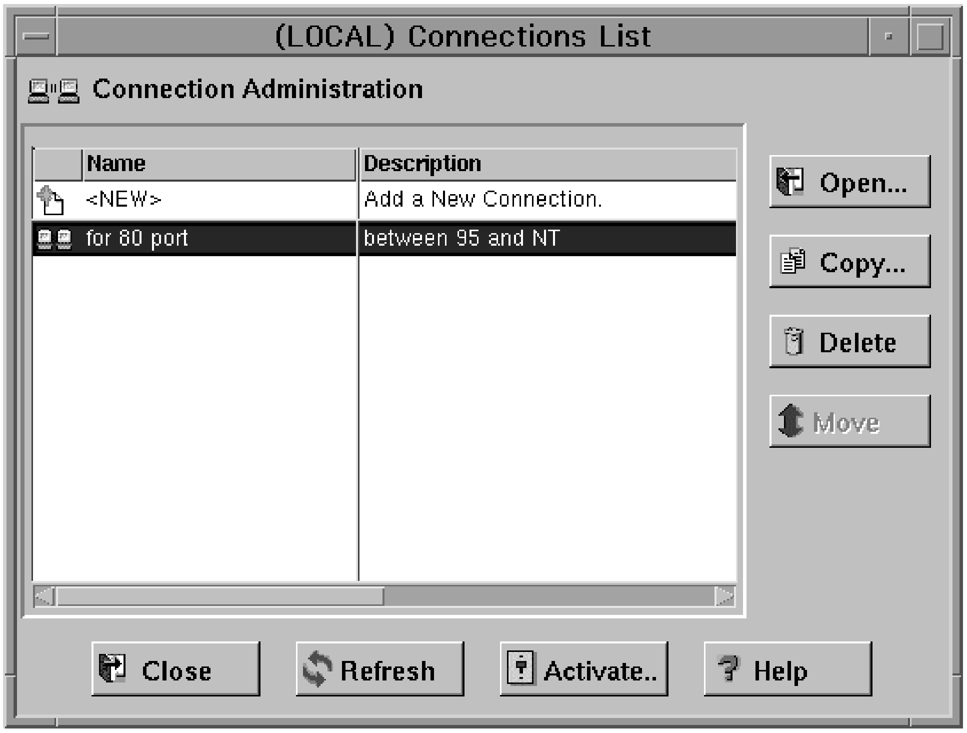
Search: http

Name	Description	Log
<NEW>	Add a New Service	
HTTP deny non-secure	Deny HTTP to non-secure interfaces	
HTTP direct out	Permit HTTP from secure network directly to non-sec	
HTTP proxy out 1/2	Permit HTTP (port 8080) from secure network to the fi	
HTTP proxy out 2/2	Permit HTTP from firewall to non-secure network	
HTTPS direct out	Permit HTTPS (SSL) from secure network to non-secu	
HTTPS proxy out 2/2	Permit HTTPS (SSL tunnel) from firewall to non-secur	
IDENTD	Permit user identification with Socks protocols	
MF Config from EFM on no	Permit config from EFM on non-secure	

Figure 426. Selecting the HTTP direct out Service in the Services List

The connection definition is saved by pressing the **OK** button in the Add a Connection window, as shown in Figure 425 on page 459.

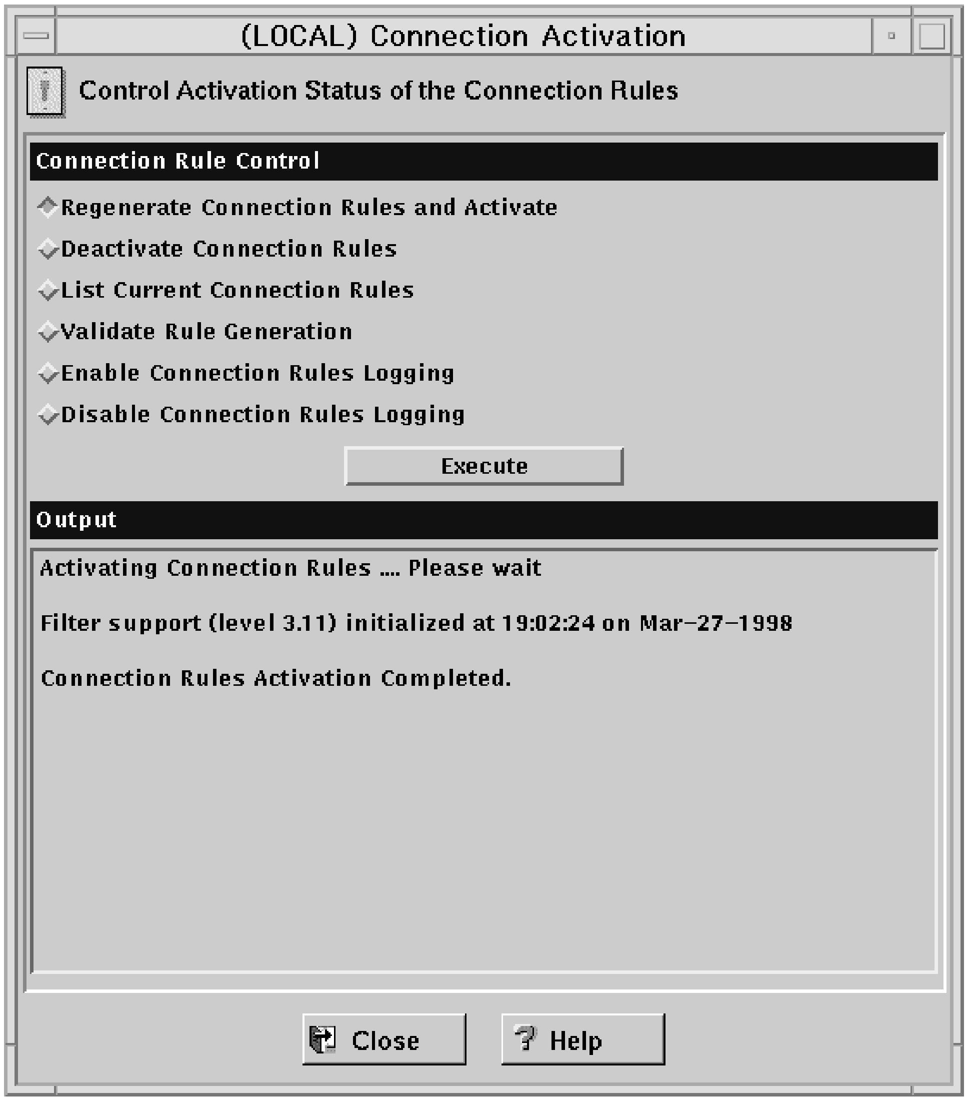
If you follow all these steps, the new connection will appear between all the current connections in the Connections List window. Be sure that this connection is actually the *only* connection in that list (eventually you can delete the other ones by clicking **Delete**), so that the Connection List window will look like the following figure:



Name	Description
<NEW>	Add a New Connection.
for 80 port	between 95 and NT

Figure 427. Connections List Window

Then you can activate this connection by pressing the **Activate...** button. This operation brought up the Connection Activation window. We simply had to mark the check box **Regenerate Connection Rules and Activate**. Then we pressed the **Execute** button, as shown in the following screen:

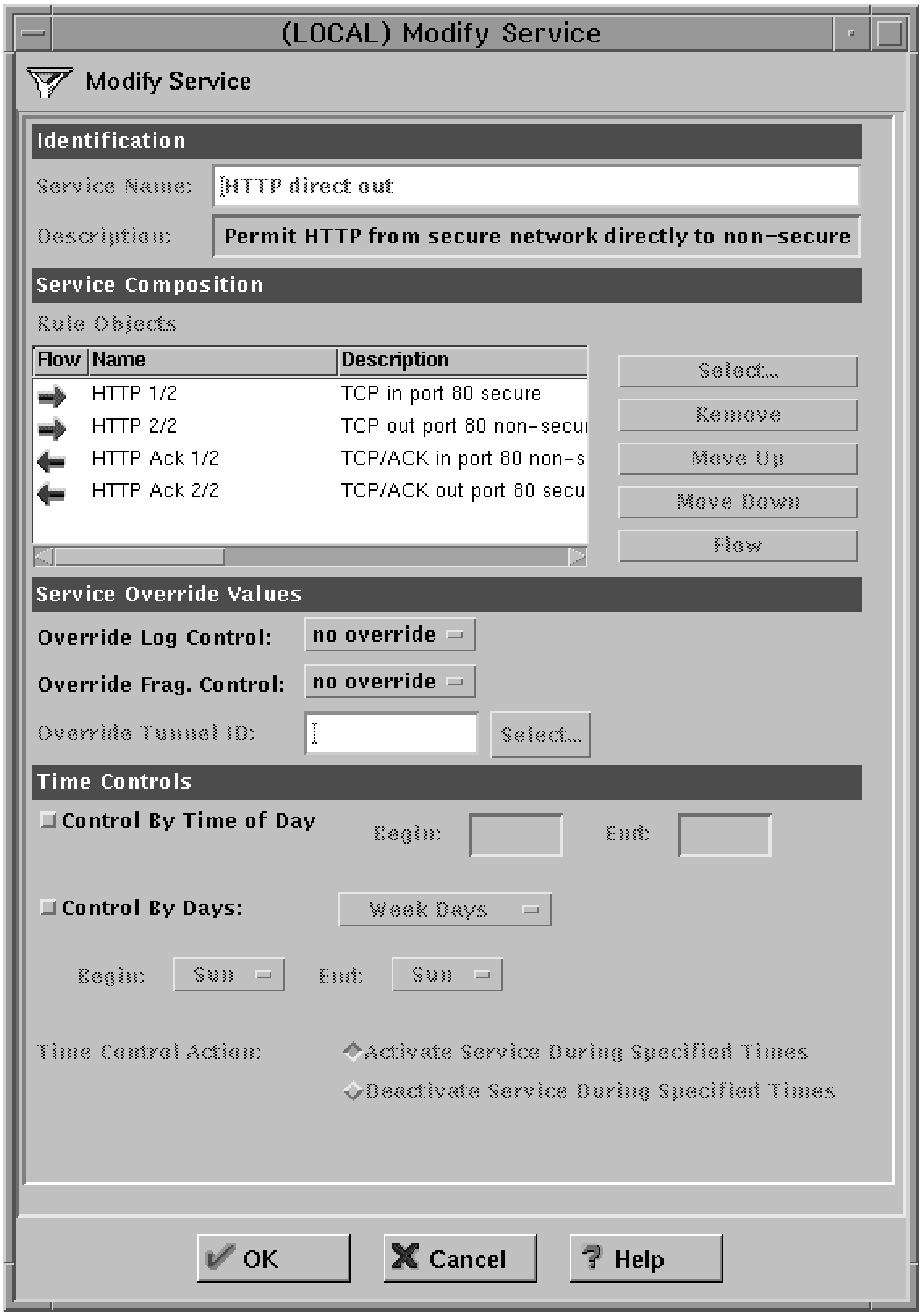

(LOCAL) Connection Activation
Control Activation Status of the Connection Rules
Connection Rule Control
Regenerate Connection Rules and Activate
Deactivate Connection Rules
List Current Connection Rules
Validate Rule Generation
Enable Connection Rules Logging
Disable Connection Rules Logging
Execute
Output
Close
Help

```
Activating Connection Rules .... Please wait

Filter support (level 3.11) initialized at 19:02:24 on Mar-27-1998

Connection Rules Activation Completed.
```

Figure 428. Connection Activation Window

This action forced IBM Firewall to create IP filters according to our connection definition.

Since we simply had to pick up a predefined service from the services list, in order to build the new connection, it was not necessary for us to open its defining window. It is useful anyway to see that window in order to have a better understanding of the HTTP direct out service. You can access this window by double clicking on the service name **HTTP direct out** from the services list:



Identification

Service Name: HTTP direct out

Description: Permit HTTP from secure network directly to non-secure

Service Composition

Rule Objects

Flow	Name	Description
➡	HTTP 1/2	TCP in port 80 secure
➡	HTTP 2/2	TCP out port 80 non-secu
⬅	HTTP Ack 1/2	TCP/ACK in port 80 non-s
⬅	HTTP Ack 2/2	TCP/ACK out port 80 secu

Service Override Values

Override Log Control: no override

Override Frag. Control: no override

Override Tunnel ID:

Time Controls

☐ Control By Time of Day Begin: End:

☐ Control By Days: Week Days

Begin: Sun End: Sun

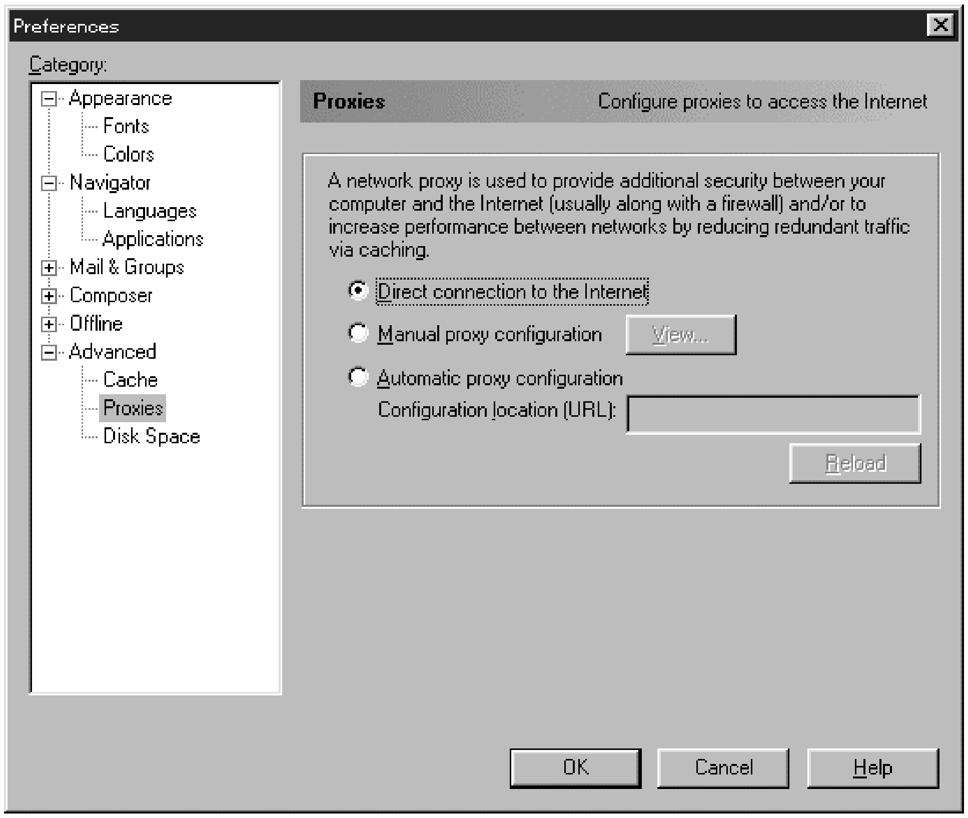
Time Control Action: ◇ Activate Service During Specified Times ◇ Deactivate Service During Specified Times

Figure 429. HTTP direct out Service Window

9.6.2 Netscape Communicator Advanced Configuration for Using IP Filters

We could then test the Account sample application provided by ServletExpress CORBA Support. In order to pass normal HTTP and IIOP over HTTP through the firewall using the IP filter technology, we had to make a very simple configuration for our Web browser on the client machine. We selected **Preferences...** from the Edit menu for the Netscape Communicator window. The Preferences window appeared. In the Category tree, on the left, we selected **Advanced** and then

Proxy and we made sure that the box named **Direct connection to the Internet** was marked, as shown in the following screen:



A network proxy is used to provide additional security between your computer and the Internet (usually along with a firewall) and/or to increase performance between networks by reducing redundant traffic via caching.

[✓ Direct connection to the Internet]
Manual proxy configuration
Automatic proxy configuration
Configuration location (URL):

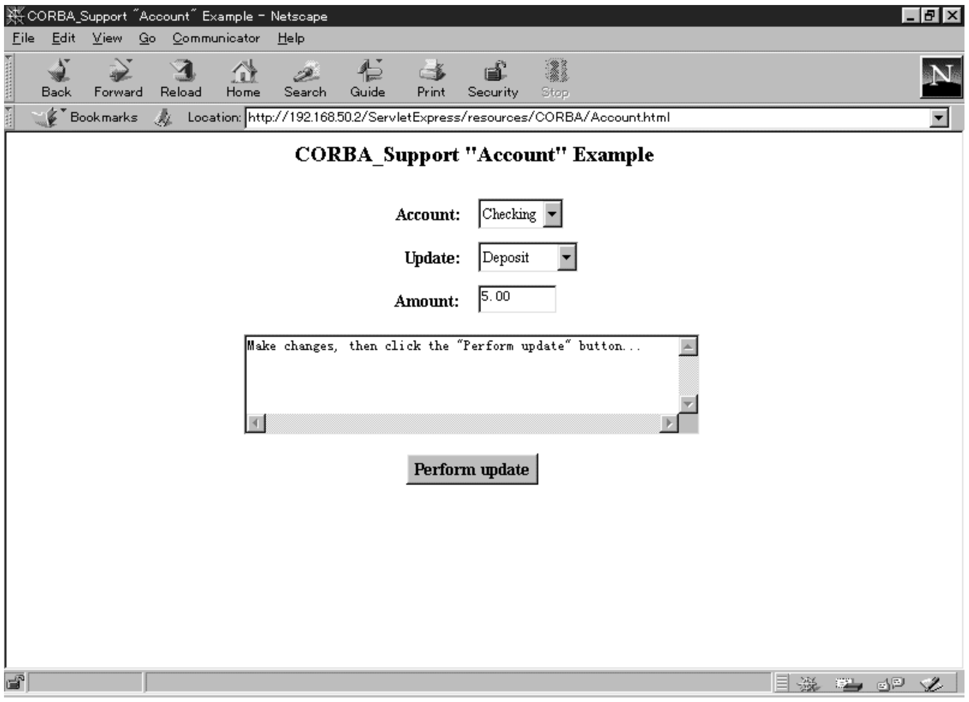
Figure 430. Netscape Communicator Advanced Preferences Configuration

Then we saved this configuration on the browser by pressing **OK.** This step is important if you have configured the firewall to simply route the traffic, without acting as a proxy or SOCKS server. This means that the firewall does not have to perform any work on behalf of the client. For this reason, the client and the server can speak directly inside your network and you should only be sure that the client's Web browser is configured properly for this.

9.6.3 Testing the Application Using IP Filters

This firewall configuration was IP filters-based and only permitted that a TCP port greater than 1023 on the 95-client object connected to port 80 on the NT-server object. No other port could be accessed on the Web server. This granted security but this is also the reason why an IIOP direct connection was not permitted.

When we tried to run the Account application, the experiment was successful and we saw the following window on the client's browser:


CORBA_Support "Account" Example - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://192.168.50.2/ServletExpress/resources/CORBA/Account.html

CORBA_Support "Account" Example

Account: Checking

Update: Deposit

Amount: 5.00

Make changes, then click the "Perform update" button...

Perform update

Figure 431. Account Sample Application

9.7 HTTP Proxy Server for HTTP and IIOP over HTTP Scenario

This section focuses on how to use the HTTP proxy server technology to enable HTTP and IIOP over HTTP to pass through the firewall. We performed this test using both Versions 3.1.1 and 3.2 of the IBM Firewall.

HTTP proxy server is a firewall technology capable of handling browser requests to the Web server. It works on the application layer of the ISO/OSI network model and for this reason it is also known as an application level gateway. If a user wants to connect the Web server machine from the client machine through a firewall implementing proxy server technology, that user must connect to the proxy server first and then request the proxy to connect the destination Web server. In this way the connection between the client and the server is broken, since the proxy acts on behalf of the secure client. You can find more information about the HTTP proxy server firewall technology in 6.1.3, “Proxy Servers” on page 280.

In this scenario, we had to guarantee that the client and the server, using regular HTTP first and then IIOP over HTTP in the CORBA 2.0 environment provided by ServletExpress, can communicate through the the HTTP proxy server running on the firewall machine.

9.7.1 How to Configure the HTTP Proxy Server for HTTP and IIOP over HTTP

This section describes how to configure the firewall to implement the HTTP proxy server technology.

The following list shows the network objects and connections that we needed to configure in order for the firewall to act as an HTTP proxy server:

1. Objects (Type: Host)
   - 95-client

     This object was a Windows 95 IBM ThinkPad located in the secure subnetwork of our intranet. Netscape Communicator 4.05 was installed on this machine as Java-enabled Web browser. This object played the role of the client in this scenario. The applet and the client side ORB ran on this machine.
   - NT-server

     This object was a Windows NT Server 4.0 IBM Personal Computer, located in the low-level security subnetwork. Lotus Domino Go Webserver 4.6.2.2 was installed on this machine and ServletExpress 1.0 beta 3 provided the servlet engine. This object played the key role of the Web server in this scenario. The sample application and the server side ORB ran on this machine.
   - RS600012T0

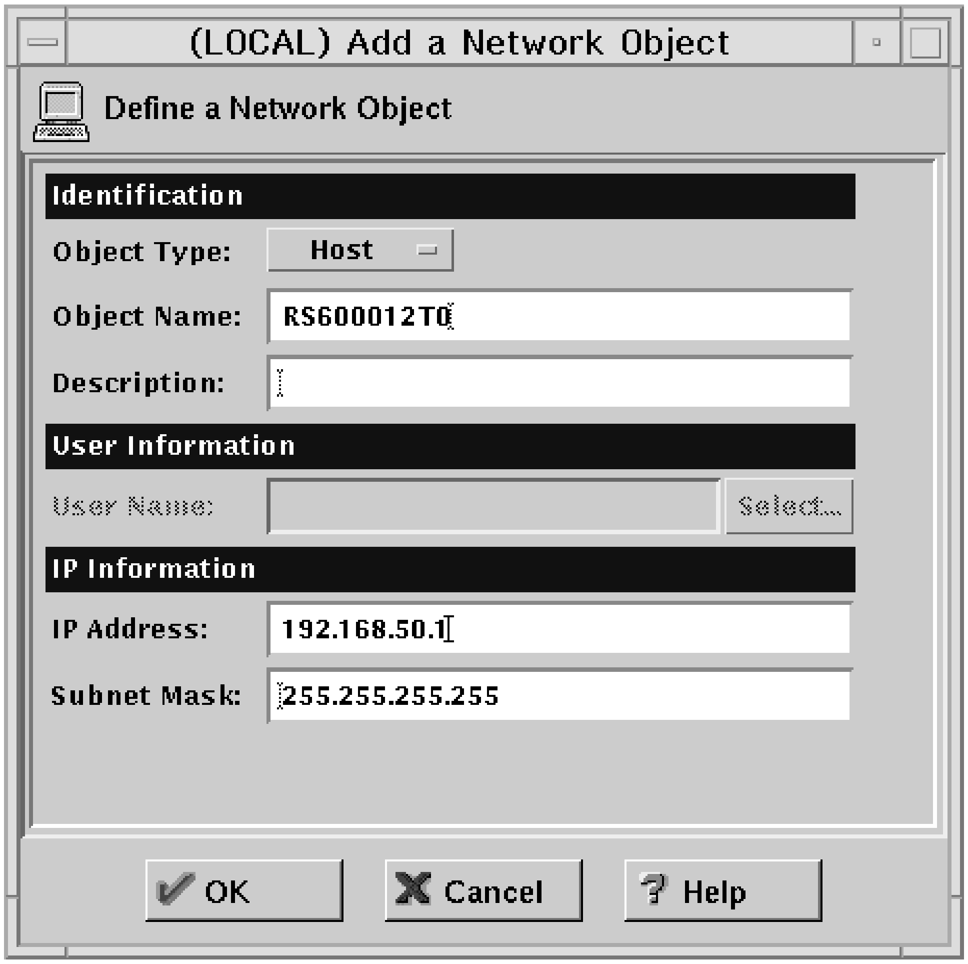
     This object is the non-secure network interface on the firewall machine to which IP address 192.168.50.1 was assigned.
   - RS600012T1

     This object is the secure network interface on the firewall machine to which IP address 192.168.51.1 was assigned.
2. Connections
   - HTTP Proxy to NT

     We created this new connection in order to pass the HTTP protocol from the RS600012T0 object to the NT-server object through the firewall.
   - Secure to HTTP Proxy

     We created this new connection in order to pass the HTTP protocol from the 95-client object to the RS600012T1 object through the firewall.

We already described in 9.6.1, “How to Configure IP Filters for HTTP and IIOP over HTTP” on page 452 how to add the two network objects 95-client (see Figure 421 on page 455) and NT-server (see Figure 422 on page 456). Following the same steps, we could define the two additional network objects RS600012T0 and RS600012T1 that we mentioned in the above list. The following two windows show how we added them:


(LOCAL) Add a Network Object

Define a Network Object

Identification

Object Type:	Host
Object Name:	RS600012T0
Description:	

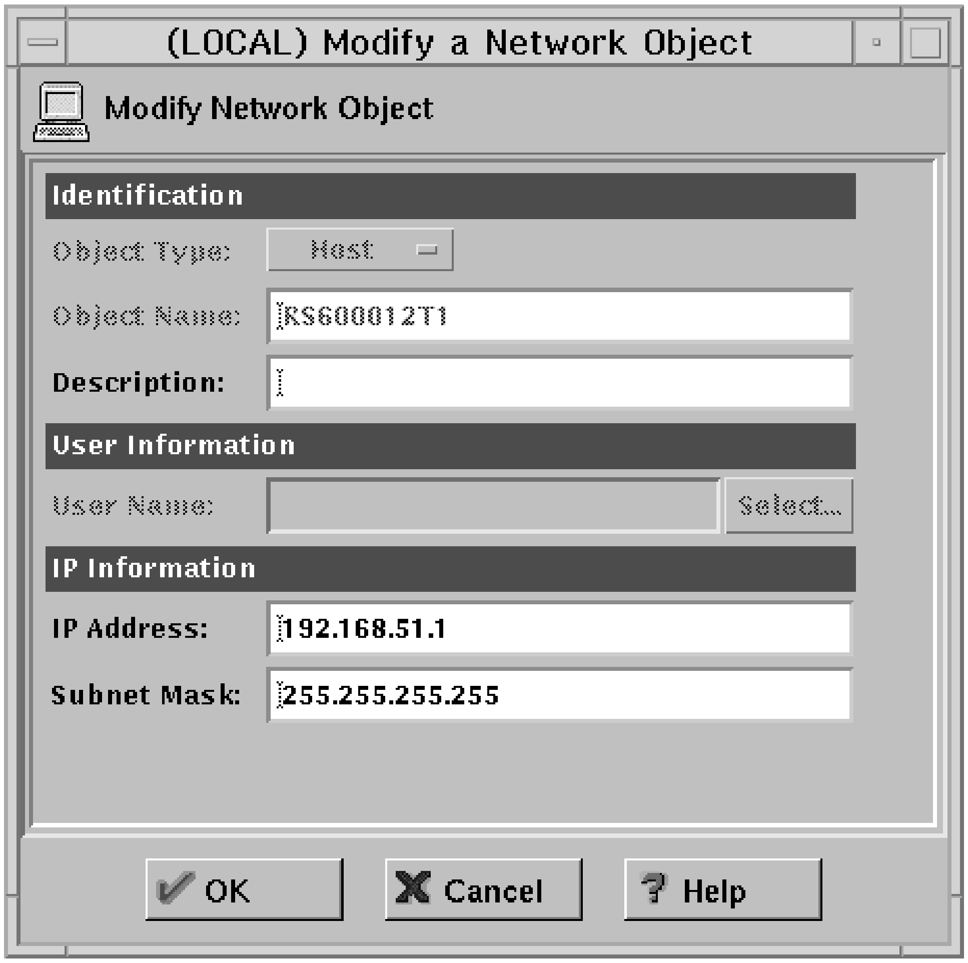
User Information

User Name:	

IP Information

IP Address:	192.168.50.1
Subnet Mask:	255.255.255.255

Figure 432. RS600012T0 Object Creation



Object Type: Host
Object Name: RS600012T1
Description:
User Name:
IP Address: 192.168.51.1
Subnet Mask: 255.255.255.255

Figure 433. RS600012T1 Object Creation

These two network objects are necessary because the communication between the client and the server is broken on the firewall when a proxy server is running.

We show now how we created the two new connections to enable the HTTP proxy server for IIOP over HTTP. The services list for IBM Firewall (see Figure 426 on page 460) already provides the two services that we need for these two connections:

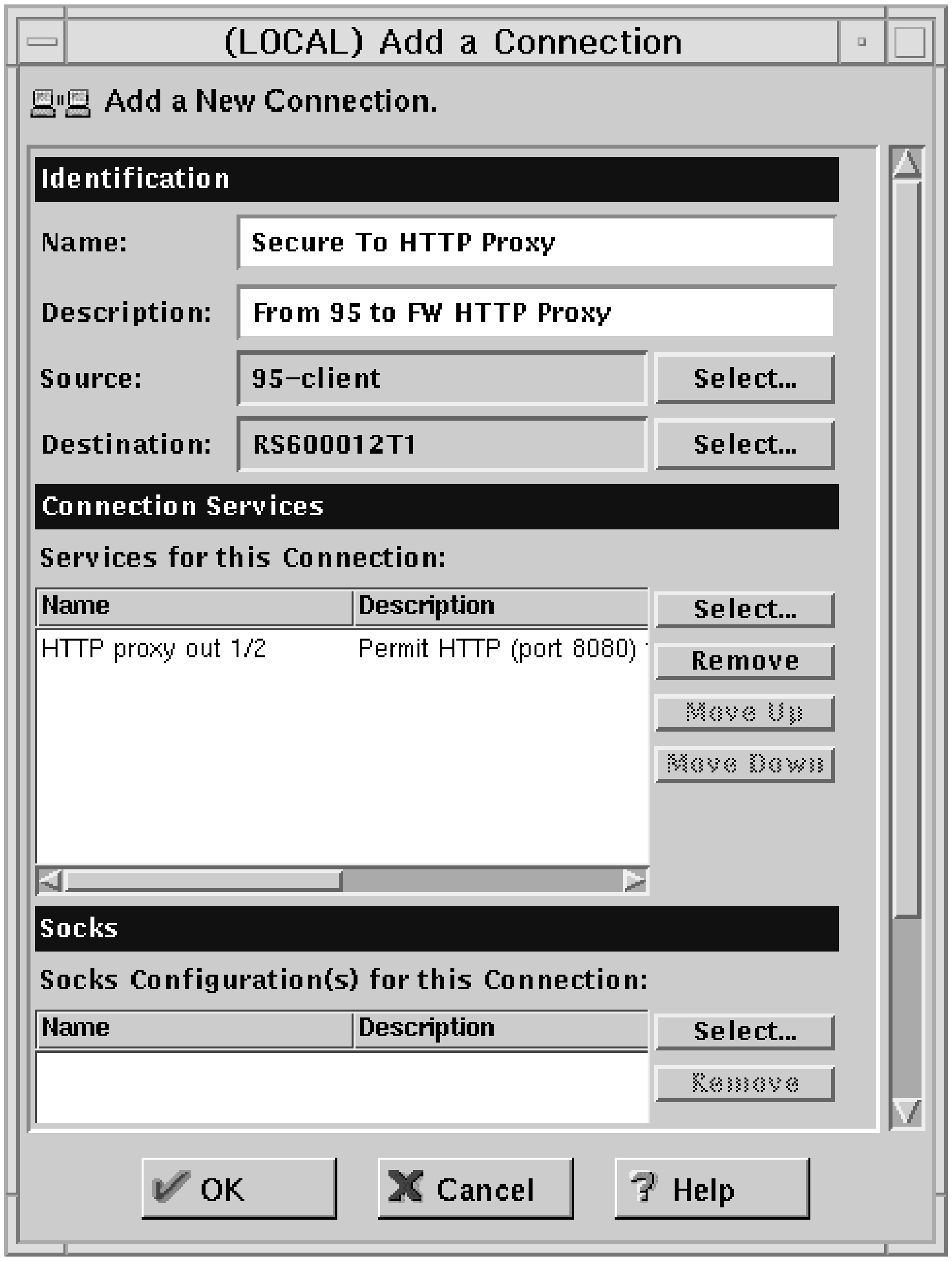
1. HTTP proxy out 1/2

   This service permits the HTTP proxy protocol from the client to the secure adapter of the firewall.

2. HTTP proxy out 2/2

   This protocol permits the HTTP protocol from the non-secure adapter of the firewall to the server.

So it was very simple to create the two connections using these predefined services. First of all, we had to select **Traffic Control** and then **Connection Setup** from the Configuration Client navigation tree, as shown in Figure 423 on page 457. The Connections List window was brought up (see Figure 424 on page 458), we selected **NEW** and then we pressed the **Open...** button in order to create a new connection. Once this operation was accomplished, we could see the Add a Connection window. We entered `Secure to HTTP Proxy` as the name for this connection and `From 95 to FW HTTP Proxy` as its description, as shown in the following window:



Name: Secure To HTTP Proxy

Description: From 95 to FW HTTP Proxy

Source: 95-client

Destination: RS600012T1

Services for this Connection:

Name	Description
HTTP proxy out 1/2	Permit HTTP (port 8080)

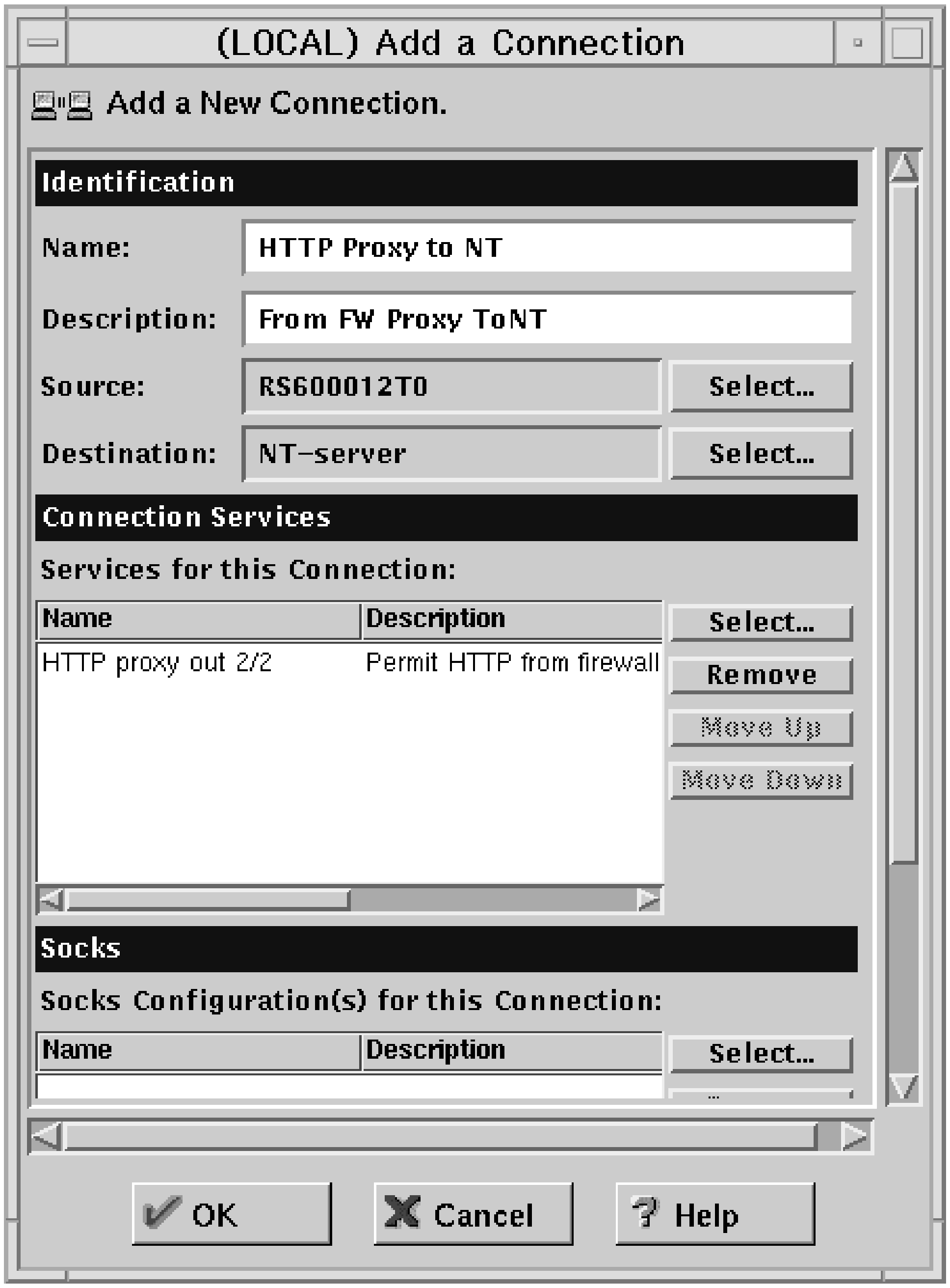
Socks Configuration(s) for this Connection:

Name	Description

Figure 434. Secure to HTTP Proxy Connection Definition

Notice that the source network object for this connection is 95-client (the Windows 95 client machine) while the destination one is RS600012T1 (the secure network adapter on the firewall). In general you can pick up the source and destination network objects by clicking **Select...** and then choosing the object you want from the Network Objects list (see Figure 420 on page 454). A similar procedure is used to include a service in the connection. In this case, we simply pressed **Select...** and then we were able to pick up **HTTP proxy out 1/2** from the Services list (see Figure 426 on page 460). Then, when we finished filling the fields in the Add a Connection window, we pressed **OK** and the new connection appeared in the Connections List window.

Following identical steps, we defined the second connection, naming it `HTTP Proxy to NT`, selecting **RS600012T0** (the non-secure network adapter on the firewall) as source network object, **NT-server** (the Web server machine) as the destination object and **HTTP proxy out 2/2** as the only service for this connection. Then we pressed **OK**, as shown in the following picture:


(LOCAL) Add a Connection

Add a New Connection.

Identification

Name:	HTTP Proxy to NT
Description:	From FW Proxy ToNT
Source:	RS600012T0
Destination:	NT-server

Connection Services

Services for this Connection:

Name	Description
HTTP proxy out 2/2	Permit HTTP from firewall

Socks

Socks Configuration(s) for this Connection:

Name	Description

Figure 435. HTTP Proxy to NT Connection Definition

You can double click **HTTP** from the Configuration Client navigation tree:

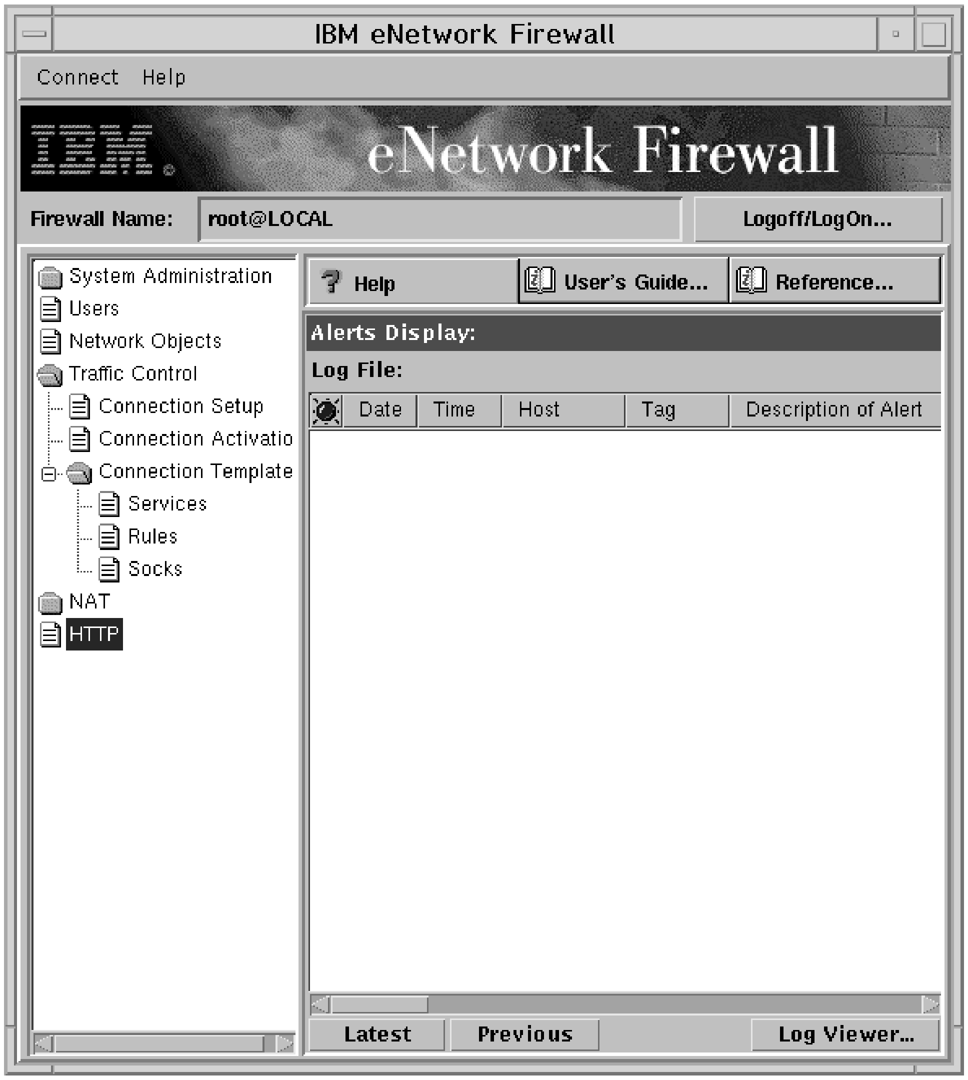
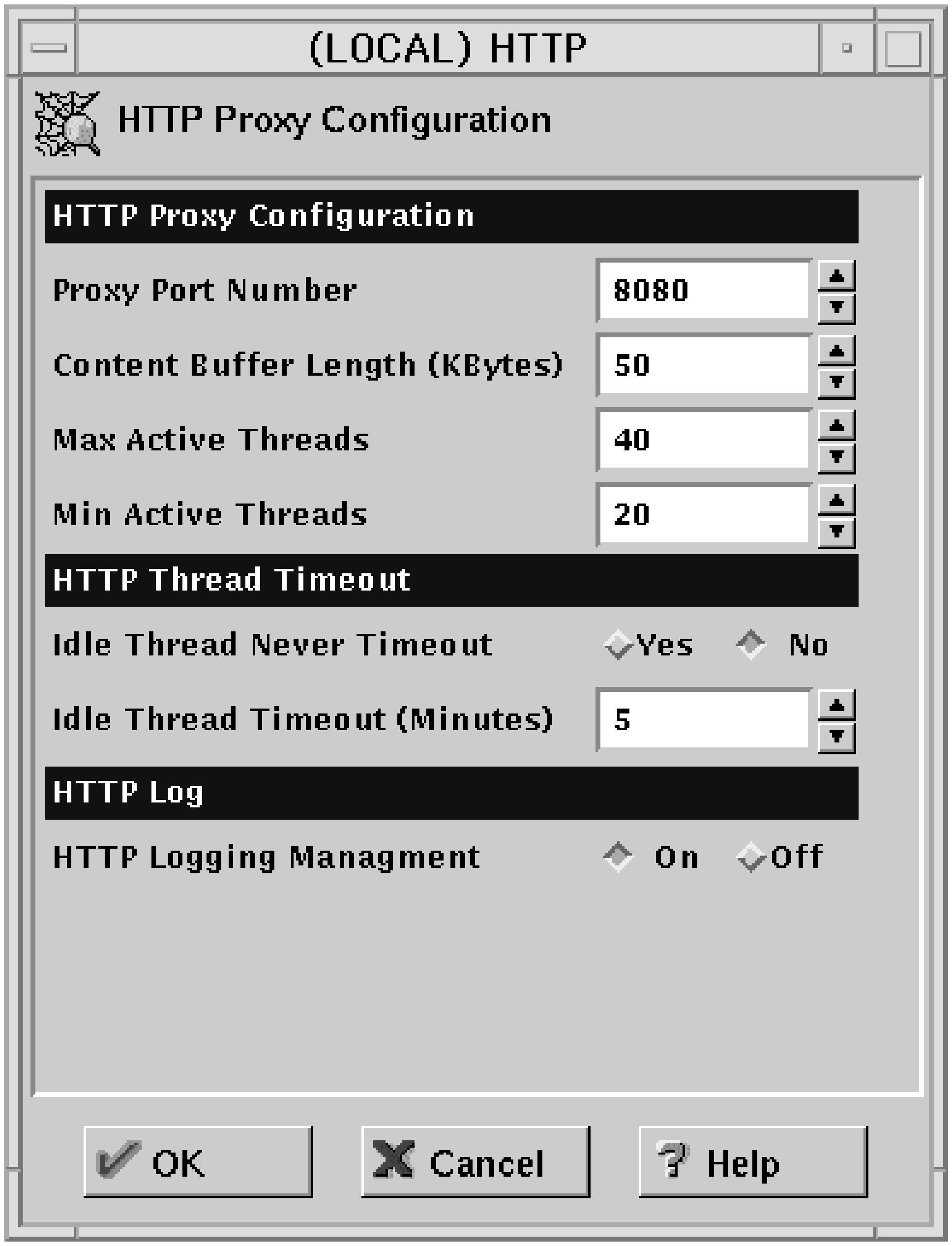


Figure 436. HTTP in the Configuration Client Navigation Tree

Then you can configure the HTTP proxy functionality through the HTTP Proxy Configuration window:



HTTP Proxy Configuration

Proxy Port Number	8080
Content Buffer Length (KBytes)	50
Max Active Threads	40
Min Active Threads	20

HTTP Thread Timeout

Idle Thread Never Timeout	Yes [✓ No]
Idle Thread Timeout (Minutes)	5

HTTP Log

HTTP Logging Managment	[✓ On] Off

Figure 437. HTTP Proxy Configuration

We accepted all the default values for this configuration, except that we changed the value for the Idle Thread Timeout field. This value specifies how many minutes the firewall should keep an idle thread available. Default is `forever`, meaning that the proxy never closes any idle threads. We changed this value to `5` minutes. Notice that by default the Proxy Port Number is 8080.

Once you finish setting up a new connection, it is necessary to activate it. To do this, the Connection Activation panel is automatically brought up (see Figure 428 on page 462).

In order for our firewall to work properly, we had to start the HTTP proxy daemon on our AIX firewall machine and so we simply entered the command

```
/usr/sbin/phttpd
```

It is also possible to start the HTTP proxy daemon at boot time. If you want to activate this function, you need to modify the /etc/rc.tcpip file: open that file and uncomment the line

```
/usr/sbin/phttpd
```

Note: In order to stop and restart the HTTP proxy daemon, you should follow these steps:

1. Discover what PID was assigned to the HTTP proxy daemon, by entering the command

   ```
   ps -ef | grep phttpd
   ```

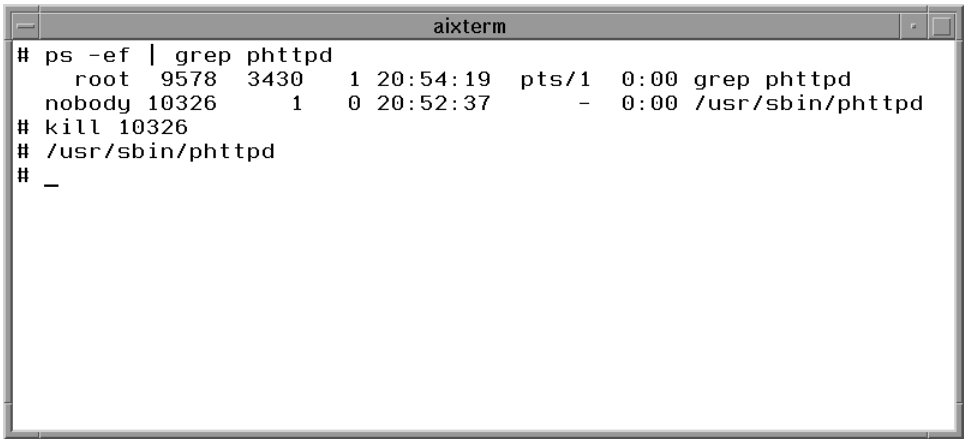
2. Kill that process using the `kill` command followed by the HTTP proxy daemon PID.

3. Enter the command

   ```
   /usr/sbin/phttpd
   ```

   in order to launch the HTTP proxy daemon again.

The following figure shows this sequence of commands:



```
# ps -ef | grep phttpd
    root  9578  3430   1 20:54:19  pts/1  0:00 grep phttpd
  nobody 10326     1   0 20:52:37      -  0:00 /usr/sbin/phttpd
# kill 10326
# /usr/sbin/phttpd
# _
```

Figure 438. How to Restart the HTTP Proxy Daemon

Notice that on the Windows NT platform we would have spoken of an HTTP proxy service rather than an HTTP proxy daemon and the way we would have started and stopped it would have been different. Moreover, the parameters shown in Figure 437 on page 472 would have been slightly different on a Windows NT platform.

9.7.2 Client/Server Communication through an HTTP Proxy Server

The AccountsUI applet is downloaded through the HTTP proxy server without any problem, since it is treated as any other component of the account.html Web page and it is transferred on the client machine because the HTML flow is permitted by the HTTP proxy server. Netscape Communicator informs us that the applet has been loaded without problems and the message

```
Applet com.ibm.jbroker.samples.account.AccountsUI loaded
```

is displayed at the bottom of the browser window.

What happens when the applet tries to connect back the Web server machine? Let's read the following code lines for the AccountsUI.java Java applet file:

```
// The object server servlet is assumed to be on the same machine as this applet
// was downloaded from, that is, host=AppletHost and port=AppletPort.
// The servlet name here is com.ibm.jbroker.GenericObjectServlet.

URL serverUrl = new URL("http", orb.getAppletHost(), orb.getAppletPort(),
                        "/servlet/com.ibm.jbroker.GenericObjectServlet");
```

Figure 439. URL Connection for the AccountsUI Applet

These code lines and the associated comments show that the serverUrl object is created by invoking the URL() constructor that takes four parameters:

1. `String protocol` - the name of the protocol
2. `String host` - the name of the remote host
3. `int port` - the port number
4. `String file` - the host file

The serverUrl object is used by the AccountsUI applet to invoke the GenericObjectServlet servlet in the Web server machine. The four parameters resolve as follows:

1. The protocol is HTTP.
2. The host is the same Web server machine from which the applet was downloaded. The method orb.getAppletHost() returns either the IP address or the host name of the Web server, depending on whether you invoked the account.html Web page by using the IP address of the Web server or its host name.
3. The port number is the same port number servicing the client applet. In our case, orb.getAppletPort() returned 80.
4. The host file is the fully qualified name of the servlet, that is com.ibm.jbroker.GenericObjectServlet.

This implies that the AccountsUI applet invokes the GenericObjectServlet servlet through one of the following URLs:

- http://wtr05218:80/servlet/com.ibm.jbroker.GenericObjectServlet

  if the HTML page was invoked through the host name of the Web server machine.

- http://192.168.50.2:80/servlet/com.ibm.jbroker.GenericObjectServlet

  if the HTML page was invoked through the IP address of the Web server machine.

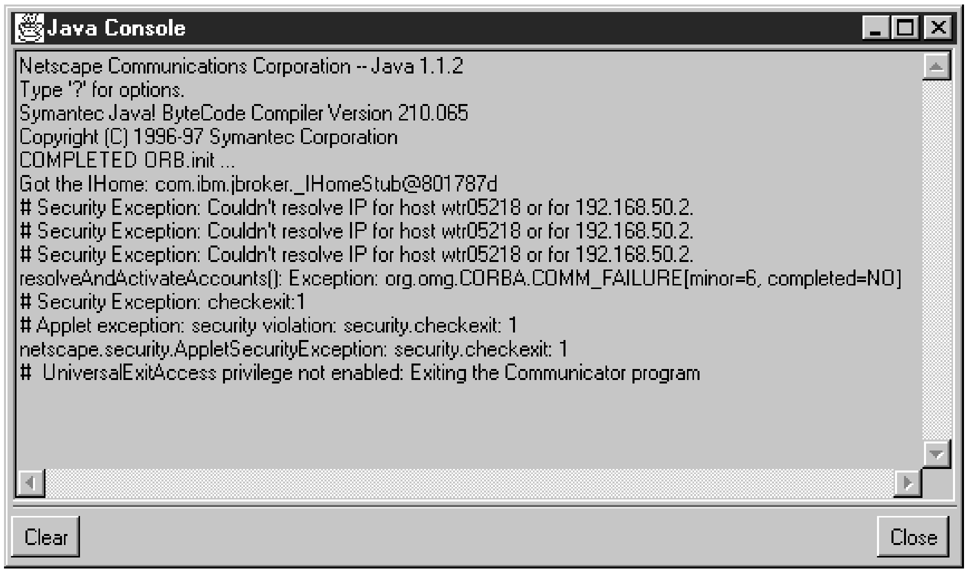
Notice that this applet makes use of the URL class from the java.net package in order to establish a network connection to the Web server from which it was downloaded.

But how can the applet request pass through the firewall when a proxy is defined? We demonstrated that this is possible (see 5.7, “Applet/Server Communication Through a Firewall” on page 258). If the DNS in your network environment is configured to translate host names into IP addresses for hosts located outside the firewall, this connection does not give any problem.

If no DNS has been configured in your network environment to translate host names into IP addresses for hosts located in the non-secure network, or if simply the firewall has been configured to disable DNS queries, then you can invoke the remote server only through its IP address. In fact in this situation, the applet will normally fail to make the desired connection and the SecurityManager throws an UnknownHostException, which is turned into a java.lang.SecurityException. If you disable DNS on your firewall, by unchecking **Permit DNS queries** in the Security Policy window (see Figure 417 on page 450), the following error message appears at the bottom of the browser window:

```
Applet com.ibm.jbroker.samples.account.AccountsUI can't start:
security violation: security checkexit: 1
```

The Java Console for Netscape Communicator immediately registers this exception, as the following figure demonstrates:



```
Netscape Communications Corporation -- Java 1.1.2
Type '?' for options.
Symantec Java! ByteCode Compiler Version 210.065
Copyright (C) 1996-97 Symantec Corporation
COMPLETED ORB.init ...
Got the IHome: com.ibm.jbroker._IHomeStub@801787d
# Security Exception: Couldn't resolve IP for host wtr05218 or for 192.168.50.2.
# Security Exception: Couldn't resolve IP for host wtr05218 or for 192.168.50.2.
# Security Exception: Couldn't resolve IP for host wtr05218 or for 192.168.50.2.
resolveAndActivateAccounts(): Exception: org.omg.CORBA.COMM_FAILURE[minor=6, completed=NO]
# Security Exception: checkexit:1
# Applet exception: security violation: security.checkexit: 1
netscape.security.AppletSecurityException: security.checkexit: 1
#  UniversalExitAccess privilege not enabled: Exiting the Communicator program
```

Figure 440. SecurityException in the Java Console

Notice that for three times the SecurityManager tried a connection, but each time it launched a SecurityException, as we can read in the Java Console:

```
SecurityException: Couldn't resolve IP for host wtr05218 or for 192.168.50.2.
```

This problem does not depend on the CORBA support used by our scenario, but it is generated each time an applet, invoked through the host name of the Webserver, tries an URL connection from behind a firewall acting like a proxy that does not provide host name resolution. In this situation, if you want to invoke the Web server through its host name, then your Netscape Communicator should be configured in order to disable the DNS lookup performed by the SecurityManager. As we explained in 5.7, “Applet/Server Communication Through a Firewall” on page 258, this operation can be performed by adding the following line to the prefs.js file

```
user_pref("security.lower_java_network_security_by_trusting_proxies", true);
```

when no copies of Communicator are currently running.

We have explained how the AccountsUI applet, running on the client machine, can invoke the GenericObjectServlet servlet, running on the Web server machine, and pass through a firewall implementing the HTTP proxy server technology.

The same considerations can be applied to the communication between the client and the server ORBs. We were informed by the WebSphere CORBA Support development team that in WebSphere Application Server Version 1.0 the client ORB uses an URLConnection object to establish a connection with the server ORB, as we already mentioned in 9.3, "Increasing Security with SSL" on page 442. When an URLConnection object is used to establish a connection with the Web server, this request is enabled to pass through a firewall implementing the HTTP proxy server technology (see 5.7, "Applet/Server Communication Through a Firewall" on page 258).

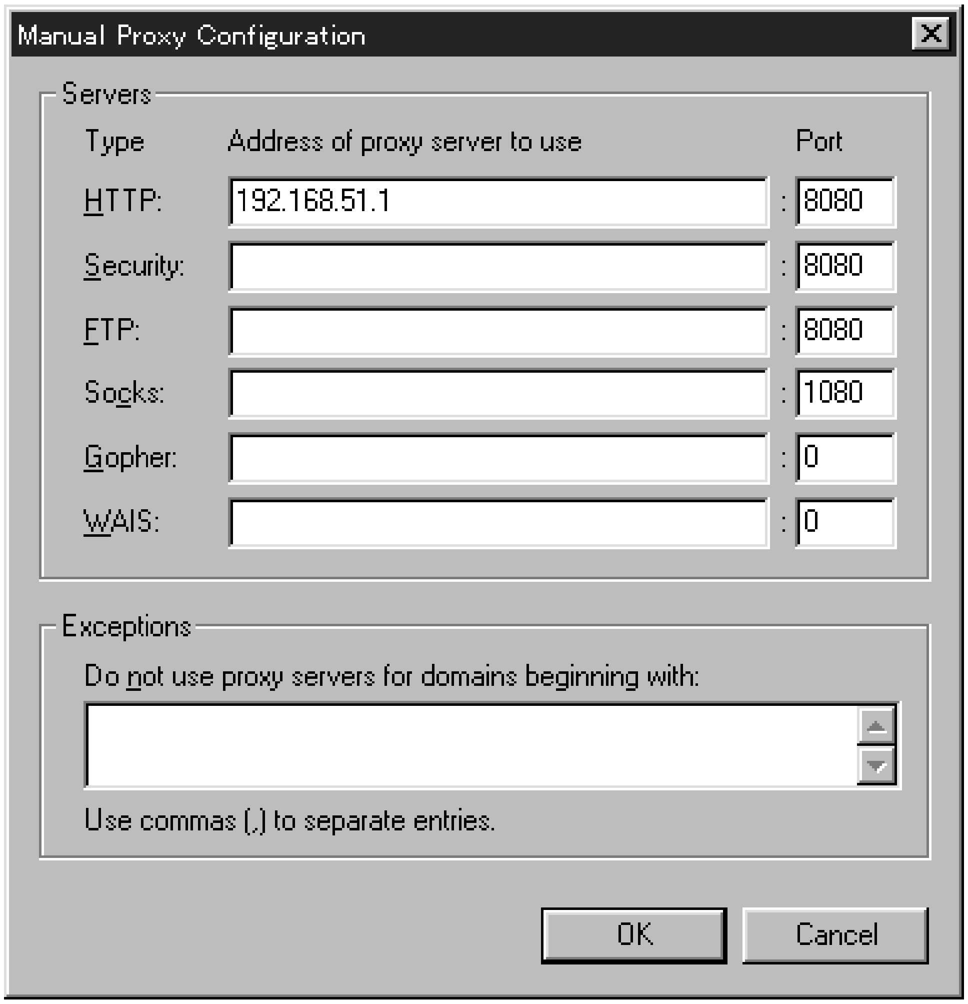
When we tested this CORBA application through the firewall, we were using the beta 3 release of ServletExpress, that presented a bug, in that the client/server ORB connection was performed via the class java.net.Socket, rather than java.net.URLConnection. As we explained in 5.7, "Applet/Server Communication Through a Firewall" on page 258, a Socket connection cannot be performed through a firewall implementing a proxy server. The problem is that support for proxies is part of the protocol that you are using above TCP/IP (such as HTTP, FTP, Gopher, etc.). It is therefore not possible to encapsulate the proxy specific stuff at the socket layer. For this reason our tests with the HTTP proxy server did not work correctly. The AccountsUI applet was able to connect the GenericObjectServlet (see Figure 439 on page 474), since in this case the URLConnection object was used, but the client ORB failed to connect the server ORB via the Socket class. So, in order for this application to work through a firewall, it became necessary in the ServletExpress beta 3 CORBA Support to add the IP filter connection `for 80 port` (see 2 on page 452 and Figure 425 on page 459) beside the HTTP proxy server connections (see 2 on page 466). The client/server HTTP communication passed through the HTTP proxy server running on the firewall, but the IIOP over HTTP communication between the two ORBs bypassed the HTTP proxy server entirely and used the direct `for 80 port` connection.

As we have mentioned, in the WebSphere CORBA Support Version 1.0 code, this problem has been fixed and the client ORB invokes the server ORB via the URLConnection class, rather than the Socket class. In this new version, opening up the HTTP port 80 through the firewall is no longer necessary. The new code works correctly through a firewall implementing an HTTP proxy server. Several tests have been done also by the CORBA Support development team and they have all been successful.

9.7.3 Netscape Communicator Advanced Configuration for Using HTTP Proxy Server

After the firewall configuration was in place, we were almost ready to run the sample Account application. In order to pass HTTP through the firewall using the HTTP proxy server, the client's browser must be configured properly, so that, when it invokes the Web server using the standard HTTP port 80, this request is captured by the HTTP proxy daemon on the firewall machine, that submits it on behalf of the client.

To configure Netscape Communicator 4.05, it is enough to open the Edit menu and select **Preferences...**. The Preferences window comes up. Select **Advanced** and then **Proxy** in the Preferences tree. Check the **Manual Proxy Configuration** radio button and click **View**. You will then be permitted to fill the HTTP proxy server field by entering the IP address of the firewall machine that you intend to use as HTTP proxy server, as we did in the following window:


Manual Proxy Configuration
Servers
Exceptions
Do not use proxy servers for domains beginning with:
Use commas (,) to separate entries.
OK
Cancel

Type	Address of proxy server to use	Port
HTTP:	192.168.51.1	8080
Security:		8080
FTP:		8080
Socks:		1080
Gopher:		0
WAIS:		0

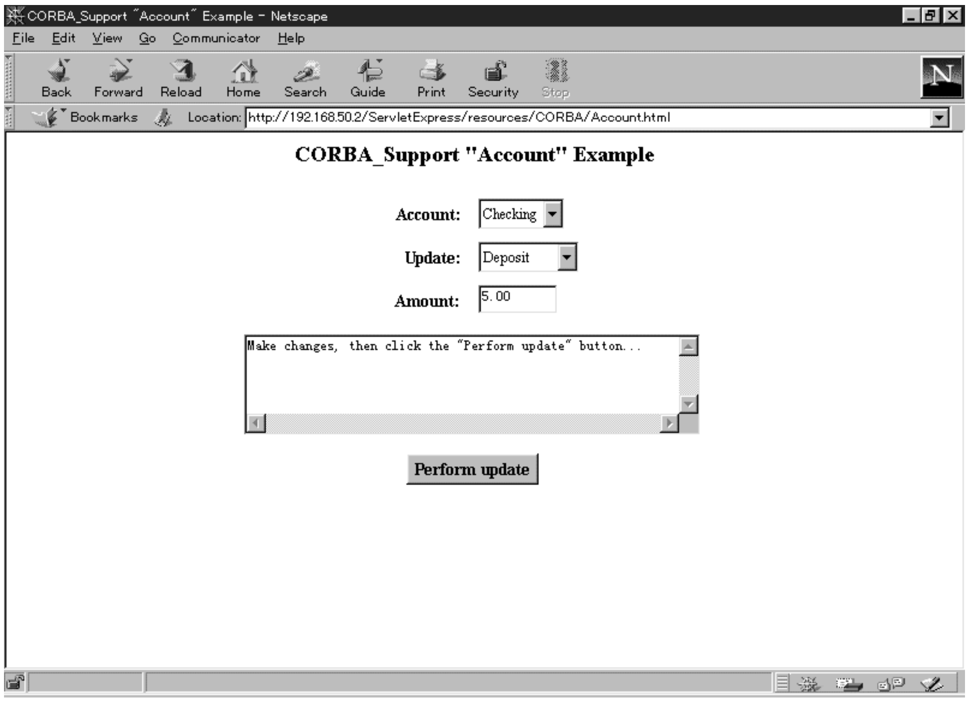
Figure 441. Netscape Configuration for Using HTTP Proxy Server.

Notice that the HTTP proxy server was configured to listen on port 8080 (see Figure 437 on page 472) and this value had to be typed in the Port field. After this simple step, you simply have to click **OK**.

9.7.4 Testing the Application through the HTTP Proxy Server

When all the described configurations were in place, we were finally able to launch the Account application and to test how it worked through the HTTP proxy server we had implemented.

The result was successful and in fact we got the following window on the client machine:


CORBA_Support "Account" Example - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://192.168.50.2/ServletExpress/resources/CORBA/Account.html

CORBA_Support "Account" Example

Account: Checking

Update: Deposit

Amount: 5.00

```
Make changes, then click the "Perform update" button...
```

Perform update

Figure 442. Account Sample Program Running Through HTTP Proxy Server

9.8 SOCKS Server for HTTP and IIOP over HTTP Scenario

This section describes how to use SOCKS server firewall technology to enable HTTP and IIOP over HTTP to pass through the firewall. We experimented with this technology with both Versions 3.1.1 and 3.2 of IBM Firewall. You will find more information about the SOCKS server technology in 6.1.4, “SOCKS Servers” on page 284.

9.8.1 How to Configure SOCKS Server for HTTP and IIOP over HTTP

We now show the steps we followed to configure our firewall to act as a SOCKS server.

First of all, we needed to specify objects and connections in this scenario. We had already defined the correct four network objects listed in 1 on page 466. It was possible to keep those definitions for this scenario.

However we had to define and use three new connections:

- Secure To SOCKS

  We created this new connection in order to pass the SOCKS protocol from the 95-client object to the RS600012T1 object.

- SOCKS To Nonsecure

  We created this new connection in order to pass the HTTP protocol from the RS600012T0 object to the NT-server object.

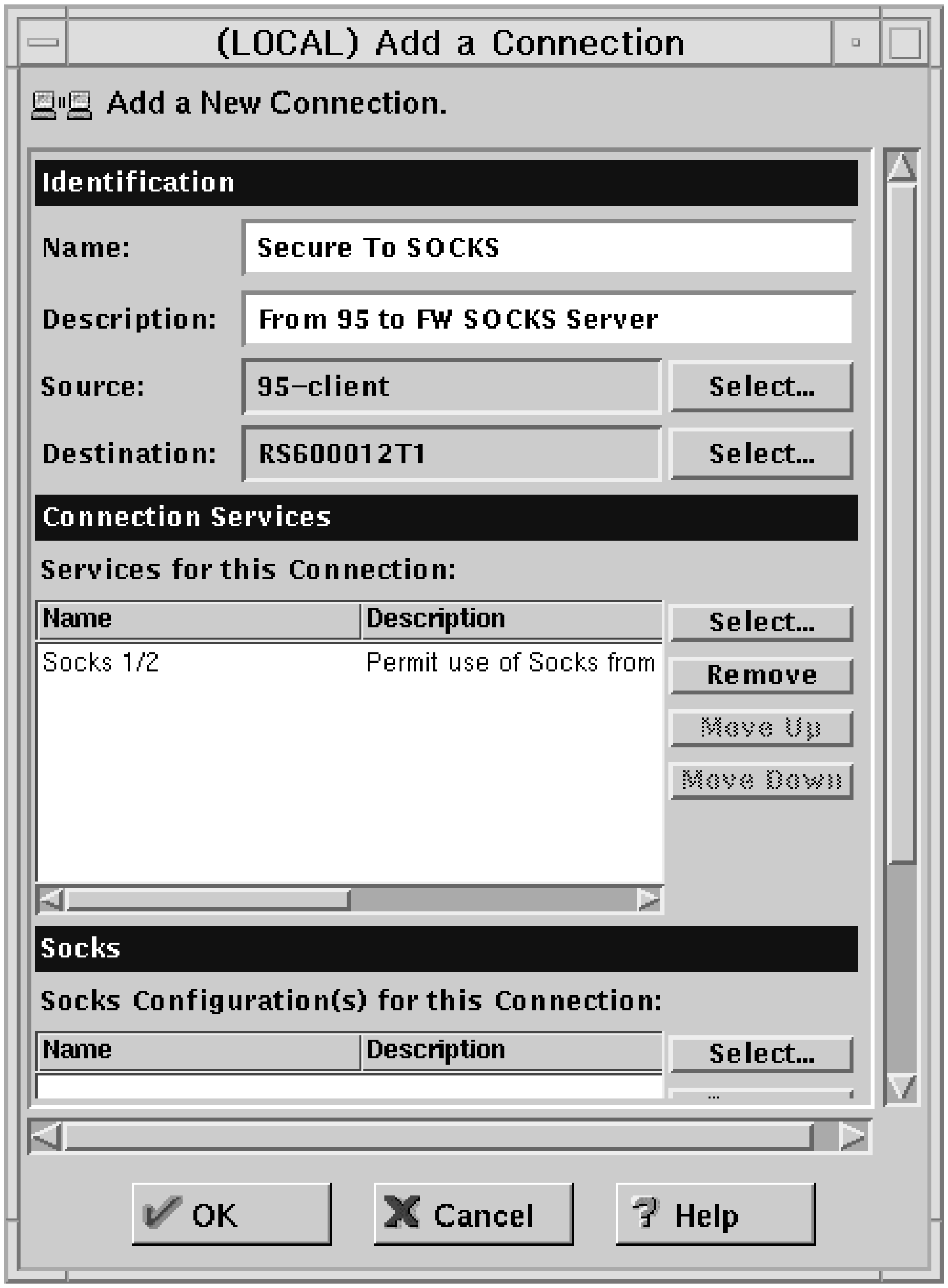
- SOCKS Config for HTTP

  We created this new connection in order to pass the SOCKSified HTTP protocol from the 95-client object to the NT-server object.

The SOCKSified HTTP protocol that we mentioned when introducing the `SOCKS Config for HTTP` connection is used to wrap the HTTP protocol with the SOCKS protocol. We described this process in 6.1.4, "SOCKS Servers" on page 284.

We created these new connections by selecting **NEW** in the Connections List window, as shown in Figure 424 on page 458.

As usual, we had to select the appropriate source and destination network objects and the appropriate services, before actually adding the new connections. The two following windows show how we added the two new connections named `Secure To SOCKS` and `SOCKS To Nonsecure`.



Name: Secure To SOCKS

Description: From 95 to FW SOCKS Server

Source: 95-client

Destination: RS600012T1

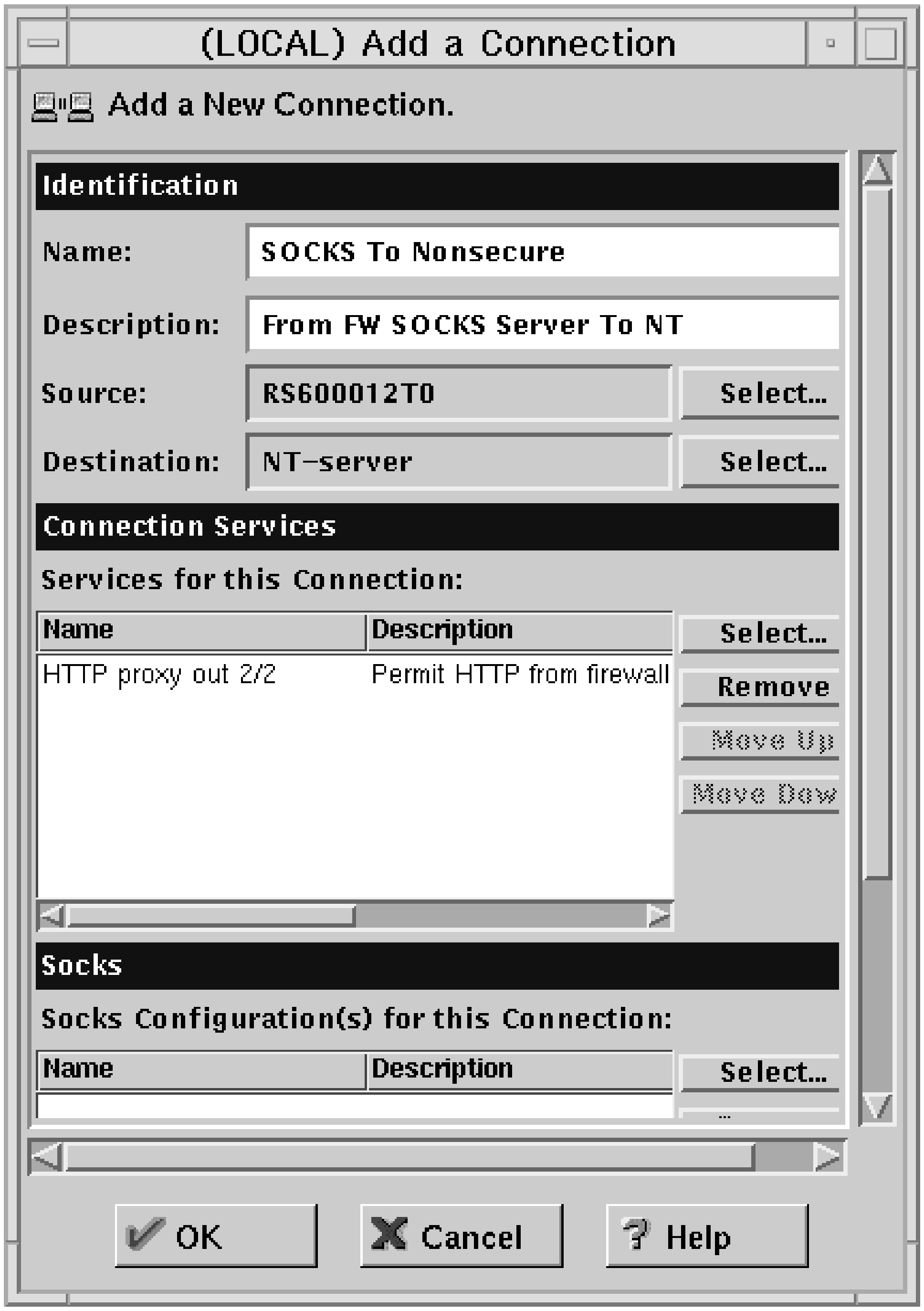
Services for this Connection:

Name	Description
Socks 1/2	Permit use of Socks from

Socks Configuration(s) for this Connection:

Name	Description

Figure 443. SOCKS Connection from the Client to the Secure Adapter of the Firewall


(LOCAL) Add a Connection

Add a New Connection.

Identification

Name: SOCKS To Nonsecure

Description: From FW SOCKS Server To NT

Source: RS600012T0

Destination: NT-server

Connection Services

Services for this Connection:

Name	Description
HTTP proxy out 2/2	Permit HTTP from firewall

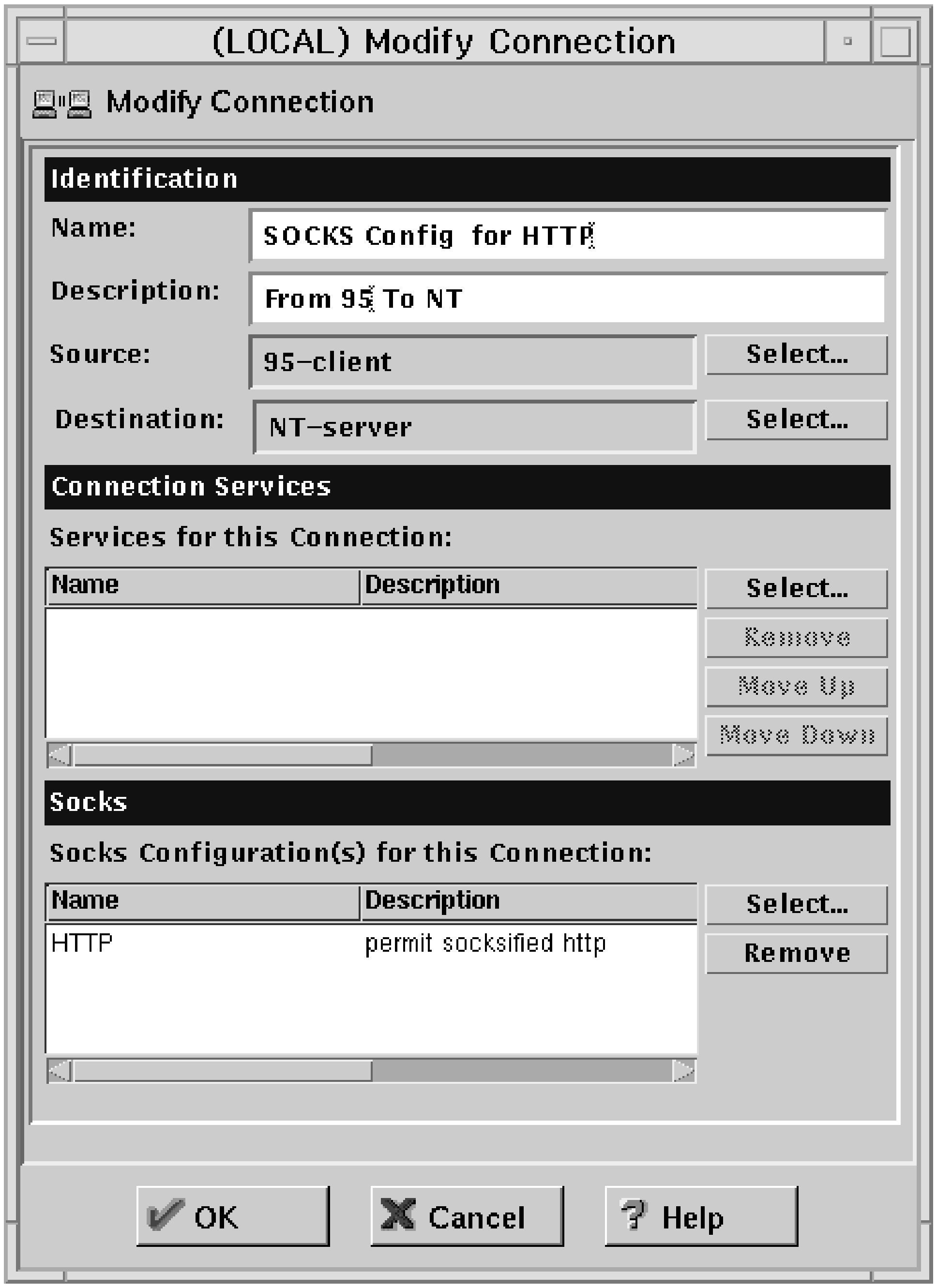
Socks

Socks Configuration(s) for this Connection:

Name	Description

Figure 444. HTTP Connection from the Nonsecure Adapter of the Firewall to the Server

After that, we added the connection to pass the SOCKSified protocol from the client to the server through the firewall, as you can see in the following screen:



(LOCAL) Modify Connection

Modify Connection

Identification

Name:	SOCKS Config for HTTP
Description:	From 95 To NT
Source:	95-client
Destination:	NT-server

Connection Services

Services for this Connection:

Name	Description

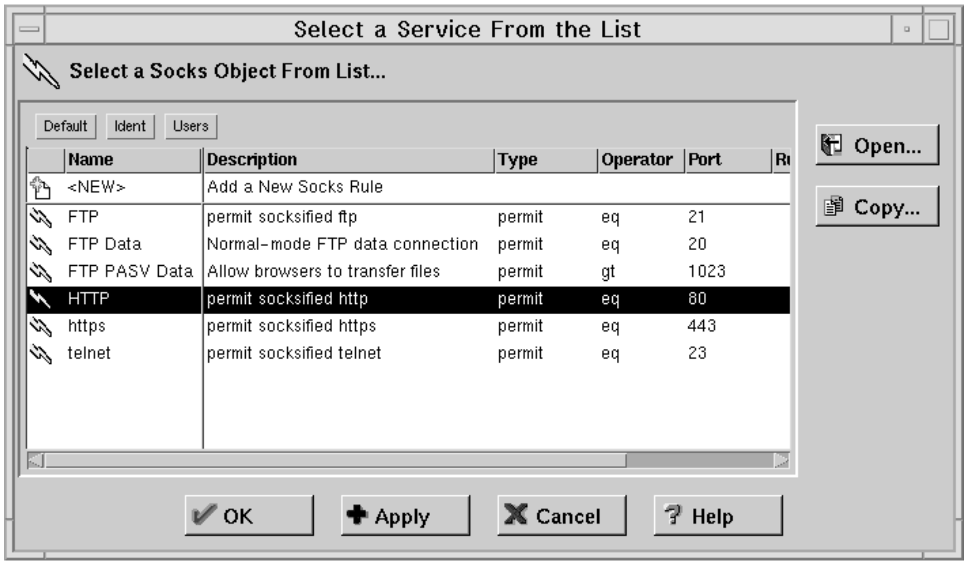
Socks

Socks Configuration(s) for this Connection:

Name	Description
HTTP	permit socksified http

Figure 445. SOCKSified HTTP Connection from the Client to the Server

It is important to notice that IBM Firewall already provides the predefined SOCKS object that must be included in this connection in order to permit the SOCKSified protocol through the firewall. In the above window, it was enough to press **Select...** in the Socks section. The following window appeared:


Select a Service From the List
Select a Socks Object From List...
Default
Ident
Users
Open...
Copy...
OK
Apply
Cancel
Help

Name	Description	Type	Operator	Port
<NEW>	Add a New Socks Rule			
FTP	permit socksified ftp	permit	eq	21
FTP Data	Normal-mode FTP data connection	permit	eq	20
FTP PASV Data	Allow browsers to transfer files	permit	gt	1023
HTTP	permit socksified http	permit	eq	80
https	permit socksified https	permit	eq	443
telnet	permit socksified telnet	permit	eq	23

Figure 446. Selecting the Socks Object from the List

We simply selected **HTTP** in the Socks Objects list and then we clicked **OK**. The HTTP SOCKS object appeared as shown in Figure 445 on page 481 and then we pressed **OK**. We were then permitted to add the new connections.

Notice that by default the SOCKS server on the firewall machine listens on TCP port 1080.

Once the three new connections that we have described had been created, they also had to be activated. To do this, make sure that these three connections are the only ones that appear in the Connections List window and then activate them through the Connection Activation window, as we have done several times in this chapter.

9.8.2 Client/Server Communication through a SOCKS Server

When a firewall implements the SOCKS server technology, the situation is very similar to that for proxies (see 9.7.2, “Client/Server Communication through an HTTP Proxy Server” on page 473). If the client applet or ORB activates a connection through the java.net.URLConnection class - as happens with the WebSphere CORBA Support Version 1.0 - there is no negative impact on users behind the firewall.

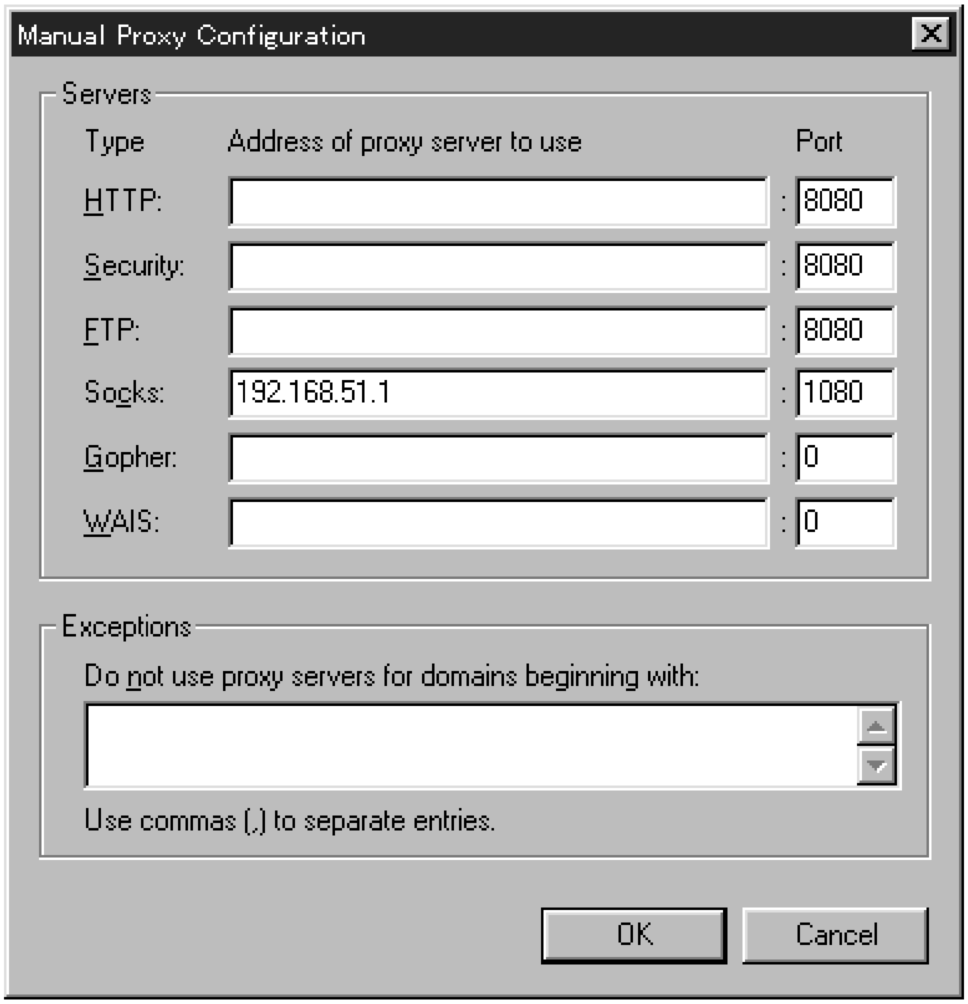
However, as we mentioned in 9.7.2, “Client/Server Communication through an HTTP Proxy Server” on page 473, we did not have the opportunity to use Version 1.0 of WebSphere Application Server. Our experiments were based upon ServletExpress Version 1.0 beta 3. In that case the connection between the AccountsUI applet and the GenericObjectServlet worked correctly, since it was accomplished through the URLConnection class, but the client ORB was not able to connect the server ORB, since this connection was implemented via the java.net.Socket class. For this reason it was not possible to control the IIOP over HTTP stream with the SOCKS server technology and it became necessary to open up TCP port 80 through the firewall in order to permit the IIOP over HTTP communication. The IP filter connection named `for 80 port`, described in 2 on page 452 and defined in Figure 425 on page 459, had to be added beside the three SOCKS connections that we had defined.

This problem, as we mentioned in 9.7.2, “Client/Server Communication through an HTTP Proxy Server” on page 473, has been removed in the WebSphere CORBA Support coming with WebSphere Application Server Version 1.0. The java.net.URLConnection class is now used also by the client ORB to activate a connection to the server ORB, so that opening up the HTTP port 80 through the firewall should no longer be necessary.

As we mentioned in 5.7, “Applet/Server Communication Through a Firewall” on page 258, with the way SOCKS works it should be possible to put SOCKS support in the java.net.Socket code, resulting in an encapsulation of the SOCKS protocol layer and allowing the enforcement of the security policy without undue negative impact on applications running behind the firewall, when the firewall acts as a SOCKS server. This has been done for JDK 1.0.2, but unfortunately not in Netscape. So, if your network environment has a SOCKS server, then everything works fine as long as you use the Applet Viewer or java.net.URLConnection, but using java.net.Socket under Netscape will give you a SecurityException. This is what we also tested in our network environment.

9.8.3 Netscape Communicator Advanced Configuration for Using SOCKS Server for HTTP

Netscape Communicator must be configured to use the SOCKS server on the firewall machine. You should open the Manual Proxy Configuration window as indicated in 9.7.3, “Netscape Communicator Advanced Configuration for Using HTTP Proxy Server” on page 476 and then fill the Socks field by typing in either the IP address or the host name of the firewall machine. We typed in `191.168.51.1`, which was the IP address assigned to the secure network adapter on the firewall machine. It is also necessary to specify the exact port used by the SOCKS server, which is by default 1080. We typed this value in the Port field related to Socks, then we pressed **OK**, as shown in the following screen:



Manual Proxy Configuration

Servers

Type	Address of proxy server to use	Port
HTTP:		8080
Security:		8080
FTP:		8080
Socks:	192.168.51.1	1080
Gopher:		0
WAIS:		0

Exceptions

Do not use proxy servers for domains beginning with:

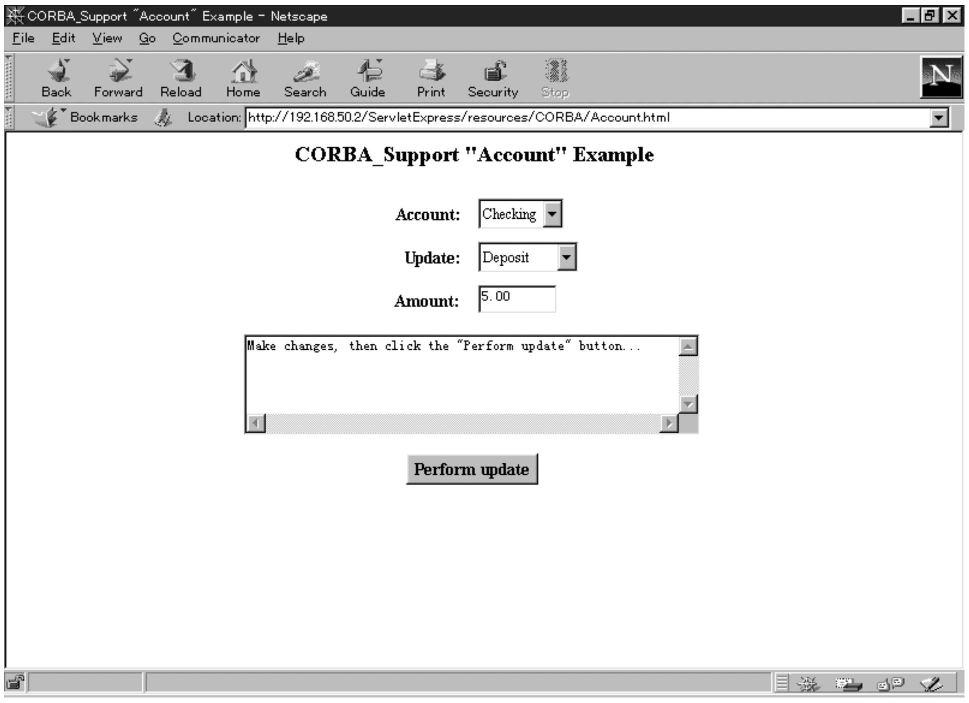
Use commas (,) to separate entries.

Figure 447. Netscape Communicator Configuration for Using SOCKS Server

9.8.4 Testing the Application through the SOCKS Server

The application worked correctly as soon as we pointed the client's browser to the URL http://192.168.50.2/ServletExpress/resources/CORBA/account.html. Of course, the IP address of the Web server machine could have also been replaced by the host name, because DNS had been configured in our intranet.

The Account application ran successfully through the firewall that we had configured as SOCKS server and we got the following window on the client's machine:


CORBA_Support "Account" Example - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://192.168.50.2/ServletExpress/resources/CORBA/Account.html

CORBA_Support "Account" Example

Account: Checking

Update: Deposit

Amount: 5.00

```
Make changes, then click the "Perform update" button...
```

Figure 448. Account Sample Program

Chapter 10. SSL Tunneling and SOCKS Server for HTTPS Scenarios

This chapter will describe how to implement the security firewall technologies SSL tunneling and SOCKS server in network environments where client/server communication is encrypted via the SSL protocol. In order to permit SSL to flow through the firewall, you could also implement an IP filter-based configuration of the firewall. This is shown in Chapter 8, “Three-Tier Applications in Firewall-Protected Network Environments” on page 361.

The hardware, network and software configuration that we use here is the same as in Chapter 9, “IIOP in Firewall-Protected Network Environments” on page 435. You can refer to 9.5, “Firewall Basic Configuration” on page 444 and Figure 406 on page 435 to see all the details. However, we could not use the SSL protocol to encrypt the IIOP over HTTP stream and get IIOP over HTTPS, since both ServletExpress beta 3 and WebSphere Application Server Version 1.0 do not allow the IIOP over HTTP flow to be encrypted through the SSL protocol (see 9.3, “Increasing Security with SSL” on page 442). We used the SSL protocol to encrypt only the HTTP communication between client and server in a two-tier NCF environment.

SSL Setup, Web Server Authentication and Client Authentication

In order to use the SSL protocol, it was necessary to set up Lotus Domino Go Webserver as a secure Web server, meaning that it had to be able to use the SSL protocol and present its certificate to each user that invoked a secure communication through an `https://` URL. We describe the process to set up Domino Go Webserver for using SSL in 3.1.2, “Lotus Domino Go Webserver SSL Setup” on page 71, while in 3.1.3, “Lotus Domino Go Webserver SSL Server Authentication” on page 79 we show how the Web server can be authenticated by the client. Both these operations are necessary if you want encrypted client/server communication using the SSL protocol.

Actually, SSL Versions 2.0 and 3.0 also provide support for client authentication to the Web server. This process is not mandatory and a secure connection between client and server can be established anyway, even if the server does not request client authentication. To enforce the security measures of this scenario, we set up our Web server to request client authentication. You can refer to 3.1.4, “SSL Client Authentication” on page 84 to see how to configure Lotus Domino Go Webserver to request client authentication.

Notice that the operating system where we had installed the firewall was AIX. Some of the considerations that you read in this chapter are then AIX-specific and could be different on a Windows NT platform.

10.1 SSL Tunneling Scenario

When a Web browser on a client machine connects using an `https://` URL to a secure Web server that is equipped with the SSL function, client/server communication uses the SSL protocol to encrypt the HTTP flow. If the two machines are located in different networks separated by a firewall, it is necessary to pass SSL through the firewall. In general, to do this, we can use SOCKS technology or SSL tunneling. In this section we will see how to use the SSL tunneling technology to pass SSL through the firewall. If you want to see how to use the SOCKS technology, refer to 10.2.1, “How to Configure SOCKS Server for HTTPS” on page 494.

SSL tunneling is a famous technology provided by Netscape Communications Corporation. Netscape Communicator 4.05 (which is the version of Communicator that we used in our scenario) supports SSL tunneling. You can find more information about SSL tunneling in 6.1.5, “Understanding SSL Tunneling” on page 287.

In this scenario we used IBM eNetwork Firewall 3.2, which supports SSL tunneling. You can also use IBM Firewall 3.1.1, which supports SSL tunneling provided you apply the fix IR36656 downloadable from http://www.ics.raleigh.ibm.com/firewall/fixes.htm.

10.1.1 How to Configure SSL Tunneling

In order to use SSL tunneling, we needed to define the following objects and connections:

1. Objects (Type: Host)
   - 95-client

     This object was a Windows 95 IBM ThinkPad located in the secure subnetwork of our intranet. Netscape Communicator 4.05 was installed on this machine as Java-enabled Web browser. This object played the role of the client in this scenario. The applet and the client side ORB ran on this machine.
   - NT-server

     This object was a Windows NT Server 4.0 IBM Personal Computer, located in the low-level security subnetwork. Lotus Domino Go Webserver 4.6.2.2 was installed on this machine and ServletExpress 1.0 beta 3 provided the servlet engine. This object played the key role of the Web server in this scenario. The sample application and the server side ORB ran on this machine.
   - RS600012T0

     This object is the non-secure network interface on the firewall machine to which IP address 192.168.50.1 was assigned.
   - RS600012T1

     This object is the secure network interface on the firewall machine to which IP address 192.168.51.1 was assigned.
2. Connections
   - HTTP Proxy to NT

We created this new connection in order to pass the HTTP protocol from the RS600012T0 object to the NT-server object through the firewall.

- Secure to HTTP Proxy

  We created this new connection in order to pass the HTTP protocol from the 95-client object to the RS600012T1 object through the firewall.

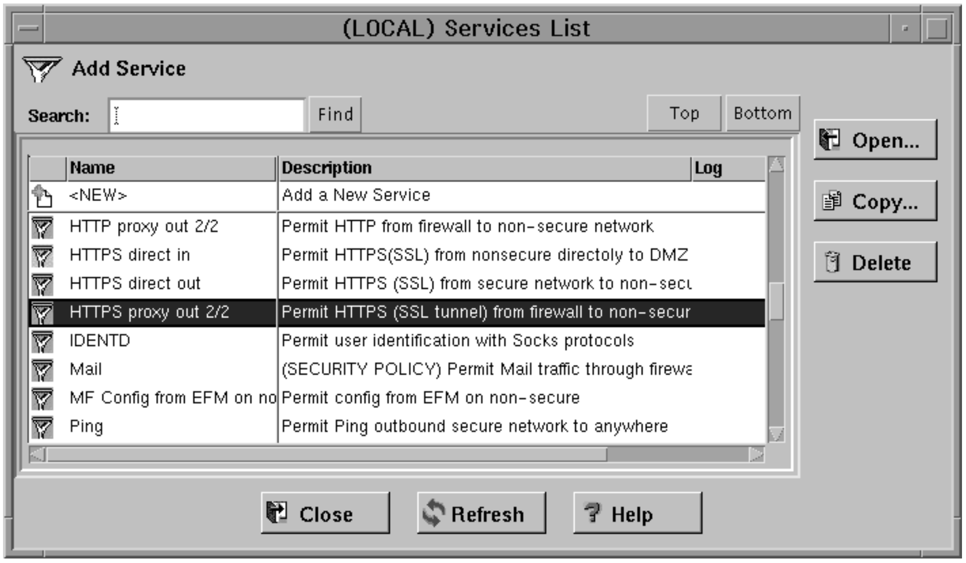
- HTTP Proxy to NT for SSL

  We created this new connection to permit the SSL protocol from the RS600012T0 to the NT-server object.

The two connections `HTTP Proxy to NT` and `Secure to HTTP Proxy` had already been defined (see Figure 434 on page 469 and Figure 435 on page 470). We used those two connections to implement the HTTP proxy server technology on our firewall machine and we are restoring these two connections now because, even if we want SSL to be tunneled through the firewall, we also want to permit HTTP to flow through the firewall. The reason for this is because a lot of times `https://` URLs are hit from Web pages that have been accessed through `http://` URLs, and you might want users in the protected network to be able to use the HTTP protocol.

In other words, our firewall will act as an HTTP proxy server using the two connections `HTTP Proxy to NT` and `Secure to HTTP Proxy`, but it will also be able to tunnel SSL using the connection `HTTP Proxy for NT to SSL` that we still have to define.

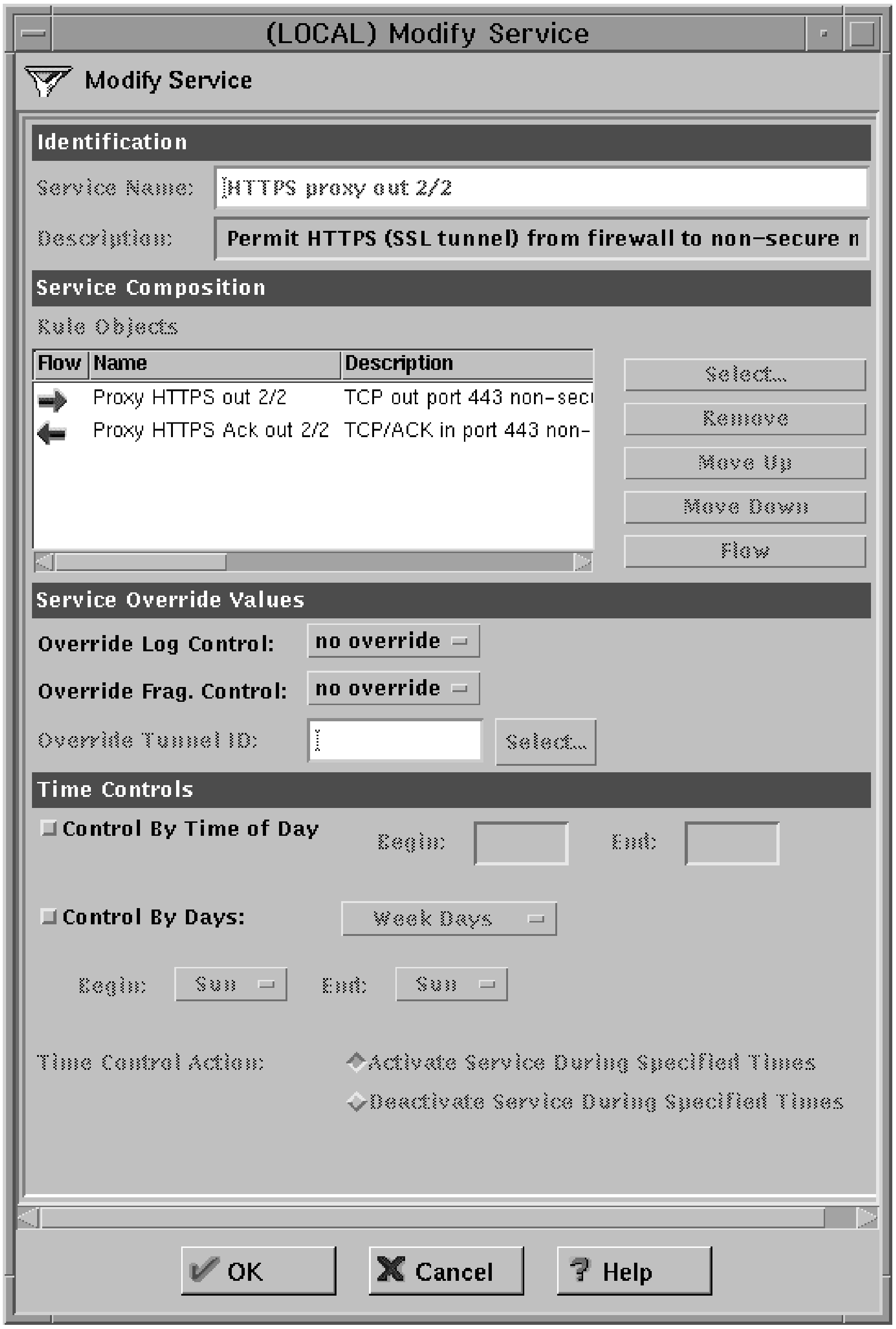
IBM eNetwork Firewall 3.2 provides a predefined service for SSL tunneling in the services list. This service is named **HTTPS Proxy out 2/2**:



Name	Description	Log
<NEW>	Add a New Service	
HTTP proxy out 2/2	Permit HTTP from firewall to non-secure network	
HTTPS direct in	Permit HTTPS(SSL) from nonsecure directoly to DMZ	
HTTPS direct out	Permit HTTPS (SSL) from secure network to non-secu	
HTTPS proxy out 2/2	Permit HTTPS (SSL tunnel) from firewall to non-secur	
IDENTD	Permit user identification with Socks protocols	
Mail	(SECURITY POLICY) Permit Mail traffic through firewa	
MF Config from EFM on no	Permit config from EFM on non-secure	
Ping	Permit Ping outbound secure network to anywhere	

Figure 449. SSL Tunneling Service in the Services List

The following figure shows the definition page for this service:


(LOCAL) Modify Service

Modify Service

Identification

Service Name: HTTPS proxy out 2/2

Description: Permit HTTPS (SSL tunnel) from firewall to non-secure n

Service Composition

Rule Objects

Flow	Name	Description
➡	Proxy HTTPS out 2/2	TCP out port 443 non-secı
⬅	Proxy HTTPS Ack out 2/2	TCP/ACK in port 443 non-

Select... Remove Move Up Move Down Flow

Service Override Values

Override Log Control: no override

Override Frag. Control: no override

Override Tunnel ID:

Time Controls

Control By Time of Day Begin: End:

Control By Days: Week Days

Begin: Sun End: Sun

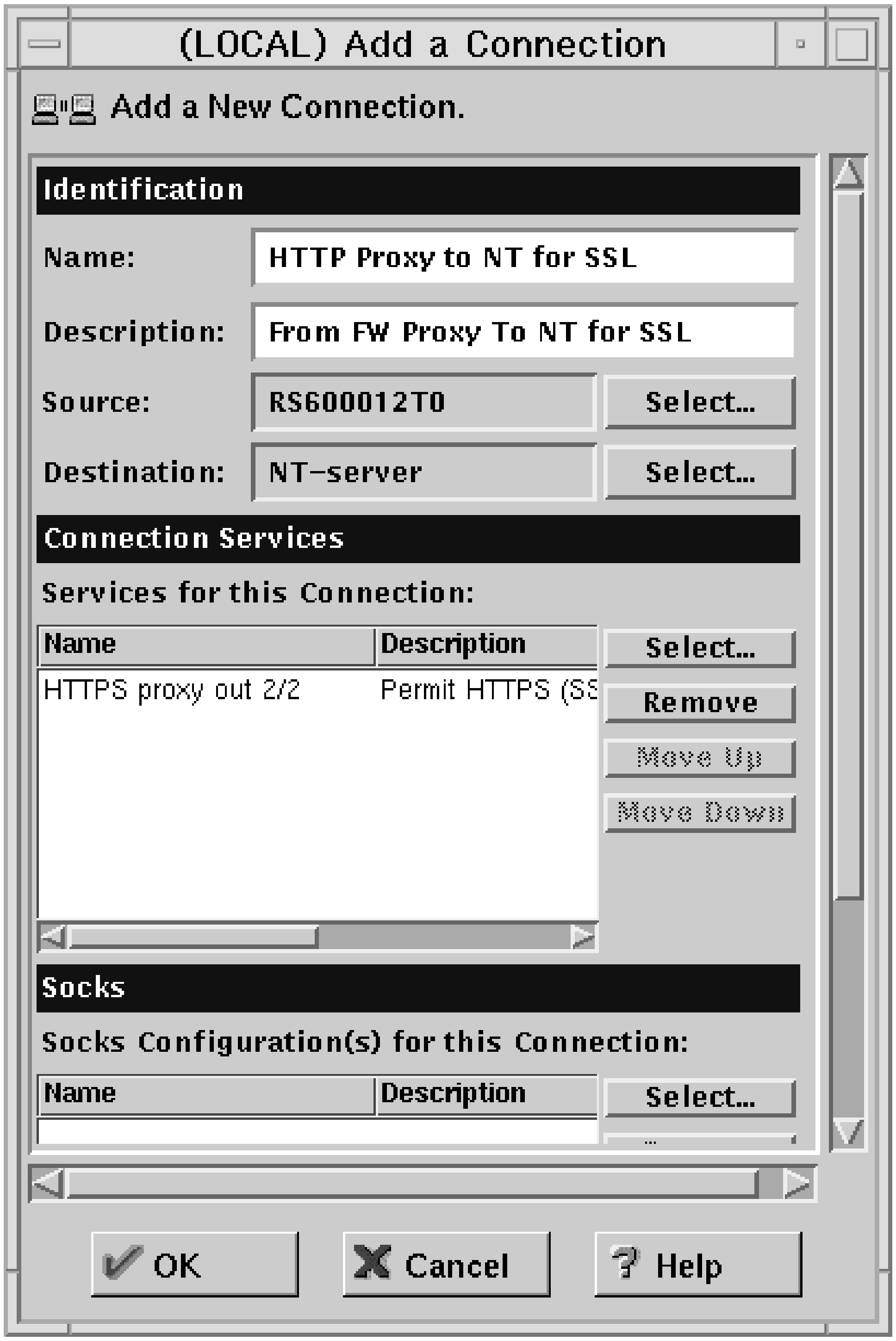
Time Control Action: Activate Service During Specified Times / Deactivate Service During Specified Times

OK Cancel Help

Figure 450. HHTPS Proxy out 2/2 Service Definition Window

Notice that the TCP port number involved in the SSL communication on the Web server machine is 443, which is the default port number for SSL communication that we accepted when we installed Lotus Domino Go Webserver (see Figure 284 on page 307).

So it was not difficult to configure IBM eNetwork Firewall 3.2 to implement the SSL tunneling technology. We only had to create the new connection named `HTTPS Proxy to NT for SSL` including the predefined service `HTTPS Proxy out 2/2`. The source object was the non-secure network adapter on the firewall and the destination object was the Web server machine. The permitted port on the firewall was 443. The following figure shows the Add a Connection window that we used to create the needed connection:



Name: HTTP Proxy to NT for SSL

Description: From FW Proxy To NT for SSL

Source: RS600012T0

Destination: NT-server

Services for this Connection:

Name	Description
HTTPS proxy out 2/2	Permit HTTPS (SS

Socks Configuration(s) for this Connection:

Name	Description

Figure 451. Connection Configuration for SSL Tunneling

Make sure that the three connections that you need to implement SSL tunneling are the only ones that appear in the Connections List window and then activate them by clicking **Activate...**. As usual, the Connection Activation window is brought up. You must check the box **Regenerate Connection Rules and Activate** and then click **Execute** (see Figure 428 on page 462).

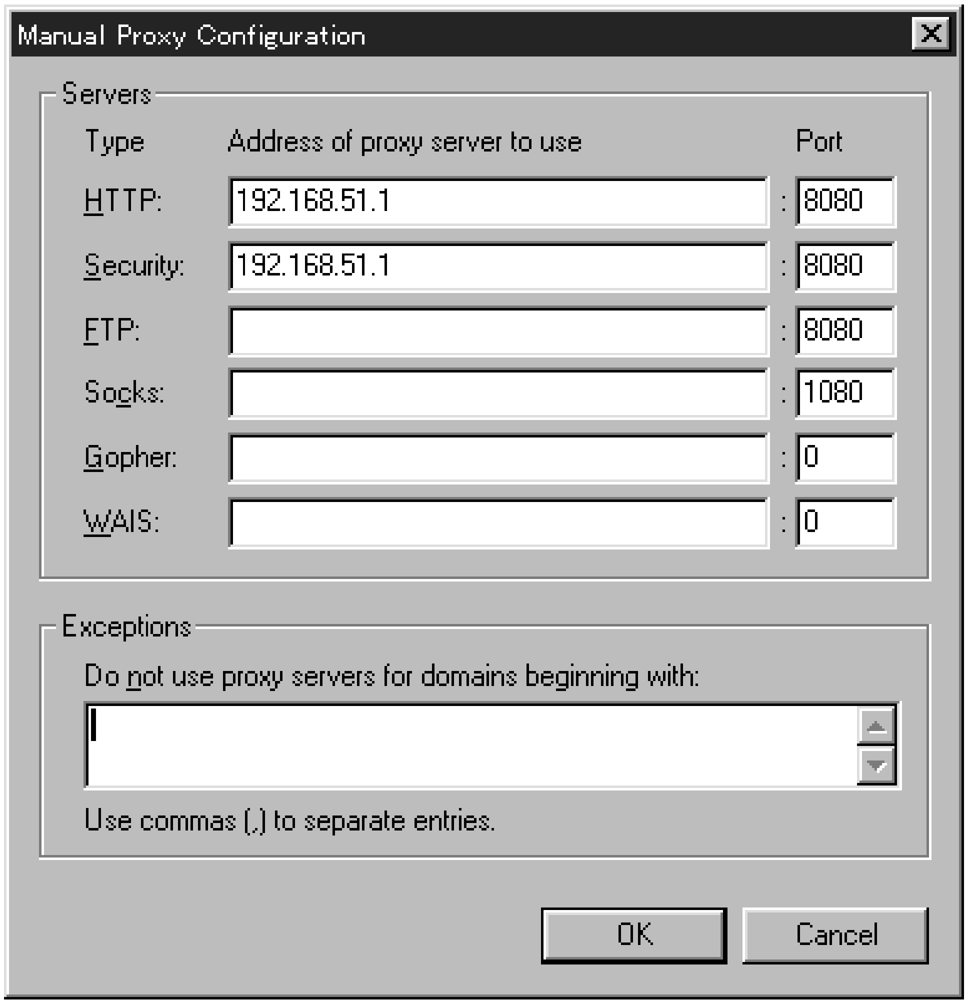
Before the firewall configuration can be considered complete, you should access the HTTP Proxy Configuration page as indicated in Figure 437 on page 472 and then launch the HTTP proxy daemon through the command

```
/usr/sbin/phttpd
```

if it is not already running.

10.1.2 Netscape Communicator Advanced Configuration for Using SSL Tunneling

Netscape Communicator must be configured for using SSL tunneling. You should open the Manual Proxy Configuration window as indicated in 9.7.3, “Netscape Communicator Advanced Configuration for Using HTTP Proxy Server” on page 476 and then fill the HTTP and Security fields with the IP address of the proxy server you want to use. In our case, we therefore typed the IP address of the secure network adapter of our firewall. The Port fields corresponding to HTTP and Security also have to be filled with the port number to which the HTTP Proxy server will listen. We typed `8080`, which is the default value that we had accepted when we had configured the HTTP proxy server on our firewall. The following figure summarizes all our selections:


Manual Proxy Configuration

Servers

Type	Address of proxy server to use	Port
HTTP:	192.168.51.1	8080
Security:	192.168.51.1	8080
FTP:		8080
Socks:		1080
Gopher:		0
WAIS:		0

Exceptions

Do not use proxy servers for domains beginning with:

Use commas (,) to separate entries.

Figure 452. Netscape Communicator Configuration for SSL Tunneling

10.1.3 Testing the HTTPS Stream through the SSL Tunnel

Our test was very simple. Since our purpose was to describe how to implement the SSL tunneling firewall technology, we did not test it using a complex application. It was enough to point the Web browser on the client machine to the URL of the secure Web server https://192.168.50.2 and what we got was the Configuration and Administration Forms home page for Lotus Domino Go Webserver.

Of course, the IP address of the Web server machine could have been replaced by the host name of the same computer, since DNS was in place in our network environment (see 9.5.2, “Domain Name Service in a Firewall Protected Network Environment” on page 445). It is important anyway that you invoke the home page by starting the URL with `https://`, since in this way you can establish a secure client/server communication using HTTPS.

The following figure shows the result we obtained:

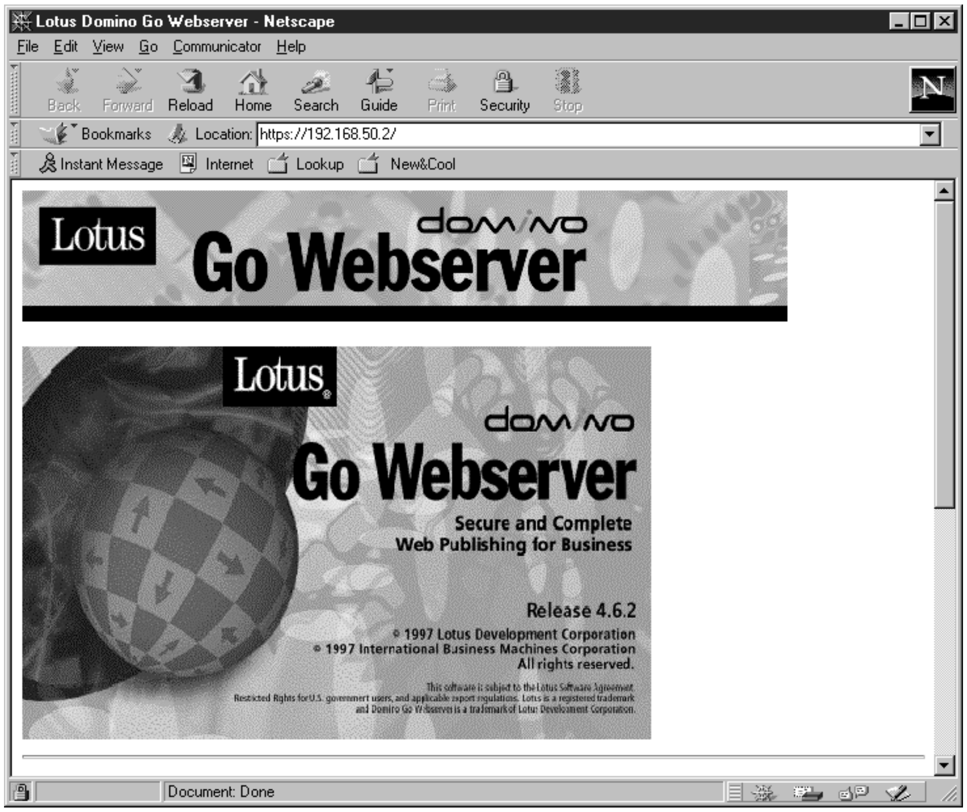

Lotus Domino Go Webserver - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: https://192.168.50.2/
Instant Message Internet Lookup New&Cool
Lotus
domino
Go Webserver
Secure and Complete
Web Publishing for Business
Release 4.6.2
© 1997 Lotus Development Corporation
© 1997 International Business Machines Corporation
All rights reserved.
Document: Done

Figure 453. SSL Communication

Notice that a closed lock appears in the bottom-left corner of the browser window. It means that a secure SSL session is running. That lock would appear open if the communication used the HTTP protocol.

The way we configured our firewall also permitted normal HTTP connection. It is possible to access an `https://` URL by linking it from a Web page accessed through an `http://` URL.

10.2 SOCKS Server for HTTPS Scenario

In this section we will show how the SOCKS server firewall technology can be used successfully to let HTTPS pass through the firewall. Of course, an HTTPS communication between the client and the server takes place only if the Web server is enabled to use SSL and if an SSL session has been activated (see 3.1.2, "Lotus Domino Go Webserver SSL Setup" on page 71, 3.1.3, "Lotus Domino Go Webserver SSL Server Authentication" on page 79 and 3.1.4, "SSL Client Authentication" on page 84).

10.2.1 How to Configure SOCKS Server for HTTPS

This section describes how we configured the firewall machine to pass HTTPS using the SOCKS server technology. We performed this test using IBM eNetwork Firewall 3.2 for AIX.

First of all, we needed to specify objects and connections in this scenario. We had already defined objects in 1 on page 488. It was possible to keep those same definitions for this scenario also. The following list shows the five connections that we needed for this scenario:

- Secure To SOCKS

  We created this new connection in order to pass the SOCKS protocol from the 95-client object to the RS600012T1 object.

- SOCKS To Nonsecure

  We created this new connection in order to pass the HTTP protocol from the RS600012T0 object to the NT-server object.

- SOCKS Config for HTTP

  We created this new connection in order to pass the SOCKSified HTTP protocol from the 95-client object to the NT-server object.

- HTTP Proxy to NT for SSL

  We created this new connection to permit the SSL protocol to pass from the RS600012T0 to the NT-server object.

- SOCKS Config for HTTPS

  We created this new connection to pass SOCKSified HTTPS from the 95-client object to the NT-server object through the firewall.

We have already shown how to define all these connections except the one named `SOCKS Config for HTTPS` (see 9.8.1, “How to Configure SOCKS Server for HTTP and IIOP over HTTP” on page 478 and 2 on page 488).

IBM eNetwork Firewall 3.2 does not provide a predefined HTTPS SOCKS rule that could be used for this purpose and we explain here how to create such a rule. Notice that implementing these connections will implicitly restore the SOCKS server technology for controlling the HTTP flow between the client and the server. In fact we want to continue granting the possibility for client and server to communicate through the HTTP protocol, since an `https://` page is often accessed as a link from an `http://` page.

To define the new HTTP SOCKS rule needed to implement the new connection, you should first select **Socks** from the Configuration Client navigation tree, as shown in the following figure:

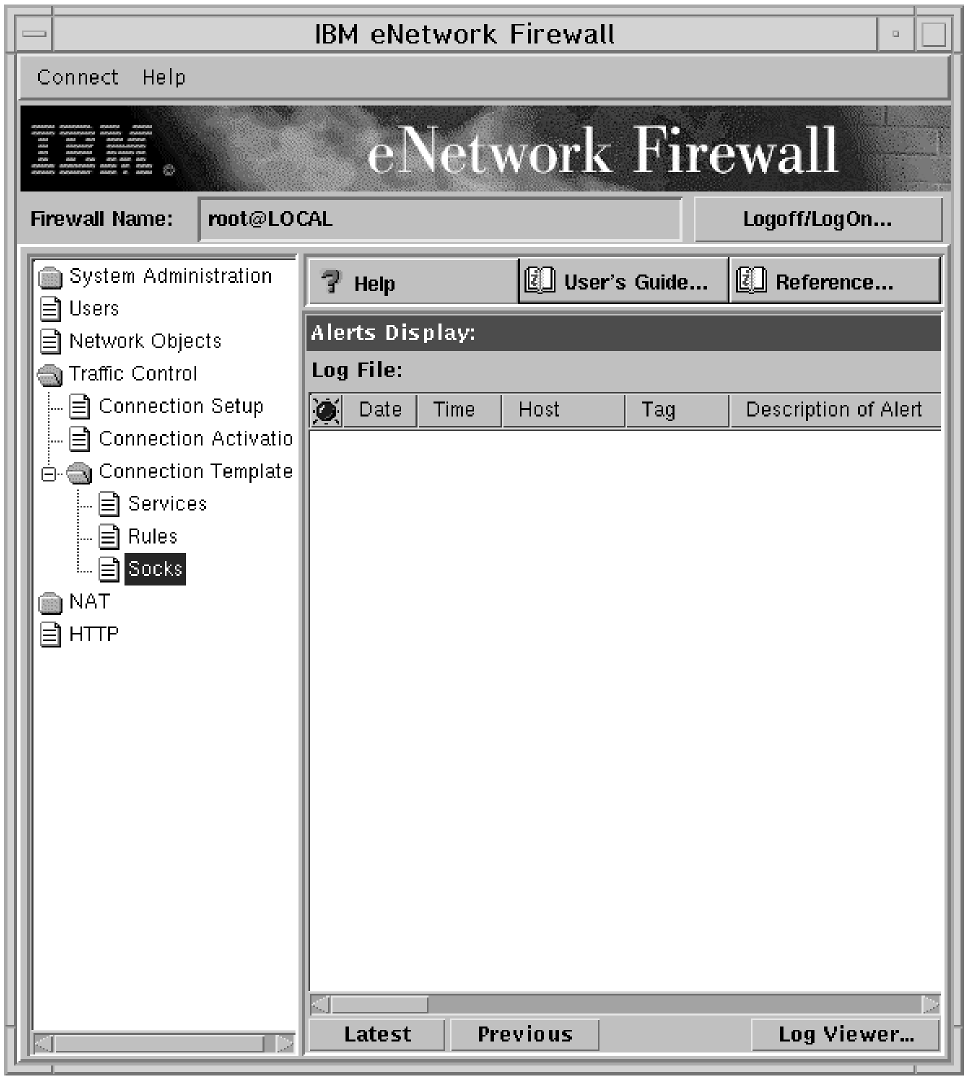
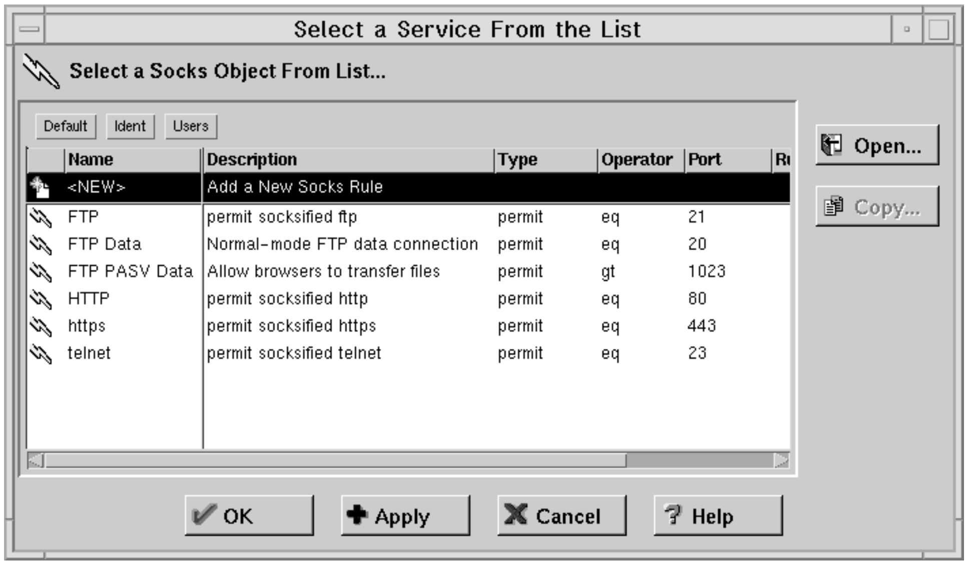


Figure 454. Socks in the Configuration Client Navigation Tree

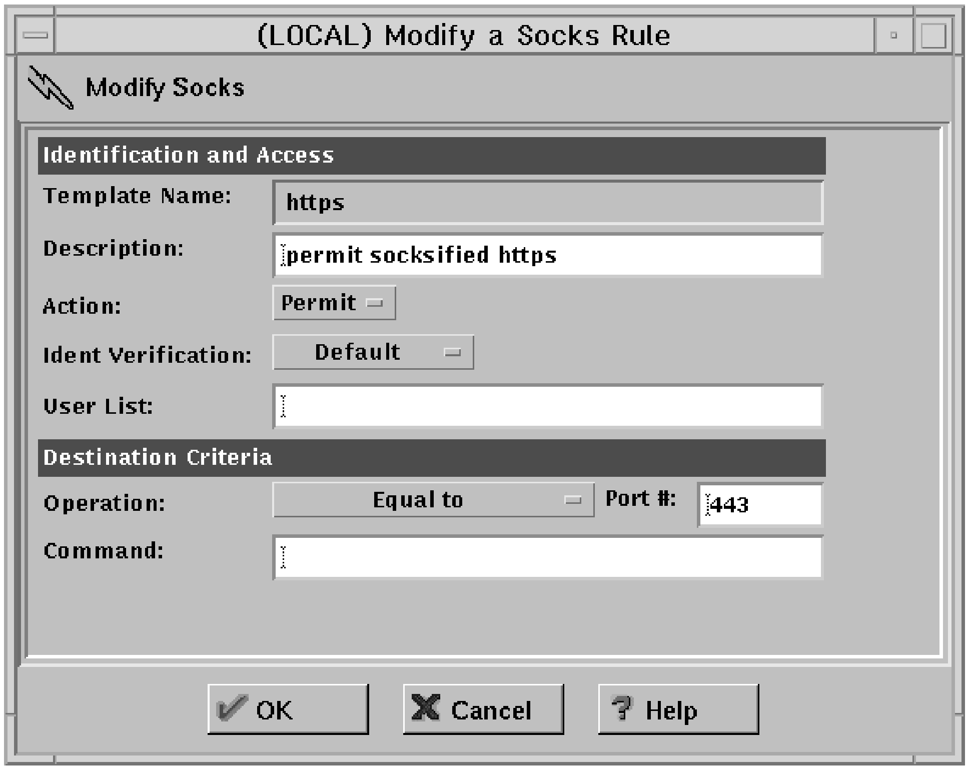
The Socks Objects list is displayed:


Select a Service From the List
Select a Socks Object From List...
Default
Ident
Users
Open...
Copy...
OK
Apply
Cancel
Help

Name	Description	Type	Operator	Port
<NEW>	Add a New Socks Rule			
FTP	permit socksified ftp	permit	eq	21
FTP Data	Normal-mode FTP data connection	permit	eq	20
FTP PASV Data	Allow browsers to transfer files	permit	gt	1023
HTTP	permit socksified http	permit	eq	80
https	permit socksified https	permit	eq	443
telnet	permit socksified telnet	permit	eq	23

Figure 455. Socks Objects List

You need to select **NEW** and click on **Open...** because you want to add a new SOCKS rule. As soon as we did that, a new window was brought up, so that we had the opportunity to fill its fields, as shown in the following figure:

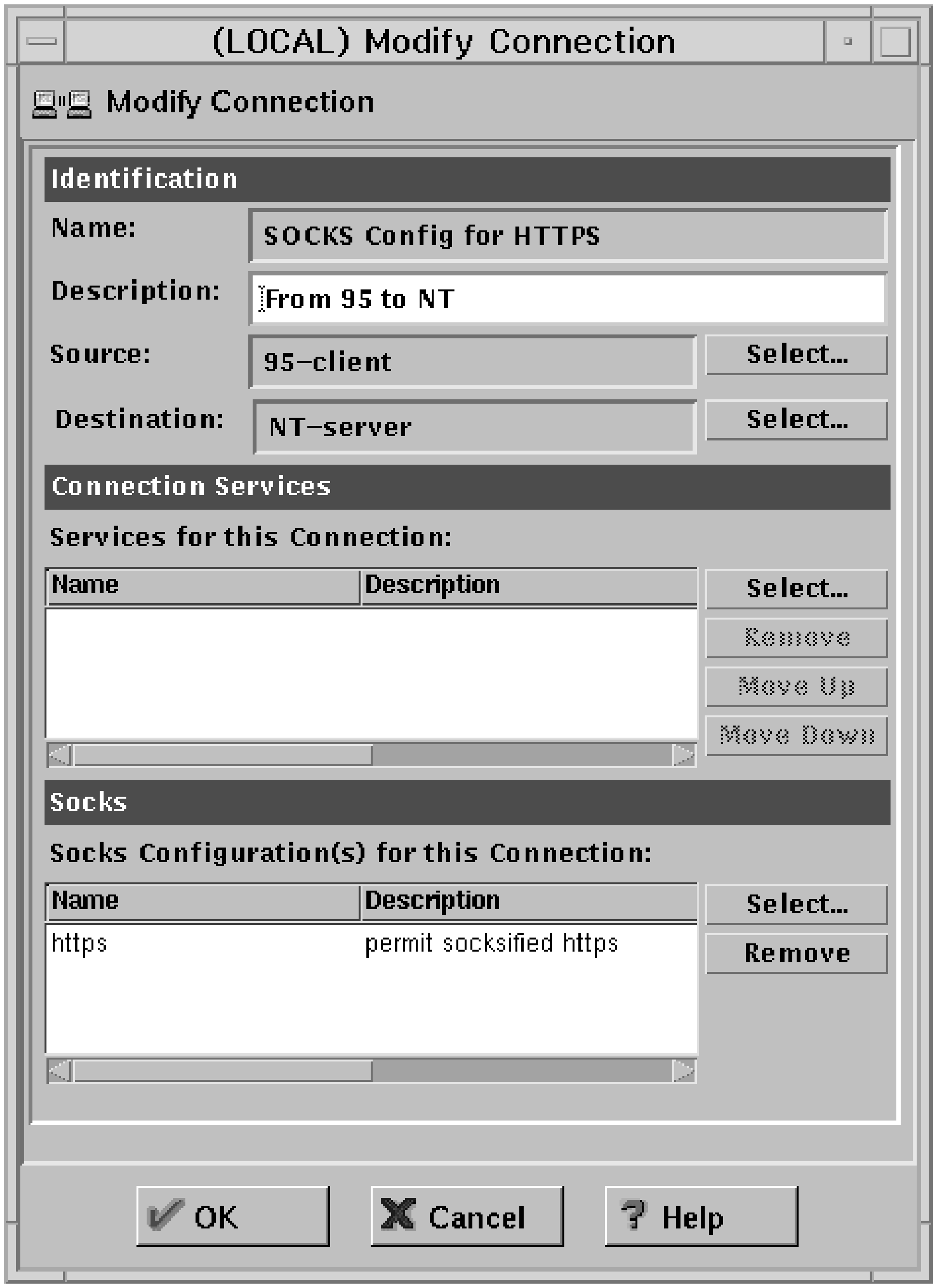


Template Name:	https
Description:	permit socksified https
Action:	Permit
Ident Verification:	Default
User List:	
Operation:	Equal to
Port #:	443
Command:	

Figure 456. SOCKS Rule for HTTPS

We named the new SOCKS rule `https` and we filled in the Description field with `permit socksified https`. Notice that the HTTPS protocol uses by default the TCP port 443 on the Web server. This was the value that we typed in the Port # field.

Once this new rule was created, we added the new connection named `SOCKS Config for HTTPS`. We selected **95-client** as source object and **NT-server** as destination object. Moreover, when we had to add a rule to this connection, we could select **https**, the new SOCKS rule that we just created. At the end, the connection window appeared as shown in the following figure:


(LOCAL) Modify Connection

Modify Connection

Identification

Name: SOCKS Config for HTTPS

Description: From 95 to NT

Source: 95-client

Destination: NT-server

Connection Services

Services for this Connection:

Name	Description

Socks

Socks Configuration(s) for this Connection:

Name	Description
https	permit socksified https

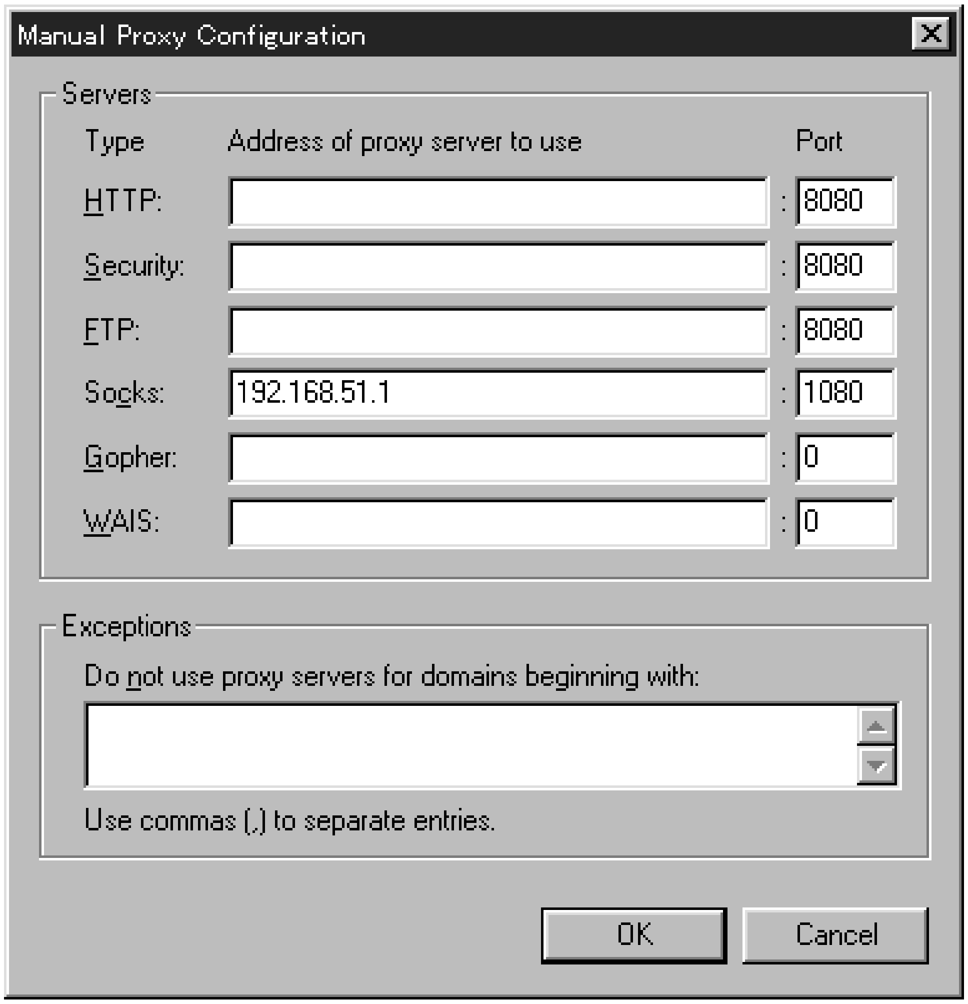
Figure 457. SOCKS Config for HTTPS Connection Definition

The firewall configuration can be considered finished as soon as you activate the five connections. You must be sure that these five connections are the only ones that appear in the Connections List window. If necessary you have to remove other connections. You can activate the five connections as usual through the Connection Activation window (see Figure 428 on page 462).

10.2.2 Netscape Communicator Advanced Configuration for Using SOCKS Server for HTTPS

Netscape Communicator must be configured to use the SOCKS server on the firewall machine. Its configuration in this case is independent of whether you are using HTTPS or HTTP, so the steps that we are going to describe here are very similar to the HTTP case (see 9.8.3, “Netscape Communicator Advanced Configuration for Using SOCKS Server for HTTP” on page 483).

You should open the Manual Proxy Configuration window as indicated in 9.7.3, “Netscape Communicator Advanced Configuration for Using HTTP Proxy Server” on page 476 and then fill the Socks field by typing either the IP address or the host name of the firewall machine. We typed `191.168.51.1`, which was the IP address assigned to the secure network adapter on the firewall machine. It is also necessary to specify the exact port used by the SOCKS server, which is by default 1080. We typed this value in the Port field related to Socks, then we pressed **OK**, as shown in the following screen:



Type	Address of proxy server to use	Port
HTTP:		8080
Security:		8080
FTP:		8080
Socks:	192.168.51.1	1080
Gopher:		0
WAIS:		0

Do not use proxy servers for domains beginning with:

Use commas (,) to separate entries.

Figure 458. Netscape Advanced Configuration for Using SOCKS Server

10.2.3 Testing the HTTPS Stream through the SOCKS Server

The experiment that we tried here was the same as we tried in 10.1.3, “Testing the HTTPS Stream through the SSL Tunnel” on page 493, and we simply invoked the Configuration and Administration Forms home page for Lotus Domino Go Webserver, installed on the Web server machine. It was enough to point the browser on the client machine to the URL https://192.168.50.2 or https://wtr05218, since DNS was in place in our network environment (see 9.5.2, “Domain Name Service in a Firewall Protected Network Environment” on page 445). Our test was successful and we obtained a result similar to the window shown in Figure 453 on page 494. The Netscape security icon - a closed lock - appeared in the bottom-left corner of the browser window. This demonstrated that we were able to have HTTPS pass through the firewall using the SOCKS server firewall technology.

It is very important to notice that in this case, as also in the SSL tunneling test, the URL must start with `https://`, so that a secure SSL session is activated.

Moreover the firewall had been configured to permit direct HTTP communication between the client and the server. Such configuration allows a user to access an `https://` URL through a Web page previously accessed via an `http://` URL, rather than pointing directly to an `https://` URL.

Appendix A. Special Notices

This publication is intended to help customers, system engineers and application developers to implement a secure Network Computing Framework environment. The information in this publication is not intended as the specification of any programming interfaces that are provided by Lotus Domino Go Webserver, IBM WebSphere Application Server, IBM DB2 Universal Database and IBM eNetwork Firewall. See the PUBLICATIONS section of the IBM Programming Announcement for Lotus Domino Go Webserver, IBM WebSphere Application Server, IBM DB2 Universal Database and IBM eNetwork Firewall 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.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	DB2
DB2 Universal Database	Distributed Relational Database Architecture
eNetwork	IBM
Net.Data	OS/2
OS/390	RS/6000
ThinkPad	

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Appendix B. Related Publications

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

B.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see "How to Get ITSO Redbooks" on page 505.

- *Network Computing Framework Component Guide*, SG24-2119
- *Protect and Survive Using IBM Firewall 3.1 for AIX*, SG24-2577
- *The Domino Defense: Security in Lotus Notes and Internet*, SG24-2109
- *A Comprehensive Guide to Virtual Private Networks, Vol. I*, SG24-5201

B.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
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

B.3 Other Publications

These publications are also relevant as further information sources:

- *IBM eNetwork Firewall for AIX User's Guide*, GC31-8419
- *IBM DB2 Universal Database Administration Guide Version 5*, S10J-8157

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 `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:

- **Redbooks Web Site on the World Wide Web**

  `http://w3.itso.ibm.com/`

- **PUBORDER** — to order hardcopies in the United States
- **Tools Disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLCAT REDPRINT
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type the following command:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

- **REDBOOKS Category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

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** — 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)001-408-256-5422 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com/
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

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.

Index

ITSO Redbook Evaluation

Internet Security in the Network Computing Framework
SG24-5220-00

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.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
__Customer __Business Partner __Solution Developer __IBM employee
__None of the above

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!)**

____

____

____

____

____

Internet Security in the Network Computing Framework **SG24-5220-00**

SG24-5220-00
Printed in the U.S.A.