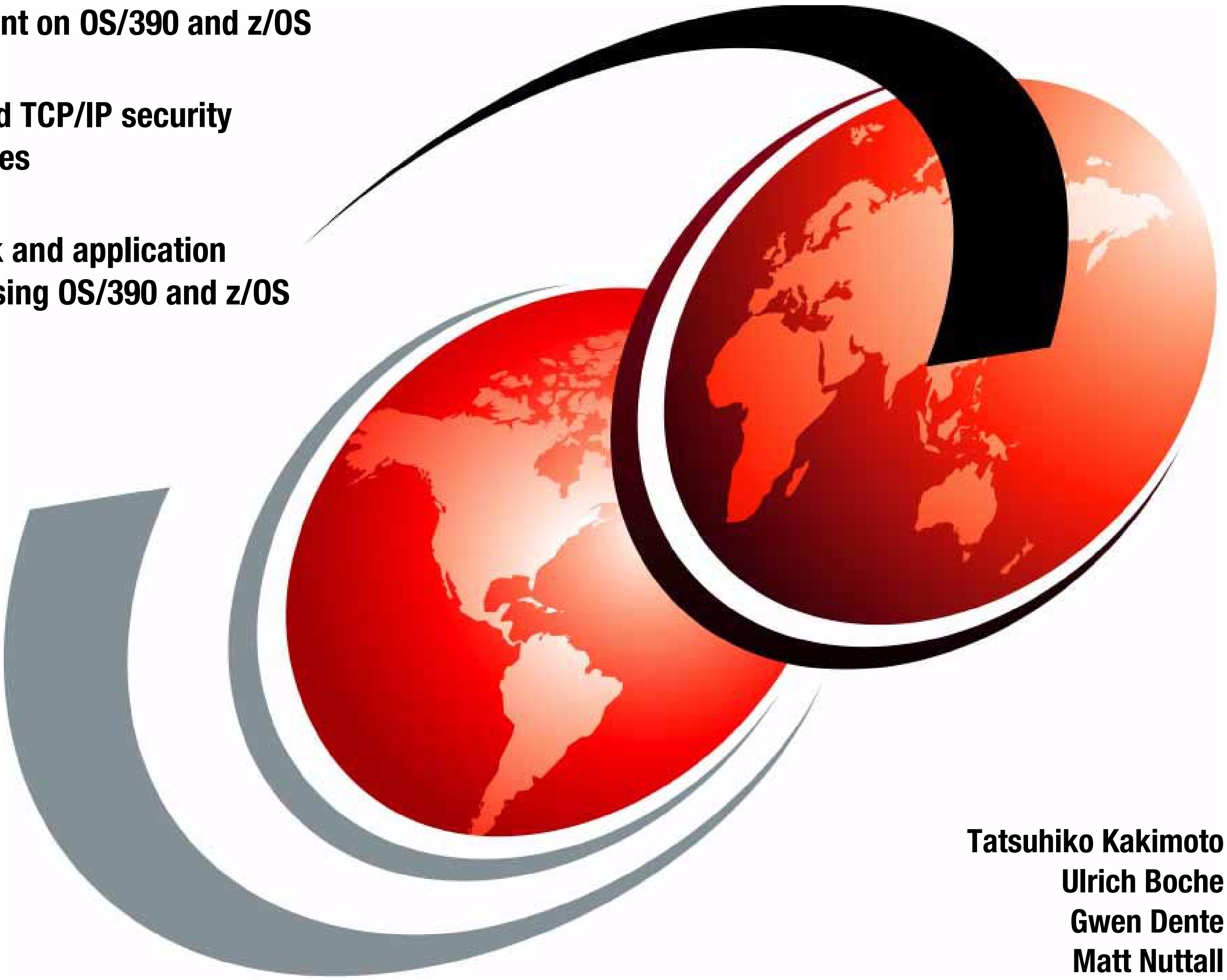
Secure e-business in TCP/IP Networks on OS/390 and z/OS

Guide to the secure e-business environment on OS/390 and z/OS

Understand TCP/IP security technologies

IP network and application security using OS/390 and z/OS



Tatsuhiko Kakimoto
Ulrich Boche
Gwen Dente
Matt Nuttall

Redbooks

ibm.com/redbooks

International Technical Support Organization

SG24-5383-01

Secure e-business in TCP/IP Networks on OS/390 and z/OS

June 2001

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix G, “Special notices” on page 451.

Second Edition (June 2001)

This edition applies to Communications Server for OS/390 V2R10 and all releases of z/OS Communications Server.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195

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

Preface

The Internet has changed the way of doing business all over the world. It enables consumers to perform many activities, such as banking, investment, shopping, and researching, from virtually anywhere they can get online. As the volume of the business activities over the Internet increases by leaps and bounds, companies that provide services require a more powerful, secure, and reliable platform, because they are expected to serve customers all over the world 24 hours a day, seven days a week.

However, as the volume of business grows, so does the risk. There are some people with malicious intentions trying to prevent those services from being performed and to impose heavy losses on businesses. OS/390 (and now z/OS) offers functionality, especially for networking and security, that has made it the ideal e-business platform with huge computing capacity, reliability, extensibility, and safety.

This redbook shows you how to secure your network access and TCP/IP applications on an OS/390 or z/OS platform, with practical examples of various configurations and implementations. The focus of this redbook, however, is not on detailed implementation information, but on helping you to establish a secure e-business environment using OS/390 or z/OS.

Further information about implementing particular applications can be found in the following redbooks:

- *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 1: Configuration and Routing*, SG24-5227
- *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228
- *IBM Communications Server for OS/390 TCP/IP 2000 Update Technical Presentation Guide*, SG24-6162

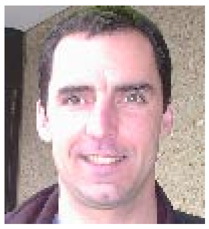
The team that wrote this redbook

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

Tatsuhiko Kakimoto is an Advisory Networking Specialist at the International Technical Support Organization, Raleigh Center. He writes extensively and teaches IBM classes worldwide on all areas of networking, especially CS for OS/390 IP Services. Before joining the ITSO in 1999, he worked in the technical support department in IBM Japan designing and implementing IP networks, and consulting for major IBM customers.

Ulrich Boche joined IBM Germany in 1970 and worked as a systems programmer with IBM mainframe systems in the IBM Germany computing centers for more than 10 years. After a two-year assignment in the MVS programming laboratory in Poughkeepsie, NY, he joined the Field Systems Center in Stuttgart, Germany in 1984, supporting the IBM SEs and MVS customers in Germany in the area of host systems security (RACF and cryptography) and later client/server security (for example DCE and Kerberos) and e-business and e-commerce security. Since 1997, he has been an IT security technical consultant in S/390 Technical Support. During numerous ITSO residency projects, he co-authored a large number of redbooks on security topics.

Gwen Dente is a Certified Consulting IT Specialist at the Washington Systems Center in Gaithersburg, Maryland. She has worked at IBM for 20 years, supporting cross-platform interoperability, with a focus on mainframe SNA and TCP/IP communications. She has written and taught extensively on Communications Server for OS/390 and is a frequent presenter at technical and user group conferences in the United States.



Matt Nuttall has worked in IBM global services for 12 years, specializing on the S/390 platform in several product areas. He has written several redbooks, including *TCP/IP in a Sysplex*, SG24-5235-00.

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

Robert Haimowitz
Gail Christensen
Linda Robinson
Margaret Ticknor
Jeanne S Tucker
International Technical Support Organization, Raleigh Center

Linwood Overby
Alfred Bundgaard Christensen
Jay Aiken
Lap Huynh
IBM z/OS Communications Server Strategy and Design, RTP

David Bruton III
Jon Franks
Mike Fox
Jake Griffin
Patrick Livecchi
Alan Packett
Jeff Trawick
David Yang
IBM z/OS Communications Server Development, RTP

David Wierbowski
IBM Firewall Technologies Development, Endicott

Paul de Graaff
IBM ITSS, New York

The first edition of this redbook
The advisor for this project was:

Jerzy Buczak
International Technical Support Organization, Raleigh Center

The authors of the second edition were:

Jorge Rivera of IBM Peru
Michyasu Takada of IBM Japan
Otto Zech of IBM Austria

Contributors to this project:

Linwood Overby
Alfred Bundgaard Christensen
CS for OS/390 Design, Research Triangle Park

Gus Kassimis
CS for OS/390 Development, Research Triangle Park

Karl Wozabal
IBM Austria

Paul de Graaff
International Technical Support Organization, Poughkeepsie Center

Martin Murhammer
Juan Rodriguez
Gail Christensen
Tatsuhiko Kakimoto
International Technical Support Organization, Raleigh Center

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 "IBM Redbooks review" on page 473 to the fax number shown on the form.
- Use the online evaluation form found at **ibm.com**/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Philosophy

The word *security* means many things to many people. In this chapter, we discuss what it means to an OS/390 IT manager, how it has been traditionally implemented in an SNA network, and the issues facing the same IT manager when confronted with the need for building a TCP/IP network.

Questions that the wise IT manager asks include:

- How can I be sure that confidential data is not being read by outsiders as it moves through my network?
- How can I be sure that nobody has modified the data traveling through the network?
- How can I be sure that the communication partner on the other end of the connection is exactly what it claims to be?
- How do I prevent malefactors from exhausting my network resources by flooding it with spurious information?
- How can I be sure that a user with valid access to one application cannot access another, more restricted, one?

All these questions are valid and reasonable whether your network is SNA or TCP/IP. However, because the design objectives and the history of the two architectures have been so different, these questions take on different levels of importance in each.

1.1 What we take for granted in an SNA network

Traditional SNA subarea networks were tightly controlled and managed. Every remote terminal in the network had to be predefined, and if the definition was absent then no communication took place. Also, putting together a workstation with a "bent" SNA protocol stack has always been more difficult than doing the same for TCP/IP. For these reasons, SNA security aspects tended to focus on those areas of concern that remain when a network is deemed trustworthy:

1. Protection against the really serious crook, which can only be assured by means of encryption
2. Message authentication
3. Verification of partners accessing your network from another company's SNA network
4. Denial-of-service attacks
5. Application access controls

The advent of APPN with its extra dynamics changed this somewhat, and the focus changed toward protection of the internal network (the SNA "intranet"). Today, SNA networks are generally regarded as very secure provided you implement the recommended definitions, user exits and encryption methods in the appropriate places. However, SNA has never had a true Internet, so that some of the weapons-grade techniques used against Internet villains have never come into vogue in SNA networks.

In this section we summarize how SNA caters to the security issues outlined above.

1.1.1 Confidential information protection

If the information carried on a particular SNA session is so sensitive that it must be protected from being read by anyone other than the session partners, then it must be encrypted. SNA provides for session-level encryption on all LU session types, including LU 6.2. The need to encrypt session data is negotiated between the partners when the session is started.

SNA uses the DES algorithm for all its encryption requirements. triple-DES support was added in CS for OS/390 Version 2 Release 8.

1.1.2 Message authentication

Information carried on a session may need to be protected from *modification*, but it may not be so sensitive that it requires full encryption. What is required here is the ability to verify that a received message has not been tampered with since it was sent by the session partner. To achieve this, you need to create a hash total of the message contents; to achieve it reliably, you need to encrypt the hash total as well. As with full data encryption, the requirement for message authentication is negotiated at session setup time. The message authentication levels available are cyclic redundancy check, no encryption (CRC), DES and triple-DES.

In an SNA network, message authentication also serves to authenticate the session partner. Both partners must have the same encryption key for the session to succeed.

1.1.3 Partner authentication

All the encryption in the world is quite useless if someone can maliciously impersonate your session partner. Therefore, a means must be available for you to verify the identity of that partner before transmitting any data.

Because of subarea SNA's rigidly controlled hierarchical structure, there was no specific partner authentication technique in the original architecture. If the need for security was sufficiently great, encryption would do the job. Because SNA uses symmetric key encryption, no impersonation of a partner could succeed unless the partner's DES key was also known to the villain.

A major exposure relating to partner authentication *without* encryption occurred when a session request was received from a business partner's network that was not considered completely trustworthy. The technique used in such a case was to check the incoming session request at the network (SNI) boundary for the origin and target LU names, using the session management exit. Any suspicion and the session request could be discarded. While this technique is not perfect, it can at least ensure that no impostor can impersonate a valid LU name in *your* network.

Within a single enterprise's network, another major exposure for unencrypted communication is that caused by a workstation on a switched connection impersonating another's node ID. Without some form of encryption, the only ways available to check the workstation's identity are dial-back or verification of the MAC (or ATM, or other DLC-level) address.

When SNA developed into APPN, new techniques were introduced to authenticate partners without the need to encrypt any session data. These techniques, all used with LU 6.2 communication alone, are still based on symmetric key encryption. One partner generates a random number, encrypts it, and sends it to the other partner. The second partner decrypts it and returns it. If the original matches the returned value then the partner must have the same key and is therefore who he claims to be. Thereafter, there is no need for further encryption. There are three variants of this technique:

- LU 6.2 session security. The partner LUs exchange random numbers to verify each other.
- LU 6.2 conversation security. The partner *transaction programs* exchange random numbers to verify each other.
- CP-CP session security. The partner *control points* exchange random numbers to verify each other. This last variant is particularly useful when two CS for OS/390 systems are linked using APPN, especially over a switched transport network. If you can be sure that your partner VTAM is authentic you can be reasonably sure that session requests coming from it are also authentic, and you need not go to the extra expense of verifying every such request.

1.1.4 Denial-of-service attacks

As indicated above, denial-of-service attacks are not considered a big issue in SNA networks. The attacker would have to bombard the target with large numbers of spurious session requests. Typically this would involve more work, and use more resources on the malicious hacker’s machine, than simply sending in bent SYN packets in a TCP/IP network. Moreover, without the anonymity of the Internet the hacker would stand a greater chance of being identified.

1.1.5 Application access control

In an SNA network, the applications are not considered to be part of the network architecture, and therefore securing them is the preserve of the operating system rather than the network software such as CS for OS/390. There are no generally recognized equivalents of FTP, Telnet and (especially) HTTP. On OS/390, an application protects itself by demanding user IDs and passwords to match those defined to the security subsystem (such as RACF) running on the operating system.

1.2 How TCP/IP differs

In terms of security issues, the major difference between SNA and TCP/IP is the presence of the Internet. This, as we all know, is populated by malevolent hackers whose hobby it is to prove that they can beat any security system devised by man. Therefore, security in TCP/IP networks places a much higher importance on partner authentication and on combating denial-of-service attacks than does SNA security. Here we summarize how TCP/IP addresses the five main areas of security exposure.

1.2.1 Confidential information protection

As with SNA, hiding sensitive information in a TCP/IP network is based on encryption. Unlike SNA, there are several security architectures available in a TCP/IP network for this purpose. Two in particular are in vogue at present:

- IP Security (IPSec) provides for encryption of all communication between IP hosts, and is particularly suited to the situation where two mutually trusted partners communicate across an untrusted network such as the Internet. IPSec enables the setup of a secure logical connection called a *virtual private network* (VPN) between partners.
- Secure Sockets Layer (SSL) encrypts communications between a client and a server at the application level. This is more suited to such applications as Web access and TN3270, where (typically) a single client on a workstation needs access to a single application on a server.

Both IPSec and SSL provide for a range of encryption algorithms to be used. Normally, symmetric key algorithms such as DES, triple-DES, RC2 and RC4 are used to protect IP datagrams once the partners have authenticated each other.

1.2.2 Message authentication

This, too, is handled in TCP/IP in a similar fashion to SNA. Both IPSec and SSL provide for message authentication using one of a number of algorithms available. The degree of message authentication (and full encryption) is determined at initial contact time, when the partners exchange handshakes.

1.2.3 Partner authentication

In a TCP/IP network, it is unwise to rely on partner IP addresses or subnetwork addresses for authentication purposes. All serious authentication is based on encryption, and the two major protocols mentioned above (SSL and IPSec) include defined methods for doing this.

The simplest method for authentication is to rely on both partners possessing the same (symmetric) key; if your partner responds sensibly to your encrypted message, then it must have the right key and is therefore authentic. However, the maintenance and distribution of such keys is a painful process in a large network, as well as being an additional security exposure. For this reason, TCP/IP makes extensive use of public/private key algorithms for authentication purposes. Such algorithms make it unnecessary to distribute secret keys manually across a network. The usual method of implementing IPSec or SSL is to use a public/private key algorithm (such as RSA) to authenticate the partner and to exchange a new symmetric (DES or other) key; the new symmetric key is then used to protect the datagrams once communication has started. Symmetric key algorithms are more efficient than public/private keys, so it is better to use such an algorithm once you are assured that both partners have the same key.

The public/private key algorithms make use of a pair of keys, such that each can decrypt a message encrypted with the other, but cannot decrypt a message encrypted with itself. Therefore, the technique works as follows:

1. You send your partner your public key, in the clear. In fact what you send is a *certificate*, which contains your public key as well as some information about you.
2. Your partner uses it to encrypt a message that it sends to you.

3. You decrypt it using your private key, which you keep secret.
4. You return the message to your partner.
5. The fact that you were able to decrypt the message proves that you have the private key, and therefore are who you say you are.

In reality it is a little more complex, because a villain could pretend to be your partner and send you a false public key that the villain could decrypt. For this reason, certificates are usually created by trusted certification authorities; such an authority signs each certificate with its own private key, which can therefore be checked by decrypting it with its well-known public key.

1.2.4 Denial-of-service attacks

Protection of a network against messages designed to tie up network resources is usually accomplished by means of a *firewall*. A firewall is a node located between a trusted network and the Internet (or any other unsecure network where ill-intentioned malefactors may lurk). A firewall does its job by inspecting arriving packets and discarding any that do not conform to strictly defined criteria for protocol usage, IP addresses, port numbers and so on. In the worst case, even if the firewall is brought to its knees by an attack the internal network can continue its work in comfort and safety.

1.2.5 Application access control

As with SNA, access to applications from the TCP/IP network can be secured using user IDs and passwords in combination with a RACF-managed security policy. However, TCP/IP itself includes several applications that are regarded as part of the architecture, and therefore require their own entries in RACF. This would be a straightforward matter for installations used to SNA security, except that on OS/390 many of these applications (for example, FTP, UNIX Telnet, and Web server) run under UNIX System Services (OS/390 UNIX or z/OS UNIX) instead of an MVS environment. Therefore, knowledge of the way OS/390 UNIX functions are defined to RACF is essential if your security is to cover all aspects.

Some TCP/IP applications (such as TFTP and anonymous FTP) require special consideration because they do not recognize individual user IDs. They must be configured carefully to avoid security exposures.

On the other hand, some TCP/IP applications provide additional access control to what is available using just RACF user IDs. Most notable among these are:

- Applications that are able to authenticate their clients using SSL/TLS, including the TN3270 server, the FTP server/client, and the HTTP server.
- Kerberized UNIX applications, which are new in z/OS V1R2, including the UNIX Telnet server, the UNIX rsh server, the FTP server/client, and the LDAP server.

Using the TCP/IP stack access functions, including network access control, stack access control, and port access control, access to applications can be policed based on where the client resides, who they are, and to which server they are wishing to connect.

1.3 TCP/IP security issues

The previous discussion concentrates on *network* security and network *application* security, in other words, the areas where the TCP/IP architecture differs significantly from its SNA brother. One area often overlooked is that of *platform* security: how well does the operating system itself protect data from users and users from each other. MVS has always been viewed as a paragon of security in this aspect, but now we also have UNIX System Services closely integrated with it. The techniques available in UNIX, and perhaps more importantly the practices used in UNIX installations, are very different from those familiar to MVS administrators. Chief among these is the habit of allowing superuser access to all sorts of operations and administrative staff. Superuser authority confers limitless powers in the traditional UNIX and HFS environment; to make OS/390 UNIX security compatible with MVS security the number of users with true superuser authority must be severely limited.

Next, we offer some hints on securing a TCP/IP environment in three main areas: the operating environment, the intranet and the Internet.

1.3.1 The operating environment

The trick here is to make sure that UNIX System Services and HFS are controlled as strictly as traditional MVS and its data sets:

- Limit severely the number of superusers.
- Limit the powers that even superusers have over the system. OS/390 UNIX permits you to define varying levels of superuser authority, by means of the BPX.DAEMON and BPX.SERVER facility classes.
- Restrict the access that default OS/390 UNIX users have to the system. Default users are those that need TCP/IP services (and therefore OS/390 UNIX) but do not need to use OS/390 UNIX directly. It is a good idea, in the case of default users, to:
  - Make their home directory something to which they have no access, or some unimportant directory such as /temp.
  - Make their default program (which runs when they log into OS/390 UNIX) something safe such as /bin/echo.
- Use the HFS *sticky bit* or the *extended attributes* to control program loading. Because a user can change the search order for *executable* HFS files, the whole HFS is considered untrustworthy in terms of program authorization. The sticky bit, formerly used to keep frequently loaded HFS program files in storage (rather like LPA), is now used to cause MVS to bypass the HFS search and to load the requested program from an MVS library in the usual way. Thus the usual MVS program authorization takes place. The extended attributes can be set for individual HFS files, by authorized users, to define those program files as authorized.
- Protect the HFS files in the same way as the MVS data sets. RACF can be used to associate MVS users and jobs with UNIX process IDs, which are then used to validate access to HFS files and directories.
- Control the use of the syslogd server. The syslogd files are accessible to all users via the syslogd daemon, and are prime candidates for denial-of-service attacks (whether intentional or accidental) as they fill up. Consider limiting

their size, giving them a separate HFS, and restricting access from the network (via UDP port 514).

CS for OS/390 V2R10 IP introduced security enhancements called syslogd isolation, which allows syslogd to isolate messages into separate logs with more granularity than was previously possible and to restrict communication with remote syslog servers more easily. See 5.12, “OS/390 and z/OS syslog daemon security” on page 232 for more information.

1.3.2 The intranet

Within your own internal TCP/IP network, the users are probably considered reasonably trustworthy, reasonably accountable, and reasonably amenable if only because they can be fired if caught misusing the system. Points to watch out for include:

- Run only those TCP/IP services and applications that you need.
- If you have to run TCP/IP applications that allow anonymous access to your system, be extremely careful in configuring them. Classic examples are anonymous FTP, TFTP and RSH.
- Where TCP/IP applications support the use of user IDs and passwords, apply the same standards in RACF authorization as you would to existing applications.
- Use encryption technology, such as SSL/TLS, IPSec, and Kerberos, for extremely sensitive data.
- Implement new sockets applications with SSL/TLS security built in.

1.3.3 The Internet

If your TCP/IP network is connected to the Internet (as is most likely), then "it’s a jungle out there" as they say. Clients that attempt to access your system cannot be assumed to be in any way trustworthy, benign, or even sane. You are strongly recommended to take the following measures:

- Use a firewall to bar access to all but the most subtle criminals.
- Use IPSec for all communication to external networks, whether those of business partners or pockets of your own enterprise.
- Use SSL for TN3270 access from business partners or your own employees. For FTP access, TLS/SSL should be used as well.
- For Web access from the jungle itself (the general public), place the Web server in its own protected subnetwork (a *demilitarized zone*, or DMZ) with firewalls on both the Internet and the intranet sides. Use SSL additionally where possible. See Chapter 7, “Security scenarios simplified” on page 301, for description on how to build a Web server on OS/390 and z/OS in a DMZ environment.

1.4 End-to-end security

The potential threats to your business can be resolved by paying attention to several distinct areas of security:

1. Ensure that data flowing in the network is protected. Encryption is the normal technique here.

CS for OS/390 has many network security-related functions built in, and these functions are being enhanced with every succeeding release. See 1.5, “OS/390 (z/OS) TCP/IP security options overview” on page 9 for the overview of its enhancements.

Chapter 2, “TCP/IP security overview” on page 11 discusses the technologies available for implementing *network* security with TCP/IP.

2. Ensure that access to network resources is controlled, such as TCP/IP stacks in an OS/390 (or z/OS) system. Also, access from/to a particular network might have to be prohibited for particular users.

   Chapter 4, “TCP/IP stack security” on page 91 discusses how access to network resources from/to an OS/390 system can be controlled by a network administrator.

3. Ensure that access to your operating environment (MVS and especially UNIX System Services) is controlled. The UNIX environment configuration and the protection of resources using RACF are vital areas to address.

   Chapter 3, “UNIX System Services security” on page 55 describes the measures available to control *platform* security in an OS/390 TCP/IP system.

4. Ensure that individual applications are protected, each according to its requirements. Again, careful customization and judicious use of RACF are essential.

   Chapter 5, “TCP/IP application security” on page 101 details the *application* security options available to protect an OS/390 TCP/IP environment.

It is relatively easy to configure OS/390 to provide a very secure mainframe environment. However, all the security products and all the definition work in the world are in vain if the other end of the communication is compromised. Therefore, your security policy must take into account your chosen communication partners, and the path to be taken by such communication, as well as the host setup.

We recommend that:

- Wherever you communicate *across* the Internet to a partner network, you should build a firewall on each side and use IPSec (in the form of a virtual private network, an encrypted tunnel) for all communications across the Internet. Each firewall should be carefully tailored to restrict the IP addresses, ports and packet types that it forwards.
- Where your partner network is at all dubious, use IPSec end-to-end all the way to the client workstation.
- Wherever you communicate *to* a Web client on the Internet, your Web server should be in a demilitarized zone to maximize the difficulty of reaching your host from the jungle. Other publicly accessible hosts, such as anonymous FTP servers, should also be in this zone.
- Wherever sensitive data is to be transmitted, use SSL for TN3270 or Web (HTTP) traffic, or session-level encryption for SNA traffic. If that means double encryption across the Internet, then so much the better.

Chapter 6, “SecureWay Security Server Firewall Technologies” on page 253 illustrates the concepts discussed in the preceding chapters by using practical examples from our laboratory network.

Chapter 7, “Security scenarios simplified” on page 301 provides a high level series of example networks to be referred to as a guide for a secure e-business environment with an OS/390 (or z/OS) system.

1.5 OS/390 (z/OS) TCP/IP security options overview

IBM Communications Server and other elements of OS/390 or z/OS have provided a number of security options to allow you to build a powerful and secure environment for your e-business applications.

Some examples of the network security options supported in SecureWay Communications Server for OS/390 V2R8 IP are:

- IPSec
  - Support for the recent RFCs 2401-2406 and 2410
  - Manual VPN definition
  - Dynamic VPN creation using the Internet Key Exchange (IKE) protocol
- SSL for TN3270
  - Server authentication
  - Client authentication
- SSL for IBM HTTP (Web) Server
  - Server authentication
  - Client authentication
- SSL application programming interface
- UNIX policy agent, which may be used to block network traffic and control the number of connections allowed

IBM Communications Server for OS/390 V2R10 IP has introduced the following security functions:

- TCP/IP stack security
  - Network access control
  - Port access control
  - Stack access control
- IPSec
  - On-Demand VPN support
- SSL for TN3270
  - System SSL support, including RACF common keyring support
  - TLS-based Telnet security support, which is being defined in an on-going draft RFC.
- FTP anonymous user support enhancements
- syslogd isolation
- Express Logon Feature (ELF)
  - CS for OS/390 V2R10 IP requires the three-tier network design.
  - z/OS V1R2 supports both the three-tier and two-tier network design.

  Note: ELF two-tier network support has been added to CS for OS/390 V2R10 IP by APAR PQ47742.

With the following security options, z/OS IBM Communications Server V1R2 has enhanced its security abilities:

- Kerberized UNIX application support, including the UNIX Telnet server, the UNIX rsh server, the FTP server/client and the LDAP server
- TLS-enabled FTP server/client, as well as the SOCKSified FTP client
- TN3270 server Express Logon Feature support
- Intrusion Detection Services (IDS), done by the traffic regulation and management daemon (TRMD) in cooperation with the UNIX policy agent

Chapter 2. TCP/IP security overview

This chapter discusses the network security techniques available with TCP/IP, and provides an overview of a number of solutions for addressing security issues in networks.

The field of network security in general and of TCP/IP security in particular is very wide, so in this chapter we concentrate on the most recent and most widely used security techniques. The following topics are covered:

2.1, "Basic concepts of cryptography and digital certificates" on page 11

2.2, "Firewall concepts" on page 24

2.3, "Virtual private network (VPN) and IPSec" on page 29

2.4, "Secure Sockets Layer" on page 36 (SSL)

2.5, "Transport Layer Security protocol (TLS)" on page 41

2.6, "Lightweight Directory Access Protocol (LDAP)" on page 42

2.7, "Kerberos-based security system" on page 46

2.8, "The OS/390 UNIX System Services policy agent" on page 53

For more details on the concepts covered in this chapter, please see *TCP/IP Tutorial and Technical Overview*, GG24-3376.

2.1 Basic concepts of cryptography and digital certificates

If you are sending data in the clear over a network that is not completely under your control from the receiver to the sender, you will be unable to ensure the following security functions:

Privacy Anyone who is able to intercept your data might be able to read it.

Integrity An intermediary might be able to alter your data.

Accountability or nonrepudiation
It may be impossible to determine the originator of a message with confidence, and thus the person who sent the message could deny being the originator.

Security functions such as identification and authentication are also impacted because if authentication data such as passwords are sent without integrity and privacy, they can be intercepted in transit between sender and receiver, making the authentication compromised and worthless.

To ensure privacy, integrity and accountability in nonsecure networks, cryptographic procedures need to be used. Today, two distinct classes of encryption algorithms are in use: symmetric and asymmetric algorithms. They are fundamentally different in *how* they work, and thus in *where* they are used.

2.1.1 Symmetric encryption algorithms

An encryption algorithm is called symmetric because the same key that is used to encrypt the data is also used to decrypt the data and recover the clear text (see Figure 1). The cipher and decipher processes are usually mathematically complex, nonlinear permutations.

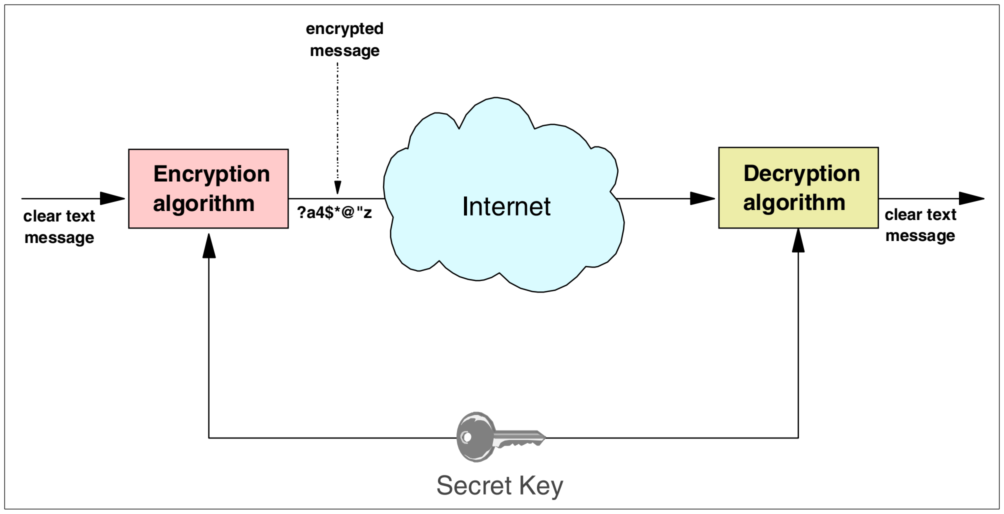


Figure 1. Symmetric encryption and decryption: using the same key

Symmetric algorithms are usually efficient in terms of processing power, so they are ideal for encryption of bulk data. However, they have one major drawback, which is key management. The sender and receiver on any secure connection must share the same key; in a large network where thousands of users may need to communicate securely, it is extremely difficult to manage the distribution of keys so as not to compromise the integrity of any one of them.

Frequently used symmetric algorithms include:

DES	Data Encryption Standard. Developed in the 1970s by IBM scientists, it uses a 56-bit key. Stronger versions called Triple DES have been developed that use three operations in sequence: “2-key Triple DES” encrypts with key 1, decrypts with key 2, and encrypts again with key 1. The effective key length is 112 bits. “3-key Triple DES” encrypts with key 1, decrypts with key 2, and encrypts again with key 3. The effective key length is 168 bits.
CDMF	Commercial Data Masking Facility. This is a version of the DES algorithm approved for use outside the U.S. and Canada (in times when export control was an issue). It uses 56-bit keys, but 16 bits of the key are known, so the effective key length is 40 bits.
RC2	Developed by Ron Rivest for RSA Data Security, Inc. A block cipher with variable key lengths operating on 8-byte blocks. Key lengths of 40, 56, 64, and 128 bits are in use.
RC4	Developed by Ron Rivest for RSA Data Security, Inc. A stream cipher operating on a bit stream. Key lengths of 40 bits, 56 bits, 64 bits, and 128 bits are in use. The RC4 algorithm always uses 128-bit keys; the shorter key lengths are achieved by “salting” the key with a known, non-secret random string.

AES	As a result of a contest for a follow-on standard to DES held by the National Institute for Standards and Technology (NIST), the Rijndael algorithm was selected. This is a block cipher created by Joan Daemen and Vincent Rijmen with variable block length (up to 256 bits) and variable key length (up to 256 bits).
IDEA	The International Data Encryption Algorithm was developed by James Massey and Xueija Lai at ETH in Zurich. It uses a 128-bit key and is faster than triple DES.

DES is probably the most scrutinized encryption algorithm in the world. Much work has been done to find ways to break DES, notably by Biham and Shamir, but also by others. However, a way to break DES with appreciably less effort than a brute-force attack (breaking the cipher by trying every possible key) has not been found.

Both RC2 and RC4 are proprietary, confidential algorithms that have never been published. They have been examined by a number of scientists under non-disclosure agreements.

With all the ciphers listed above, it can be assumed that a brute-force attack is the only means of breaking the cipher. Therefore, the work factor depends on the length of the key. If the key length is n bits, the work factor is proportional to 2**(n-1).

Today, a key length of 56 bits is generally only seen as sufficiently secure for applications that do not involve significant amounts of money or critically secret data. If specialized hardware is built (such as the machine built by John Gilmore and Paul Kocher for the Electronic Frontier Foundation), the time needed for a brute-force attack can be reduced to about 100 hours or less (see: *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design,* by Electronic Frontier Foundation, John Gilmore (Editor), 1988). Key lengths of 112 bits and above are seen as unbreakable for many years to come, since the work factor rises exponentially with the size of the key.

2.1.2 Asymmetric encryption algorithms

Asymmetric encryption algorithms are so called because the key that is used to encrypt the data cannot be used to decrypt the data; a different key is needed to recover the clear text (see Figure 2 on page 14). This key pair is called a public key and a private key. If the public key is used to encrypt the data, the private key must be used to recover the clear text. If data is encrypted with the private key, it can only be decrypted with the public key.

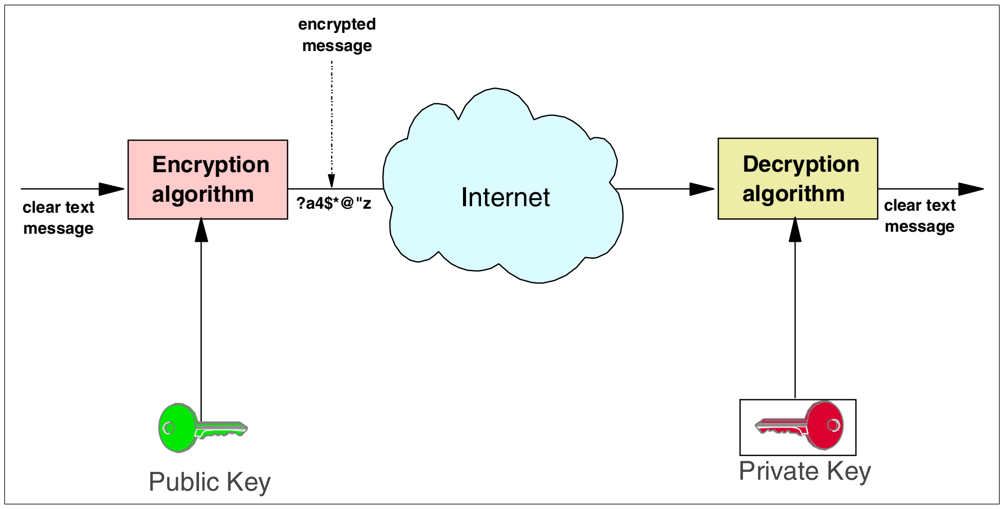


Figure 2. Public-key cryptography: using a key pair

Asymmetric encryption algorithms, commonly called Public-Key Cryptosystems Standards (PKCS), are based on mathematical algorithms. The basic idea is to find a mathematical problem that is very hard to solve. The algorithm in most widespread use today is RSA. However, some companies have begun to implement public-key cryptosystems based on so-called “elliptic curve” algorithms. With the growing proliferation of IPSec, the Diffie-Hellman algorithm is gaining popularity. A brief overview of all three methods follows:

RSA	Invented 1977 by Rivest, Shamir, and Adleman (who formed RSA Data Security Inc.). The idea behind RSA is that integer factorization of very large numbers is extremely hard to do. Key lengths of public and private keys are typically 512 bits, 768 bits, 1024 bits, or 2048 bits. The work factor for RSA with respect to key length is sub-exponential, which means the effort does not rise exponentially with the number of key bits. It is roughly 2**(0.3*n).
Elliptic Curve	Public-key cryptosystems based on elliptic curves use a variation of the mathematical problem of finding discrete logarithms. It has been stated that an elliptic curve cryptosystem implemented over a 160-bit field has roughly the same resistance to attack as RSA with a 1024-bit key length. Properly chosen elliptic curve cryptosystems have an exponential work factor (which explains why the key length is so much smaller). Elliptic curve cryptosystems are now standardized by FIPS PUB 186-2, the digital signature standard (January 2000).
Diffie-Hellman	W. Diffie and M.E. Hellman, the inventors of public key cryptography, published this algorithm in 1976. The mathematical problem behind Diffie-Hellman is computing a discrete logarithm. Both parties have a public-private key pair each; they are collectively generating a key only known to them. Each party uses its own private key and the public key of the other party in the key generation process. Diffie-Hellman public keys are often called *shares*.

The beauty of asymmetric algorithms is that they are not subject to the key management issues that beset symmetric algorithms. Your public key is freely available to anyone, and if someone wants to send you a message he or she encrypts it using that key. Only you can understand the message, because only you have the private key. Asymmetric algorithms are also very useful for authentication. Anything that can be decrypted using your public key must have been encrypted using your private key, in other words, by you.

2.1.3 Performance issues of cryptosystems

Elliptic curve cryptosystems are said to have performance advantages over RSA in decryption and signing. While the possible differences in performance between the asymmetric algorithms are somewhere in the range of a factor of 10, the performance differential between symmetric and asymmetric cryptosystems is far more dramatic.

For instance, it takes about 1000 times as long to encrypt the same data with RSA (an asymmetric algorithm) than with DES (a symmetric algorithm), and implementing both algorithms in hardware does not change the odds in favor of RSA.

As a consequence of these performance issues, the encryption of bulk data is usually performed using a symmetric cryptosystem, while asymmetric cryptosystems are used for electronic signatures and in the exchange of key material for secret-key cryptosystems. With these applications, only relatively small amounts of data need to encrypted and decrypted, and the performance issues are less important.

2.1.4 Cryptosystems for data integrity

Data integrity is the ability to assert that the data received over a communication link is identical to the data sent. Data integrity in an insecure network requires the use of cryptographic procedures. However, it does not imply that only the receiver is able to read the data, as with data privacy. Data could be compromised not only by an attacker, but also by transmission errors (although those are normally handled by transmission protocols such as TCP).

2.1.4.1 Message digest algorithms

A message digesting algorithm (often also called a *digital hash*) is an algorithm that “digests” (condenses) a block of data into a shorter string (usually 128 or 160 bits), which is called a message digest, Secure Hash, or Message Integrity Code (MIC). See Figure 3 for a graphical representation. The principle behind message digesting algorithms is as follows:

- The message cannot be recovered from the message digest.
- It is very hard to construct a block of data that has the same message digest as another given block.

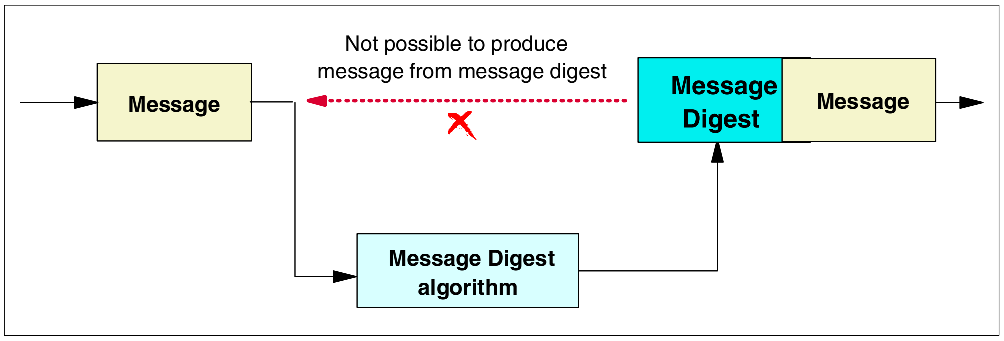


Figure 3. Message digest

Common message digest algorithms are:

MD2	Developed by Ron Rivest of RSA Data Security, Inc. The algorithm is mostly used for PEM (Privacy Enhanced Mail) certificates. MD2 is fully described in RFC 1319. Since weaknesses have been discovered in MD2, its use is discouraged.
MD5	Developed in 1991 by Ron Rivest. The algorithm takes as input a message of arbitrary length and produces as output a 128-bit message digest of the input. The MD5 message digest algorithm is specified in RFC 1321, *The MD5 Message-Digest Algorithm*. Collisions have been found in MD5 (see Hans Dobbertin: *Cryptanalysis of MD5 Compress*, available at `http://www.cs.ucsd.edu/users/bsy/dobbertin.ps`).
SHA-1	Developed by the National Security Agency (NSA) of the U.S. Government. The algorithm takes as input a message of arbitrary length and produces as output a 160-bit "hash" of the input. SHA-1 is fully described in standard FIPS PUB 180-1, also called the Secure Hash Standard (SHS). SHA-1 is generally recognized as the strongest and most secure message digesting algorithm.
SHA-256, SHA-512	Developed by the National Security Agency (NSA) of the U.S. Government. The security of a hash algorithm against collision attacks is half the hash size and this value should correspond with the key size of encryption algorithms used in applications together with the message digest. Since SHA-1 only provides 80 bits of security against collision attacks, this is deemed inappropriate for the key lengths of up to 256 bits planned to be used with AES. Therefore, extensions to the Secure Hash Standard (SHS) have been developed. SHA-256 provides a hash size of 256 bits, while SHA-512 provides a hash size of 512 bits.

2.1.4.2 Message digests for data integrity

The sender of a message (block of data) uses an algorithm, for example, SHA-1, to create a message digest from the message (see Figure 4). The message digest can be sent together with the message to provide data integrity. The receiver runs the same algorithm over the message and compares the resulting message digest to the one sent with the message. If both match, the message is unchanged.

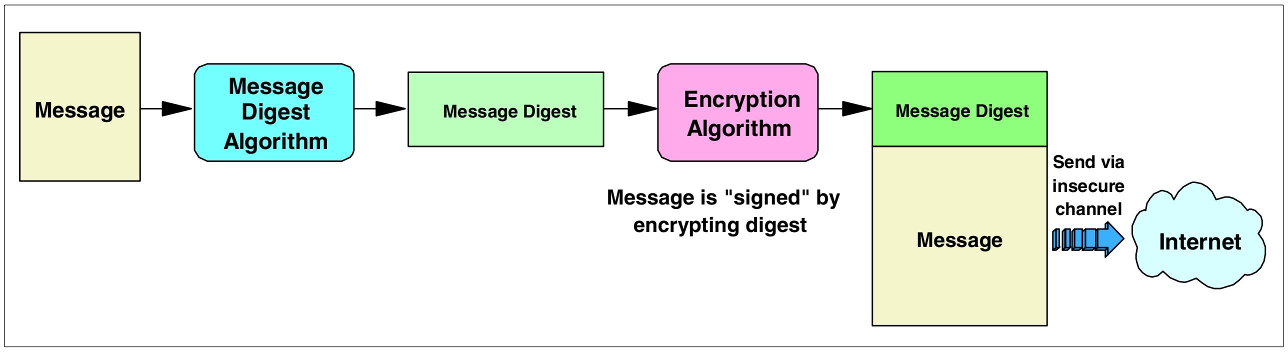


Figure 4. Message digest for data integrity

The message digest should not be sent in the clear: Since the digest algorithms are well-known and no key is involved, a man-in-the-middle could not only forge the message but also replace the message digest with that of the forged message. This would make it impossible for the receiver to detect the forgery. The solution for this is to use a message digest algorithm that uses cryptography when creating the message digest — that is, to use a Message Authentication Code (MAC).

2.1.4.3 Message authentication codes (MAC)

Secret-key cryptographic algorithms, such as DES, can be used for encryption with message digests. A disadvantage is that, as in secret-key cryptosystems, the keys must be shared by sender and receiver. Furthermore, since the receiver has the key that is used in MAC creation, this system does not offer a guarantee of non-repudiation. That is, it is theoretically possible for the receiver to forge a message and claim it was sent by the sender. Therefore, message authentication codes are usually based on public/private-key encryption in order to provide for non-repudiation. This is discussed further in 2.1.5, “Digital signatures” on page 18.

Note: Some message authentication codes are based on symmetric-based cryptography, not public key encryption. Examples are SHA and MD5, which use one-way symmetric key.

2.1.4.4 Keyed hashing for message authentication (HMAC)

H. Krawczyk and R. Canetti of IBM Research and M. Bellare of UCSD invented a method to create a message authentication code called HMAC, which is defined in RFC 2104 as a proposed Internet standard. A simplified description of how to create the HMAC is as follows. The key and the data are concatenated and a message digest is created. The key and this message digest are again concatenated for better security, and another message digest is created, which is the HMAC.

HMAC can be used with any cryptographic hash function. Typically, either MD5 or SHA-1 are used. In the case of MD5, a key length of 128 bits is used (the block length of the hash algorithm). With SHA-1, 160-bit keys are used. Using HMAC actually improves the security of the underlying hash algorithm. For instance, some collisions (different texts that result in the same message digest) have been found in MD5. However, they cannot be exploited with HMAC, therefore the weakness in MD5 does not affect the security of HMAC-MD5.

HMAC is now a PKCS#1 V.2 standard for RSA encryption (proposed by RSA Inc. after weaknesses were found in PKCS#1 applications). For further details, see `http://www.ietf.org/rfc.html`. HMAC is also used in the Transport Layer Security (TLS) protocol, the successor to SSL.

2.1.4.5 Message authentication used with SSL

In the Secure Sockets Layer protocol (SSL), a slightly different MAC algorithm has been implemented. A simplified description follows: the MAC write-secret and the sequence number of the message are concatenated with the data, and a message digest is created. The MAC write-secret and this message digest are again concatenated for better security, and another message digest is created which is the MAC. Again, for the hash function, either MD5 or SHA-1 can be used. If compression is used, the text is compressed before the MAC is calculated. For further details, see:

`http://home.netscape.com/eng/ssl3/draft302.txt`

2.1.5 Digital signatures

Digital signatures are an an additional means of securing data integrity. While data integrity only ensures that the data received is identical to the data sent, digital signatures go a step further: they provide non-repudiation. This means that the sender of a message (or the signer of a document) cannot deny authorship, similar to signatures on paper. As illustrated in Figure 5, the creator of a message or electronic document that is to be signed uses a message digesting algorithm such as MD5 or SHA-1 to create a message digest from the data. The message digest and some information that identifies the sender are then encrypted with an asymmetric algorithm using the sender's private key. This encrypted information is sent together with the data.

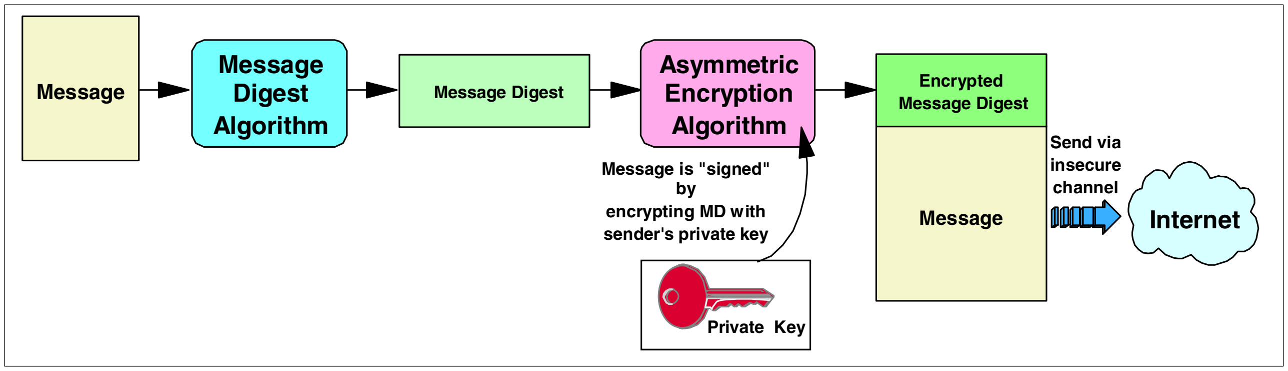

Message
Message Digest Algorithm
Message Digest
Asymmetric Encryption Algorithm
Encrypted Message Digest
Message is "signed" by encrypting MD with sender's private key
Private Key
Message
Send via insecure channel
Internet

Figure 5. Digital signature creation

The receiver, as shown in Figure 6, uses the sender's public key to decrypt the message digest and identification of the sender. He or she will then use the message digesting algorithm to compute the message digest from the data. If this message digest is identical to the one recovered after decrypting the digital signature, the signature is recognized as valid proof of the authenticity of the message.

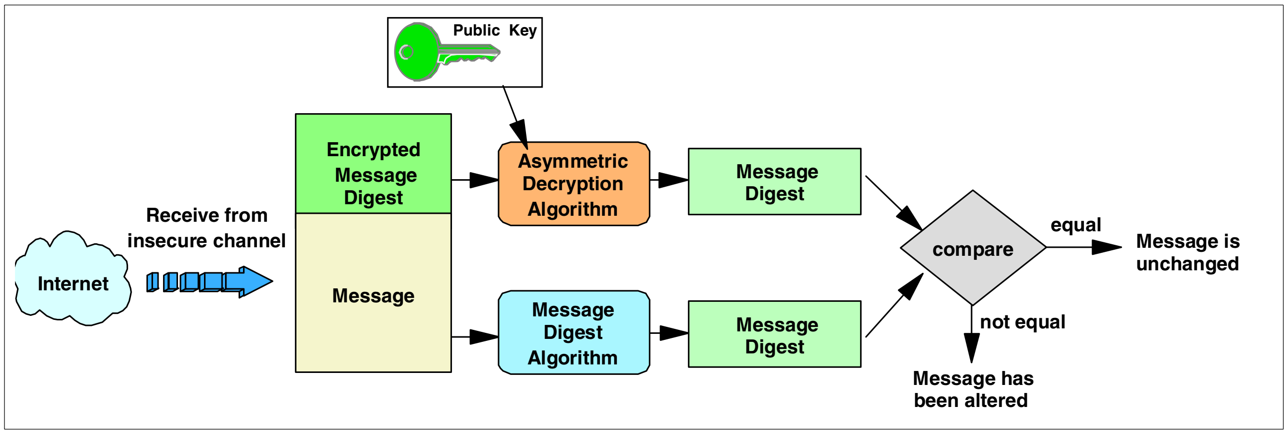


Figure 6. Digital signature verification

With digital signatures, only public-key cryptosystems can be used. If secret-key cryptosystems are used to encrypt the signature, it would be very difficult to make sure that the receiver (having the key to decrypt the signature) could not misuse this key to forge a signature of the sender. The private key of the sender is known to nobody else, so nobody is able to forge the sender's signature.

Note the difference between encryption using public-key cryptosystems and digital signatures:

- With encryption, the sender uses the receiver's public key to encrypt the data, and the receiver decrypts the data with his private key. This means everybody can send encrypted data to the receiver that only the receiver can decrypt. See Figure 7 for a graphical representation.

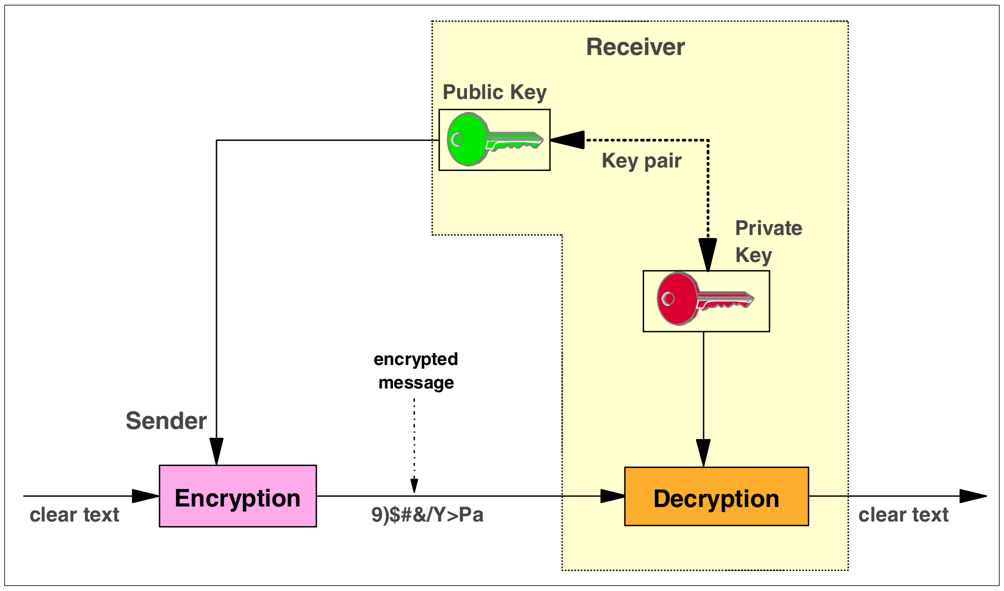


Figure 7. Encrypting data with the receiver's public key

- With digital signatures, the sender uses his private key to encrypt his signature, and the receiver decrypts the signature with the sender's public key.

This means that only the sender can encrypt the signature, but everybody who receives the signature can decrypt and verify it.

The tricky part with digital signatures is the trustworthy distribution of public keys, since a genuine copy of the sender's public key is required by the receiver. A solution to this problem is provided by digital certificates, which are discussed next.

2.1.6 Public Key Infrastructure

A Public Key Infrastructure (PKI) offers the basis for practical usage of public key cryptography. A PKI defines the rules and relationships for certificates and Certificate Authorities (CAs). It defines the fields that can or must be in a certificate, the requirements and constraints for a CA in issuing certificates, and how certificate revocation is handled.

PKI has been exploited in many applications or protocols, such as Secure Sockets Layer (SSL), Secure Multimedia Internet Mail Extensions (S/MIME), IP Security (IPSec), Secure Electronic Transactions (SET), and Pretty Good Privacy (PGP). We will describe PKI here, only insofar as its use with Web serving and Secure Sockets Layer (SSL) is concerned. For more information on PKI, see the redbook *Deploying a Public Key Infrastructure*, SG24-5512.

2.1.6.1 Digital certificates

When using a PKI, the user must be confident that the public key belongs to the correct remote person (or system) with which the digital signature mechanism is to be used. This confidence is obtained through the use of public key digital certificates. A digital certificate is analogous to a passport: the passport certifies the bearer's identity, address and citizenship. The concepts behind passports and other identification documents (for instance, drivers' licenses) are very similar to those that are used for digital certificates.

Passports are issued by a trusted authority, such as a Government passport office. A passport will not be issued unless the person who requests it has proven their identity and citizenship to the authority. Specialized equipment is used in the creation of passports to make it very difficult to alter the information in it or to forge a passport altogether. Other authorities, for instance the border police in other countries, can verify a passport's authenticity. If they trust the authority that issued the document, they implicitly trust the passport.

A digital certificate serves two purposes: it establishes the owner's identity and it makes the owner's public key available. Similar to a passport, a certificate must be issued by a trusted authority, the CA, and, like a passport, it is issued only for a limited time. When its expiration date has passed, it must be replaced.

Trust is a very important concept in passports, as well as in digital certificates. In the same way as, for instance, a passport issued by the governments of some countries, even if recognized to be authentic, will probably not be trusted by the US authorities, each organization or user has to determine whether a CA can be accepted as trustworthy.

As an example, a company might want to issue digital certificates for its own employees from its own Certificate Authority; this could ensure that only authorized employees are issued certificates, as opposed to certificates being obtained from other sources such as a commercial entity such as VeriSign.

The information about the certificate owner's identity is stored in a format that follows RFC 2253 and the X.520 recommendation, for instance: CN=Ulrich Boche, O=IBM Corporation; the complete information is called the owner's distinguished name (DN). The owner's distinguished name and public key and the CA's distinguished name are digitally signed by the CA; that is, a message digest is calculated from the distinguished names and the public key. This message digest is encrypted with the private key of the CA.

Figure 8 shows the layout of a digital certificate.

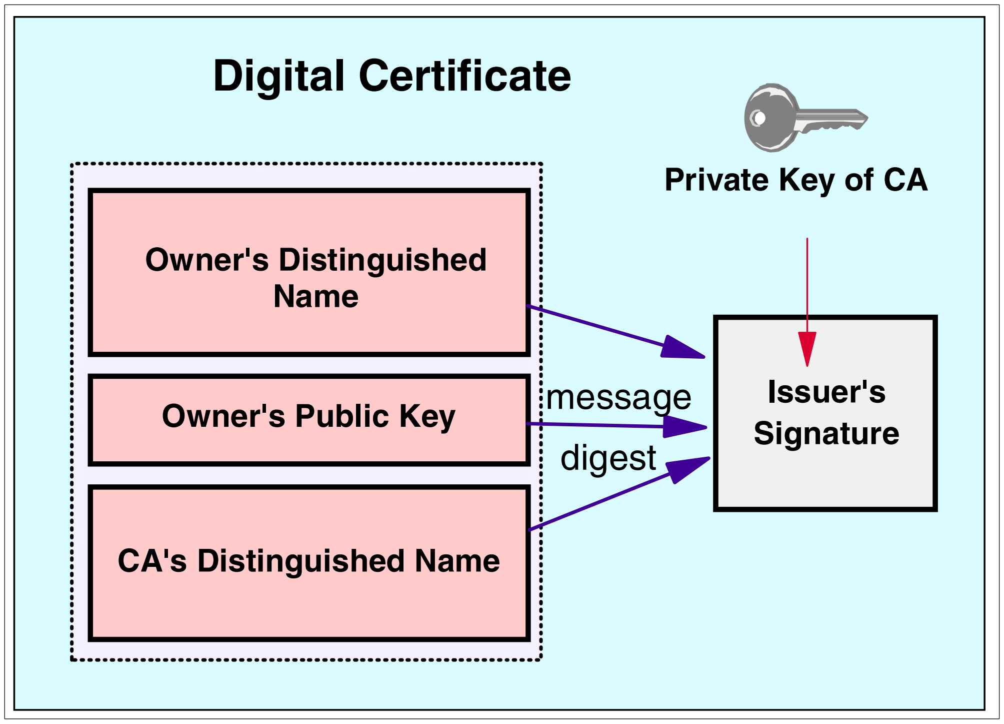


Figure 8. Simplified layout of a digital certificate

The digital signature of the CA serves the same purpose as the special measures taken for the security of passports, such as laminating pages with plastic material: it allows others to verify the authenticity of the certificate. Using the public key of the CA, the message digest can be decrypted. The message digest can be recreated; if it is identical to the decrypted message digest, the certificate is authentic.

Security considerations for certificates

If I send my certificate with my public key in it to someone else, what keeps this person from misusing my certificate and posing as myself? The answer is: my private key.

A certificate alone can never be proof of anyone's identity. The certificate just allows the identity of the certificate owner to be verified by providing the public key that is needed to check the certificate owner's digital signature. Therefore, the certificate owner must protect the private key that matches the public key in the certificate. If the private key is stolen, the thief can pose as the legitimate owner of the certificate. Without the private key, a certificate cannot be misused.

An application that authenticates the owner of a certificate cannot accept just the certificate. A message signed by the certificate owner should accompany the certificate. This message should use elements such as sequence numbers, time stamps, challenge-response protocols, or other data that allow the authenticating application to verify that the message is a “fresh” signature from the certificate owner and not a replayed message from an impostor.

2.1.6.2 Certificate Authorities and trust hierarchies

A user of a security service requiring knowledge of a public key generally needs to obtain and validate a certificate containing the required public key. To verify that the certificate is authentic, the receiver needs the public key of the CA that issued the certificate.

Most Web browsers come configured with the public keys of common CAs (such as Verisign). However, if the user does not have the public key of the CA that signed the certificate, an additional certificate would be needed in order to obtain that public key. In general, a chain of multiple certificates may be required, comprising a certificate of the public key owner signed by a CA, and possibly additional certificates of CAs signed by other CAs. Many applications that send a subject's certificate to a receiver send, not only just that certificate, but also all the CA certificates necessary to verify the certificate up to the root.

2.1.6.3 Obtaining and storing certificates

As we have discussed, certificates are issued by a CA. Clients usually request certificates by visiting the CA’s Web site. After verifying the validity of the request, the CA sends back the certificate in an e-mail message or allows it to be downloaded.

Requesting server certificates

Server certificates can be either self-signed or they can be signed by an external CA. The server environment will determine which kind of certificate should be used: in an intranet environment, it is generally appropriate to use self-signed certificates. In an environment where external users are accessing the server over the Internet, it is usually advised to acquire a server certificate from a well-known CA, because the steps needed to import a self-signed certificate might seem obscure, and most users will not have the ability to discern whether the action they are performing is of trivial consequence or not. It should also be noted that a root CA certificate received over an untrusted channel, such as the Internet, does not deserve any kind of trust.

2.1.6.4 Certificate management in OS/390 and z/OS

To manage certificates on an OS/390 system, appropriate software is required. One alternative is to use a utility program that runs as an OS/390 UNIX program and works with a key database. The other alternative is to use certificate management services provided by RACF that use the `RACDCERT` command and work with key rings stored in the RACF database.

Certificate management utilities in OS/390 and z/OS

To create a self-signed server certificate or to create a certificate request that can be sent to a CA, a key management utility can be used. Which utility needs to be used depends on the SSL support that the certificate and key pair are to be used for. Different levels of SSL tool kits contained in applications such as the web server or the TN3270E server use the MKKF or IKEYMAN utilities. System SSL introduced the GSKKYMAN utility and applications that make use of System SSL require this utility to be used.

Note that the key database are not compatible: a key database created by MKKF can be converted (but not directly used) by ikeyman or GSKKYMAN. A key database created by ikeyman can be used by GSKKYMAN, but once it has been modified by GSKKYMAN it can no longer be used by IKEYMAN due to a version mismatch.

Should it be necessary to copy a server certificate into a different key database, the certificate management utilities allow exporting certificates to PKCS#12 files, which can then be imported into other key databases.

Using RACF to store server certificates and private keys

As of OS/390 V2.8, RACF can be used to manage digital server certificates and key rings. Digital certificates can be stored in the RACF database.

If the S/390 Cryptographic Coprocessor Feature and its software interface, Integrated Cryptographic Service Facility (ICSF), are used, the private keys belonging to the server certificates can be stored in the Public Key Data Set encrypted under a 168-bit Triple-DES master key.

When RACF is used to store server certificates and private keys, a key database is not used. This also makes the use of a password stash file, which might be viewed as a security exposure, unnecessary.

The following table shows the different product releases and the key management utilities they require:

Table 1. Key management packages used by SSL products

Product	MKKF	IKEYMAN	GSKKYMAN	RACDCERT
Domino Go Webserver 5.0		X		
IBM HTTP Server 5.1		X		
IBM HTTP Server 5.2		X		
IBM HTTP Server 5.3			X	X
TN3270 Server OS/390 V2R6	X			
TN3270 Server OS/390 V2R7	X			
TN3270 Server OS/390 V2R8, R9			X	
TN3270 Server OS/390 V2R10			X	X
LDAP Server OS/390 V2R5, R6	X			
LDAP Server OS/390 V2R7			X	
LDAP Server OS/390 V2R8 and up			X	X

Product	MKKF	IKEYMAN	GSKKYMAN	RACDCERT
Note: Each version of Webserver or HTTP server is included in the following release(s) of OS/390: - Domino Go Webserver Version 5.0 - OS/390 V2R6 - IBM HTTP Server Version 5.1 - OS/390 V2R7 - IBM HTTP Server Version 5.2 - OS/390 V2R8 and V2R9 - IBM HTTP Server Version 5.3 - OS/390 V2R10				

2.2 Firewall concepts

A firewall machine is a computer used to separate a secure network from a non-secure network (see Figure 9). Such networks are typically based on the TCP/IP protocol, but the concept of a firewall concept is not restricted to just TCP/IP.

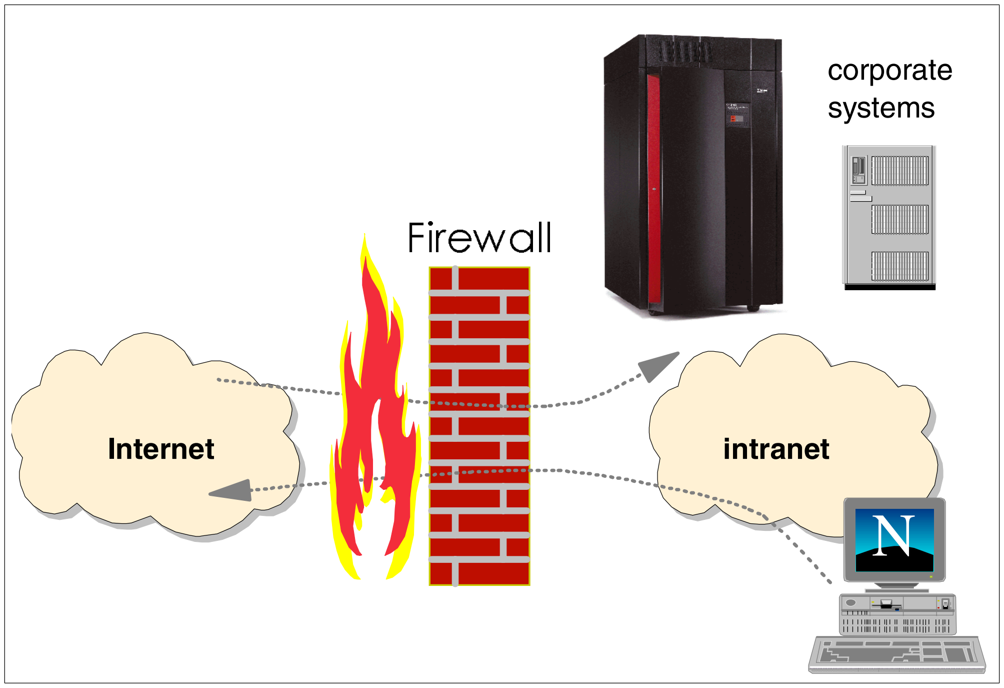


Figure 9. The firewall concept

Firewalls have become an important concept in TCP/IP-based networks, because the global Internet is a TCP/IP-based network and is often perceived as being an non-secure place to enter or traverse. Yet you still want your intranet (perceived as being a secure place) to be connected to the non-secure Internet.

The reasons for establishing connections between an intranet and the Internet are many, but generally fall into two categories:

- You want to provide a service to the Internet community or want to conduct business on the Internet.
- You want to allow your internal employees to access the vast amount of services on the Internet, as well as the ability to exchange or share information with other users on the Internet or through the Internet.

At this point, it might be useful to define the following terms:

- The term *intranet* refers to an internal TCP/IP network.
- The term *Internet* refers to the World Wide Web, and the associated infrastructure of news groups, e-mail, chat rooms and other services.
- The term *extranet* refers to TCP/IP networks of different companies connected with a secure connection, perhaps using virtual private network technology (VPN).

Doing e-business on the Internet is very different from just serving static information out of a Web server. e-business means that you have to establish an environment where users on the Internet are able to interact with the applications and data that your daily existence as a company is based on and relies upon.

That data and those applications are likely, to a large extent, to be located in your OS/390 environment, which means that you probably already are, or in the near-term future will be, challenged with the request to establish Internet access to your OS/390 production environment.

When you connect your intranet to the Internet and define a strategy for how your firewall should function, you may think that it is sufficient to block all types of traffic that represent a risk, and allow the remaining traffic to pass through the firewall. However, such a strategy is based on the assumption that all risks are known in advance and that existing well-behaving traffic will remain well-behaving; such an assumption is a mistake. New ways of exploiting existing applications and well-known application protocols are being found every week, so an application that may be considered harmless today may be the instrument of an attack tomorrow.

2.2.1 General guidelines for implementing firewalls

A few general guidelines for implementing firewall technologies are worth including in this context.

Before you start connecting your intranet to the Internet, define a security policy for how your firewall should function and how demilitarized zones should be configured. Decide what type of traffic is allowed through the firewall and under what conditions. Decide what kind of servers are to be placed in demilitarized zones and what type of traffic is allowed between the demilitarized zone and the intranet.

When actually configuring your firewall, start by disallowing everything and then proceed by enabling those services you have defined in your security policy. Everything that is not specifically allowed should be prohibited.

If you establish more than a single gateway between your internal network and the Internet, make sure that all gateways implement the same level of security. It is common practice to use different firewall products in a vertical setup (product A between the Internet and the demilitarized zone and product B between the demilitarized zone and the intranet). That way, a malicious hacker exploiting a vulnerability in product A is still stopped by product B. Of course, it does not make sense to use this concept in a horizontal setup (one gateway uses product A, the other one product B) because a malicious hacker will get in at the weakest link.

If you build a perfect firewall on one end of your network while users on the other end dial in to the Internet from their LAN-attached PCs, enabling those PCs to act as IP routers between your internal network and the Internet, a malicious hacker is soon going to exploit that back door into your network instead of wasting his time trying to break through your firewall.

One of the most important aspects of a firewall is its ability to log both successful and rejected access events. However, these logs are worth nothing if you do not set up daily administrative procedures to analyze and react to the information that can be derived from these logs.

By analyzing the firewall logs, you should be able to detect if unauthorized accesses were attempted and if your firewall protection succeeded in rejecting such attacks, or if it failed and allowed an intruder to gain access to resources that should not have been accessed. In addition, it might be a good idea to install an intrusion detection system.

This list is not all-inclusive, but merely points out some of the most important aspects of implementing firewall technologies in your network.

So far, we have only considered the Internet to be the non-secure place, while your internal network has been considered the secure place. However, that may in some situations be an oversimplification. For example, consider a research department that works with highly confidential information. In such an environment, you may want to protect that research department from your regular users by implementing a firewall between your regular internal network and the network in your research department.

2.2.2 Firewall categories

There are many firewall technologies available, but they can in general be grouped into two major categories:

- Those that allow IP packets to be routed between two or more networks, namely packet-filtering routers.
- Those that disable IP routing, but relay data through specialized application programs, namely application-level gateways or proxies.

2.2.2.1 Packet filtering

A packet-filtering router, as shown in Figure 10, is a special type of IP router. What differentiates a firewall packet-filtering router from a normal IP router is that it applies one or more technologies to analyze the IP packets and decide if a packet is allowed to flow through the firewall or not. Such a firewall is sometimes also referred to as a screening filter, or router firewall.

Some packet-filtering techniques only act on data in the headers of individual packets, while others also look at data depending on the type of packet. The traditional packet-filtering router is stateless (each packet is handled independently) but there are products that save state over multiple packages and base their actions on the state information.

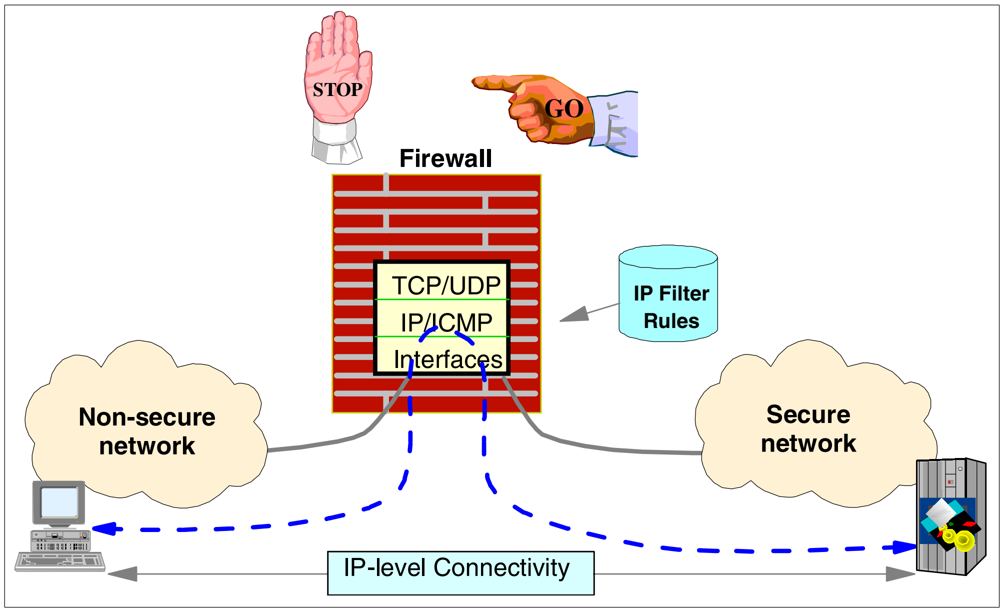


Figure 10. Packet-filtering firewall

2.2.2.2 Application-level gateway

An application-level gateway, sometimes referred to as a bastion host, is a machine that disables IP-level routing between the non-secure network and the secure network, but allows specialized application gateway programs (termed proxies) that run on the firewall to communicate with both the secure network and the non-secure network. See Figure 11.

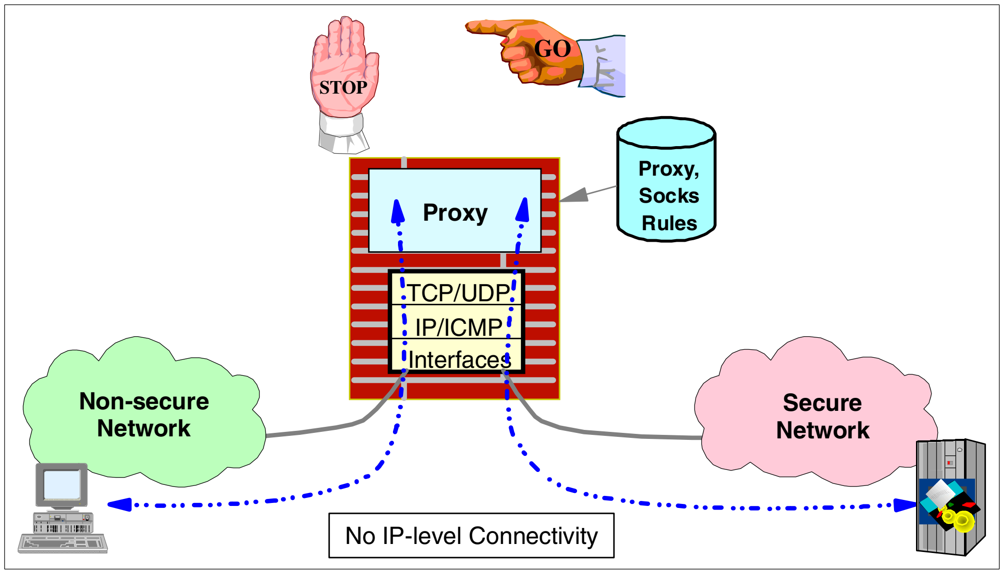


Figure 11. Application gateway firewall

The proxy applications on the firewall act as relay applications between users or applications on the secure and the non-secure networks. Examples of such proxy applications are HTTP or FTP proxy servers. The SOCKS server is also an application-level gateway, but a special kind, sometimes referred to as a circuit level gateway. A SOCKS server can relay all TCP and UDP connections, not just HTTP or FTP sessions. It does not provide any extra packet processing or filtering, and unlike proxy servers, it is often used for outbound connections through a firewall.

A firewall may not always have to be configured as either a packet-filtering router or as a proxy; it may be configured to act as a packet-filtering router for certain application protocols, while it acts as a proxy for other applications.

An excellent discussion of firewall technologies can be found in the redbook, *TCP/IP Tutorial and Technical Overview*, GG24-3376.

2.2.2.3 OS/390 firewall technologies

Starting with OS/390 V2.4, firewall functions have been made available on OS/390. The following functions are supported:

- IP filtering
- Network Address Translation (NAT)
- Virtual private networks (VPN)
- FTP proxy server
- SOCKS server
- Domain name services

We would suggest using OS/390 firewall technologies as tools to protect an OS/390 or z/OS server machine from the network(s) it is connected to, rather than as a firewall that handles all traffic between the Internet and an intranet. For more information on OS/390 firewall technologies, see *OS/390 Firewall Technologies Guide and Reference*, SC24-5835.

2.2.2.4 The demilitarized zone

The demilitarized zone (DMZ) is a term often used when describing firewall configurations. Figure 25 shows a typical example. A DMZ is an isolated subnet between your secure network and the Internet. Much as the no-man's land between two entrenched armies, anyone can enter it, but the only things present are those that you wish to allow access to anyway. Nowadays, we don't plant mines in a demilitarized zone; instead, it is an area in which you place the Web servers and other servers for public access, but which you also wish to protect to some degree.

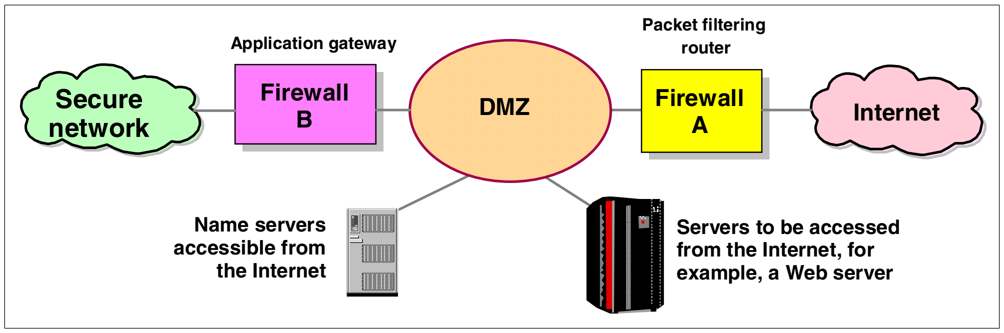

Application gateway
Packet filtering router
Secure network
Firewall B
DMZ
Firewall A
Internet
Name servers accessible from the Internet
Servers to be accessed from the Internet, for example, a Web server

Figure 12. A demilitarized zone

This is achieved by placing an outer firewall (often a packet-filtering router) between the Internet and the servers in the DMZ, and another firewall (often an application-level gateway) between your secure network and the DMZ. The outer firewall is designed to allow into the DMZ only those requests you wish to receive at your Web servers, but could also be configured to block denial-of-service attacks and to perform network address translation of the servers in your DMZ. The inner firewall is designed to prevent unauthorized access to your secure network from the DMZ and also perhaps to prevent unauthorized access from your secure network to the DMZ or the connected non-secure network. IPSec authentication can be used as an alternative means to control access to secure network (see Chapter 7, “Security scenarios simplified” on page 301 for further discussion).

When you put an OS/390 server into a DMZ, we would strongly suggest that you use OS/390 firewall technologies. You should use OS/390 firewall technologies to block all traffic into and out of your OS/390 server that does not belong to the services you are going to offer from this server. This control should be in place even if you already have a filtering router or firewall between the insecure network and this server.

2.3 Virtual private network (VPN) and IPSec

A virtual private network (VPN) provides secure connections across the Internet, by establishing a “tunnel” between two secure networks. It is a generic solution that is both application and protocol independent. A VPN encapsulates the IP datagram into another IP datagram in order to maintain data privacy. It can be used by two disparate parts of a corporation to connect their internal private networks by means of a non-secure network such as the Internet. An example for a VPN configuration is shown in Figure 13.

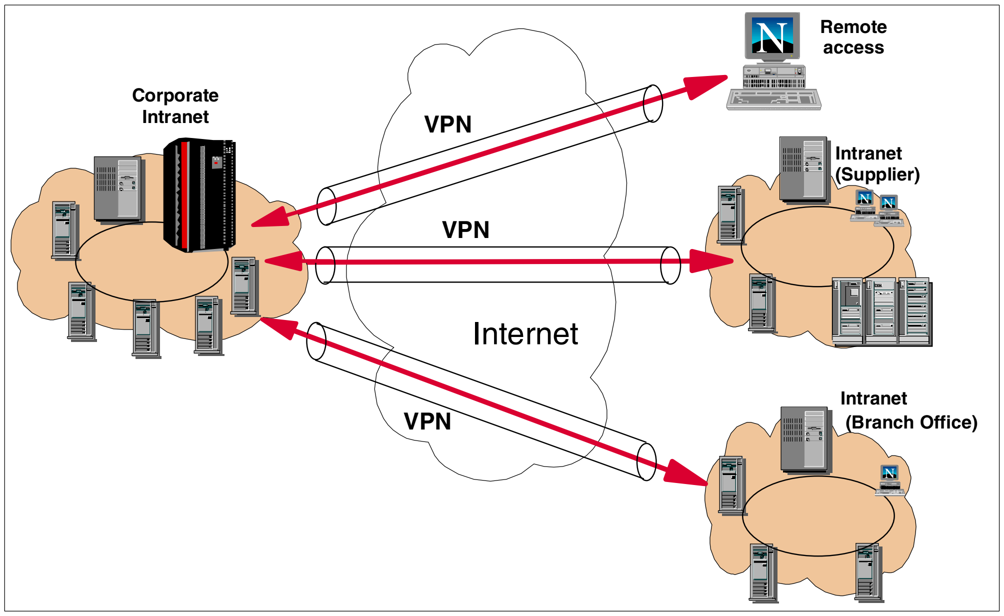


Figure 13. Virtual private networks

2.3.1 IPSec

In Figure 14 the TCP/IP layered protocol stack is shown, with the security-related protocols associated with each layer:

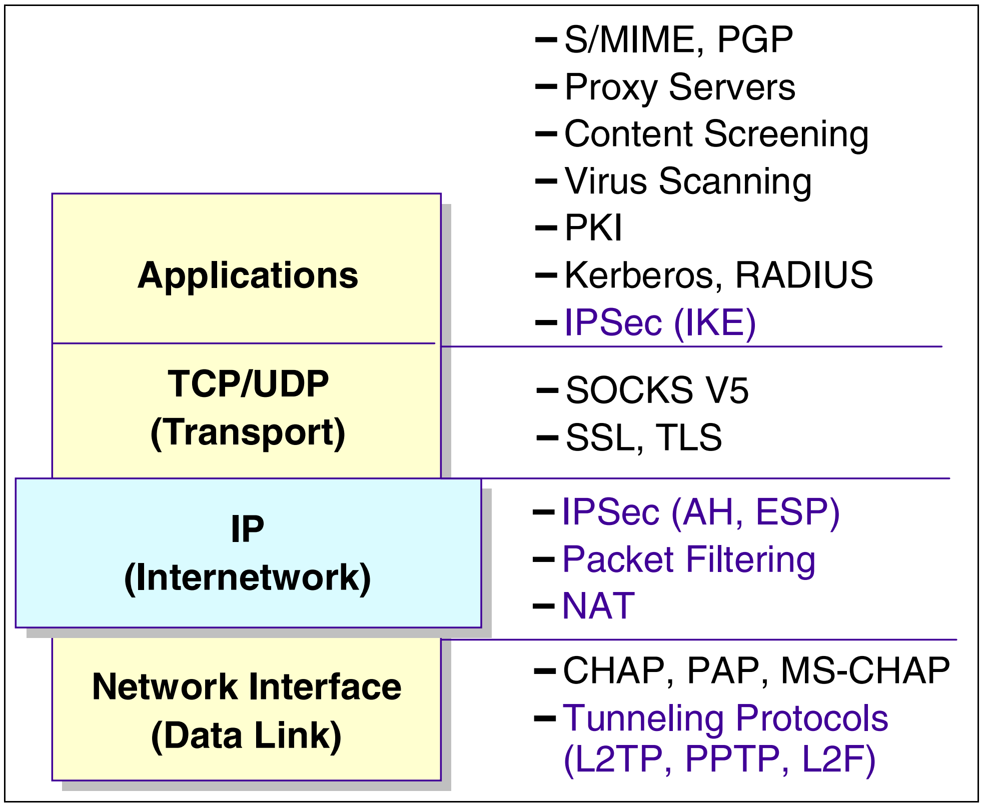


Figure 14. The TCP/IP protocol stack and the security-related protocols

Within the layered communications protocol stack model, the network layer (IP in the case of the TCP/IP stack) is the lowest layer that can provide end-to-end security. Network-layer security protocols provide blanket protection for all upper-layer application data carried in the payload of an IP datagram, without requiring a user to modify the applications.

The IP Security Architecture (IPSec) open framework is defined by the IPSec Working Group of the IETF. IPSec is called a framework because it provides a stable, long-lasting base for providing network-layer security. It can accommodate today's cryptographic algorithms, and can also accommodate newer, more powerful algorithms as they become available. IPv6 implementations must support IPSec, and IPv4 implementations are strongly recommended to do so.

IPSec is comprised of a number of components described in individual RFCs that are designed to operate together:

- Security Protocols - IP Authentication Header (AH) provides data origin authentication, data integrity, and replay protection, while IP Encapsulating Security Payload (ESP) provides data confidentiality, data origin authentication, data integrity, and replay protection.
- Security Associations - an SA is a kind of session between two hosts defining the protocols to be used when transmitting data. ISAKMP (Internet Security Association and Key Management Protocol) is a generic framework for negotiating SAs and keys.

- Key Management - Internet Key Exchange (IKE) provides a method for automatically setting up security associations and managing and exchanging their cryptographic keys.

2.3.1.1 Security Associations

An IPSec Security Association (SA) defines the set of protocols and, with these, the negotiated algorithms and keys that are to be used when transmitting data between two hosts. An SA for data traffic is always unidirectional, so for a pair of hosts that are to communicate securely, at least two SAs, one for each direction, are needed. This differs from other protocols that make use of sessions such as, for instance, SSL; an SSL session covers the transmission in both directions.

A Security Association has three parts:

1. A Security Parameter Index (SPI), a unique 32-bit value identifying the SA
2. The source IP address
3. The protocol used (AH or ESP)

2.3.1.2 Negotiating Security Associations (ISAKMP and IKE)

Before any data can be sent between two hosts using IPSec, an SA needs to be established. The IPSec architecture provides two methods for establishing an SA: a manual tunnel or ISAKMP/IKE.

With manual tunnels, the SA and cryptographic keys are generated on one of the hosts (the tunnel owner), transferred to the other host (the tunnel partner) with an out-band transport mechanism, and then imported. This procedure needs to be repeated whenever the validity of the keys has expired and new cryptographic keys need to be generated.

Contrary to manual tunnels, ISAKMP and IKE provide automatic management of sessions and keys. ISAKMP provides a generic framework for the negotiation of SAs and keying material. It defines the procedures and packet formats to establish, negotiate, modify and delete SAs, but it does not provide any specific key generation techniques or cryptographic algorithms.

Internet Key Exchange (IKE) is based on two protocols: Oakley (see *The Oakley Key Determination Protocol*, by H. Orman; RFC 2412, November 1998) and SKEME (see *SKEME: A Versatile Secure Key Exchange Mechanism for the Internet*, by H. Krawczyk; IEEE Proceedings, 1996). For the key exchange, Diffie-Hellman (DH) shares are used and the shared key thus obtained is used to derive the keys for data encryption and message authentication. Authentication can be performed with one of three alternatives:

- Digital signatures
- Public key encryption
- A shared secret (a key previously known to both parties)

The use of DH shares causes the connection to have a property called “perfect forward secrecy”. This means that even if the keys for one session are completely compromised, the keys for previous sessions are still safe.

Phases: it takes two

Two hosts can communicate with each other in many different ways that may need different sorts of protection. For instance, some traffic may need encryption and authentication while other traffic may only need authentication.

IKE uses a two-phase approach to be able to meet these different needs with minimal overhead. In phase 1, an ISAKMP SA is negotiated to create a secure, authenticated channel between the two hosts. The ISAKMP SA is a single, bi-directional Security Association. In phase 2, the SAs for the individual type of traffic (one SA for each direction) are negotiated using the authenticated channel established in phase 1.

Due to the Diffie-Hellman key exchange and the authentication, phase 1 is computationally rather expensive. Phase 2 does not involve key exchange nor authentication and is much less expensive. Performing phase 1 just once for a pair of hosts and then multiple phase 2 operations for the individual connections is a concept that can improve performance considerably.

Identity protection

In phase 1, certificates and authentication data are exchanged between the hosts. IKE offers *identity protection*, meaning that all information that could identify a host to an attacker or eavesdropper can be encrypted. Depending on whether identity protection is really required, IKE supports two modes for phase 1: *main mode* offers identity protection while *aggressive mode* does not. In main mode, a shared, secret key is established before the identification information (for instance, the host's digital certificate) is sent. For a diagram showing IKE main mode, see Figure 15.

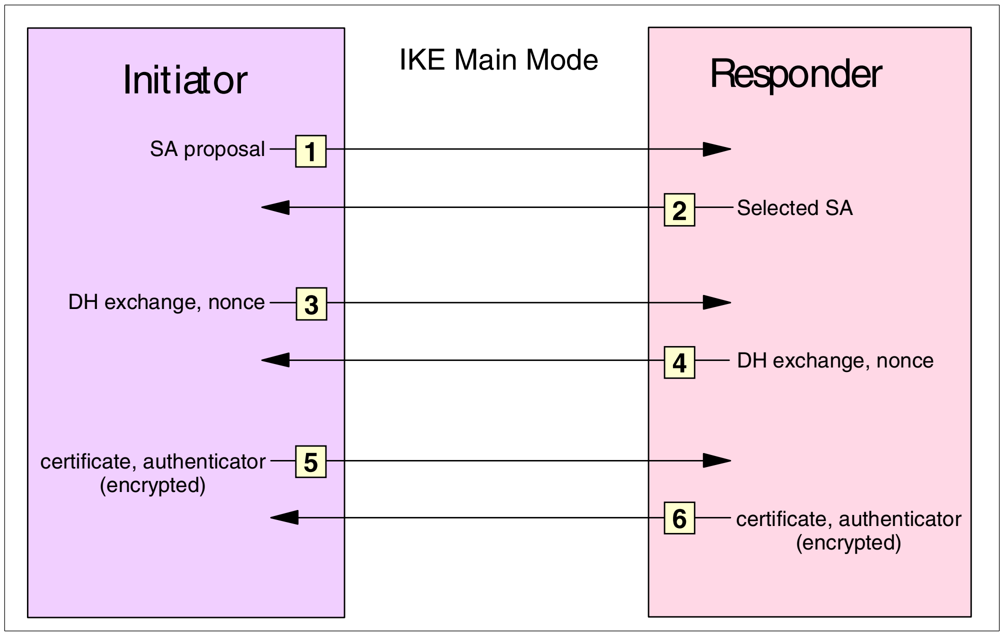


Figure 15. IKE phase 1: main mode

Aggressive mode does not require the DH key exchange to be completed before sending the remaining information. Therefore, there is only one exchange of messages in aggressive mode (see Figure 16).

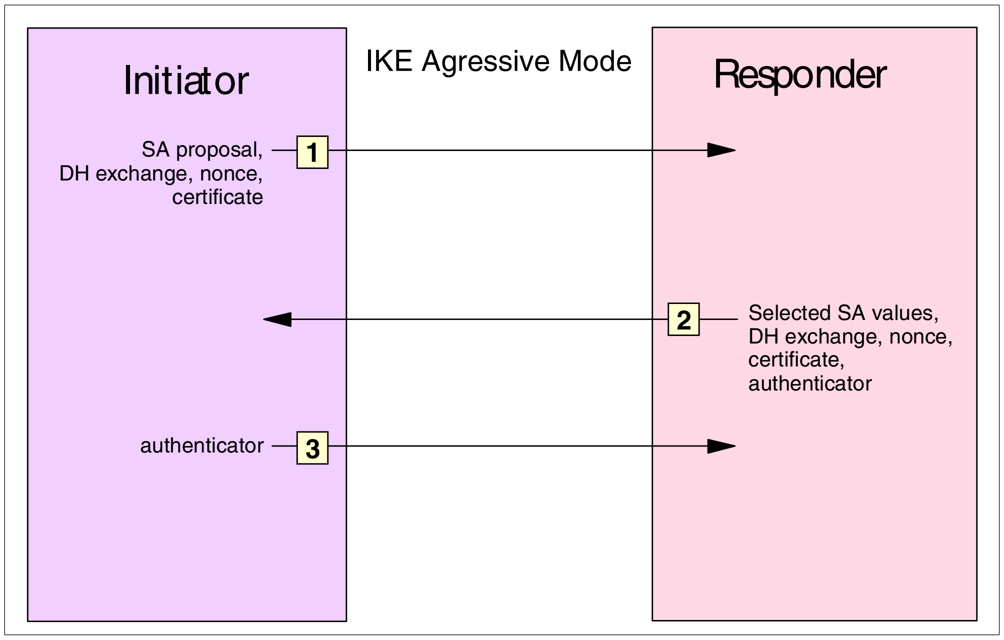


Figure 16. IKE phase 1: aggressive mode

The exchange of messages taking part in phase 2 (negotiation of the SAs for the individual type of traffic) is called *quick mode*. In this mode, the pair of SAs for the intended type of communication is set up. The required keys for encryption and message authentication are generated from the shared key obtained in phase 1.

2.3.1.3 Transmitting data with IPSec

When a host wants to transmit one or more packets to another host it had not contacted before, it will perform the necessary IKE exchanges to set up the required SAs with the other hosts. Once this has all been performed and the necessary keys are generated, the host proceeds with sending the first packet.

IPSec has two formats for sending data that are serving slightly different purposes. *Authentication Header* (AH) provides for message authentication and replay protection, whereas *Encapsulating Security Payload* (ESP) provides for data encryption in addition to message authentication and replay protection. The SA for a communication selects whether AH, ESP, or a combination of both is to be used.

Depending on the type of VPN connection between the two hosts, there are two modes, *tunnel mode* and *transport mode*, that are to be used.

ESP and AH in transport mode

If a VPN connection is being established between two hosts that are the endpoints for the packets transmitted between them, transport mode should be used. Figure 17 shows the format of Authentication Header (AH) in transport mode.

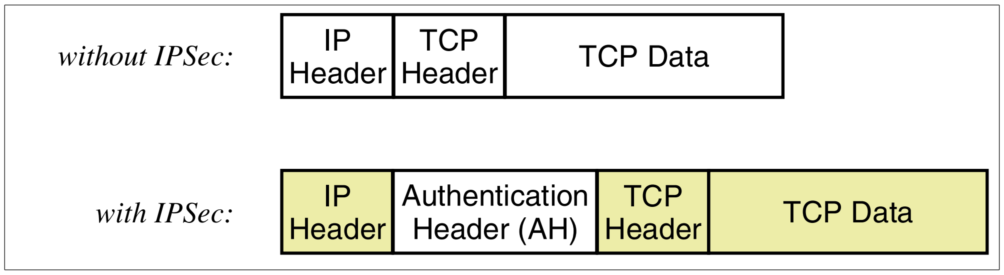


Figure 17. AH in transport mode

The message authentication applied by AH protects the parts of the packet that are shaded in Figure 17. Note that although the IP header is shaded in the diagram, parts of it are not authenticated because they can change in transit between sender and receiver.

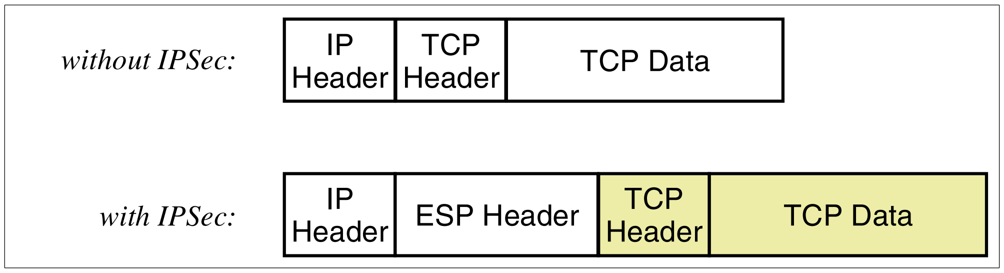


Figure 18. ESP in transport mode

With the Encapsulating Security Payload (ESP) format in transport mode, the TCP header and data are encrypted and, optionally, authenticated. But as can be seen in Figure 18 (where the protected areas are shaded), the IP header is afforded no protection at all. However, this should not be a problem because sending and receiving hosts have been authenticated and verified in the SA.

ESP in tunnel mode

A common application of VPNs is the use of a protected tunnel between two secure networks. IPSec-capable firewalls at each end of the tunnel encrypt the packets they send from the secure network through the tunnel; they decrypt the packets they receive from the tunnel and route them to the destination hosts. In this scenario, the SAs do not authenticate the destination hosts (just the firewalls) and an attacker's modification of the IP headers could go undetected.

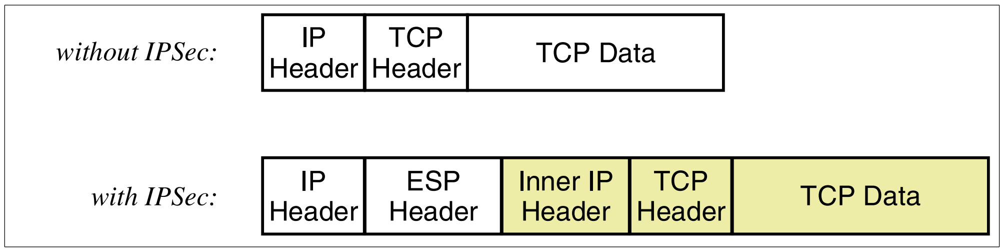


Figure 19. ESP in tunnel mode

In this environment, tunnel mode is to be used. Figure 19 shows the format of ESP packets in this mode; again, protected areas are shaded. The complete original packet, including the original IP header, is used as payload for an ESP packet. The inner IP header has the address of the destination host while the outer IP header addresses the firewall at the end of the tunnel. In this way, the complete packet including the IP header is protected.

In some cases, the AH and ESP formats are combined (applied one after the other) in order to reap both the benefits of IP header authentication with AH and payload (data) encryption with ESP.

2.3.2 Alternative VPN solutions: Layer 2 Tunneling Protocol

A remote access dial-up solution for mobile users is a very simple form of a virtual private network, typically used to support dial-in access to a corporate network whose users are all company employees. To eliminate the long-distance charges that would occur if a remote user were to dial in directly to a gateway on the home network, the IETF developed a tunneling protocol, Layer 2 Tunneling Protocol (L2TP). This protocol extends the span of a PPP connection: instead of beginning at the remote host and ending at a local ISP′s point of presence, the virtual PPP link now extends from the remote host all the way back to the corporate gateway. In effect, the remote host appears to be on the same subnet as the corporate gateway.

Since the host and the gateway share the same PPP connection, they can take advantage of PPP′s ability to transport protocols other than just IP. For example, L2TP tunnels can be used to support remote LAN access as well as remote IP access. Figure 20 outlines a basic L2TP configuration.

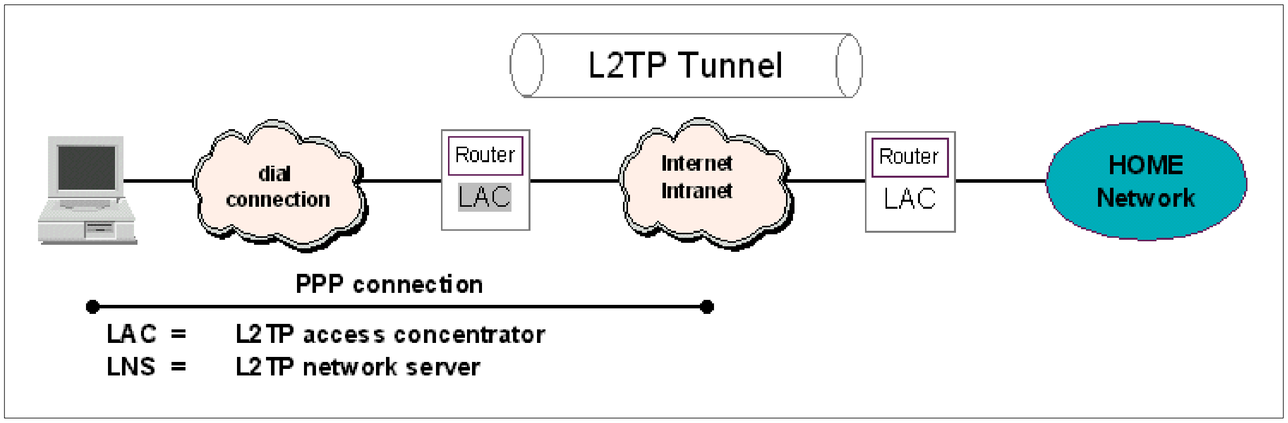


Figure 20. Layer 2 Tunneling Protocol (L2TP) scenario

Although L2TP provides cost-effective access, multiprotocol transport, and remote LAN access, it does not provide cryptographically robust security features. For example:

- Authentication is provided only for the identity of tunnel endpoints, but not for each individual packet that flows inside the tunnel. This can expose the tunnel to various attacks.
- Without per packet integrity, it is possible to mount denial-of-service attacks by generating bogus control messages that can terminate either the L2TP tunnel or the underlying PPP connection.

- L2TP itself provides no facility to encrypt user data traffic. This can lead to embarrassing exposures when data confidentiality is an issue.
- While the payload of the PPP packets can be encrypted, the PPP protocol suite does not provide mechanisms for automatic key generation or for automatic key refresh. This can lead to someone listening in on the wire, breaking that key, and gaining access to the data being transmitted.

2.4 Secure Sockets Layer

Secure Sockets Layer (SSL) is a protocol developed by the Netscape Communications Corporation that uses encryption to provide privacy and authentication between two applications in TCP/IP. SSL can be regarded as a *transport layer* equivalent of IPSec. Like IPSec, it uses asymmetric cipher algorithms (RSA is normally used) to authenticate users and sign messages, and symmetric algorithms to ensure confidentiality. Unlike IPSec, SSL is used to protect connections between application users; IPSec protects data at the network layer between IP hosts.

HTTP can use SSL to secure its communications. This allows Web browsers and servers to pass confidential or sensitive data through the Internet or intranet. SSL is also used by the Lightweight Directory Access Protocol (LDAP) for secure connections between LDAP clients and LDAP servers, by Telnet/3270, and by Host On-Demand for connections between the client and the host system. SSL-enabled applications on OS/390 comprise:

- IBM HTTP Server for OS/390
- TN3270 Server
- OS/390 LDAP Server
- OS/390 Firewall Configuration Server
- CICS Web Interface
- OS/390 UNIX policy agent
- Digital Certificate Access Server (DCAS) used in Express Logon Feature
- FTP server and client (z/OS V1R2 and later)

2.4.1 SSL overview

SSL was originally developed to protect traffic between a client and a server communicating across the Internet. The latest version of SSL from Netscape (and final version from Netscape) is SSL 3.0. At time of writing, it is by far the most commonly used SSL protocol. According to the latest SSL standard (RFC 2246, *The TLS Protocol Version 1.0*, Appendix E) SSL 2.0 should be phased out “with all due haste”. The IETF TLS-Based Telnet Security document (see 2.5, “Transport Layer Security protocol (TLS)” on page 41) goes a step further to say that SSL 2.0 is not an acceptable protocol at all. See Figure 21 for an outline of some of the SSL protocols and standards.

The use of SSL for Web access is through a protocol called HTTPS. HTTPS is a unique protocol that combines SSL and HTTP. You need to specify `https://` instead of `http://` as an anchor in HTML documents that link to SSL-protected documents. A client user can also open a URL by specifying `https://` to request SSL-protected documents.

Because HTTPS and HTTP are different protocols and use different ports (443 and 80, respectively), you can run both SSL and non-SSL requests at the same time. As a result, you can choose to provide information to all users using no security, and specific information only to browsers that make secure requests. This is how a retail company on the Internet can allow users to look through the merchandise without security, but then fill out order forms and send their credit card numbers using security.

SSL relies on digital certificates and a hierarchy of trusted authorities, as described in 2.1.6.1, “Digital certificates” on page 20, to ensure authentication of clients or servers.

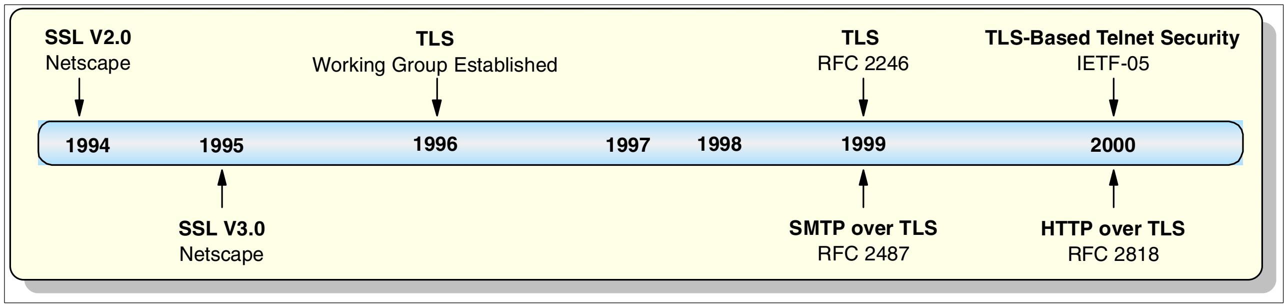


Figure 21. Evolution of SSL

2.4.2 Establishing secure communications with SSL

To use SSL, both the client and the server need to have the software to support this protocol. Because SSL started with HTTP communication, we use this as an illustration. Please see Figure 22.

The latest Netscape and Microsoft browsers support SSL 3.0 and all its features on the client. SSL is composed of two subprotocols:

- SSL Handshake Protocol
- SSL Record Protocol

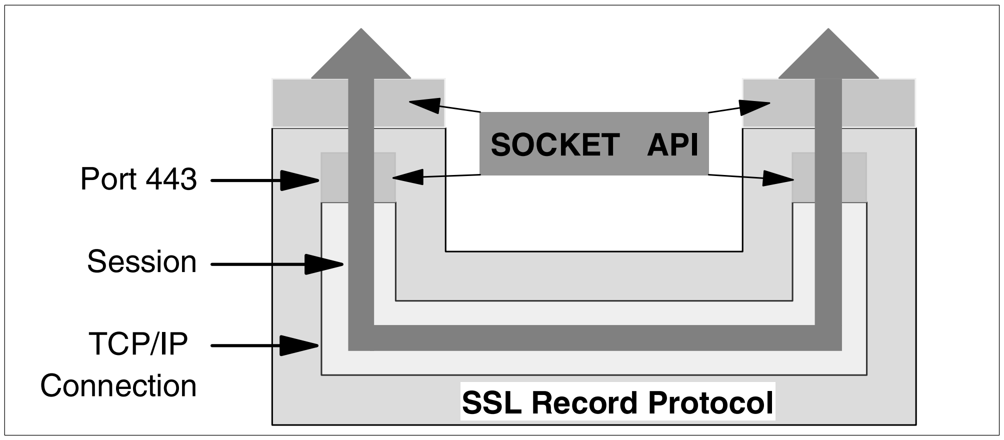


Figure 22. SSL subprotocols

The SSL Handshake Protocol initializes a secure session, with authentication of the server (and optionally, the client), agreement of an encryption scheme, and the transfer of encryption keys. A public-key algorithm, usually RSA, is used for the exchange of the symmetric encryption key and for digital signatures. With the server certificate, the client is also able to verify the server's identity. With SSL

Version 3.0, the possibility of authenticating the client identity by using client certificates in addition to server certificates was added. The overall flow of these steps is shown in Figure 23:

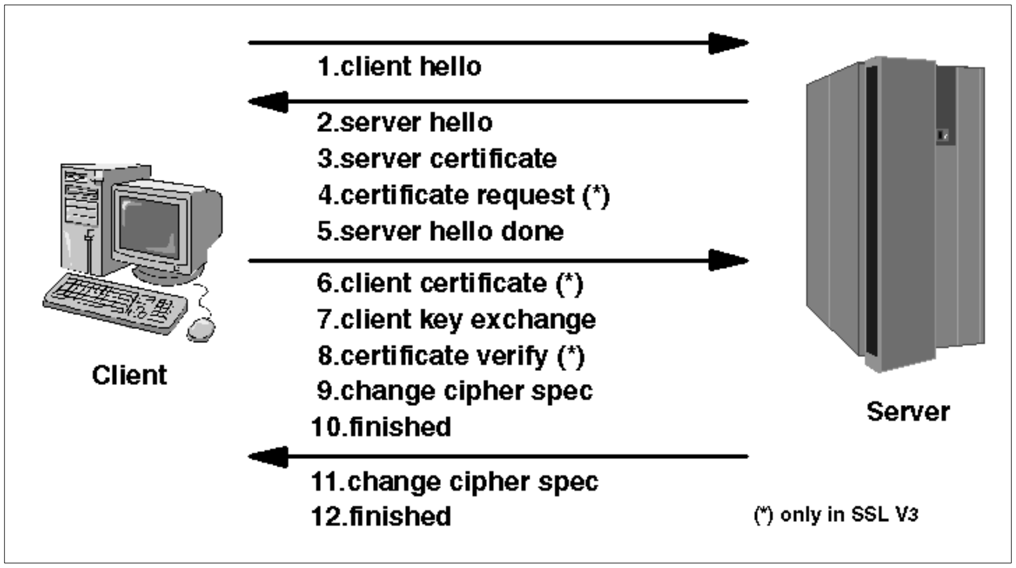


Figure 23. Overview of SSL handshake protocol

1. First, the client sends a `client hello` message which lists the cryptographic capabilities of the client (sorted in client preference order) and contains the SSL/TLS protocol version desired. It also contains a random value, a nonce. A nonce is a random value used in communication protocols, typically for replay protection.
2. The server responds with a `server hello` message, which contains the cryptographic method (cipher suite) selected by the server, the session ID, another random number, and the acceptable SSL/TLS protocol version. The client and server must support at least one common cipher suite or the handshake will fail.
3. Following the `server hello` message, the server sends its certificate. With Secure Sockets Layer, X.509 V.3 certificates are used.
4. If SSL Version 3 is used and the server application (for example, the Web server) requires a certificate for client authentication, the server sends a `certificate request` message. In the certificate request message, the server sends a list of the types of certificates supported and the distinguished names of acceptable certification authorities.
5. The server then sends a `server hello done` message and waits for a client response. Upon receipt of the `server hello done` message, the client (the Web browser) verifies the validity of the server's certificate and checks that the server hello parameters are acceptable.
6. If the server requested a client certificate, the client sends a certificate or, if no suitable certificate is available, a `no certificate` alert. This alert is only a warning, but the server application can fail the session if client authentication is mandatory.
7. The client then sends a `client key exchange` message. This message contains the so-called pre-master secret, a 46-byte random number that will be used in

the generation of the symmetric encryption keys and the Message Authentication Code (MAC) keys, encrypted with the public key of the server.

8. If the client sent a certificate to the server, the client will now send a `certificate verify` message, which is signed with the client's private key. By verifying the signature of this message, the server can explicitly verify the ownership of the client certificate.

   A similar process to verify the server certificate is not necessary. If the server does not have the private key that belongs to the certificate, it cannot decrypt the pre-master secret nor create the correct keys for the symmetric encryption algorithm, and the handshake must fail.

9. Now, the client uses a series of cryptographic operations to convert the premaster secret into a master secret, from which all key material required for encryption and message authentication is derived. Then, the client sends a `change cipher spec` message to make the server switch to the newly negotiated cipher suite.

10. The `finished` message immediately following is the first message encrypted with this cipher method and keys.

11. After the server responds with a `change cipher spec` and a `finished` message of its own, the SSL handshake is completed and encrypted application data can be sent.

The SSL Record Protocol transfers application data using the encryption algorithm and keys agreed upon during the handshake phase. As explained above, symmetric encryption algorithms are used, since they provide much better performance than asymmetric algorithms.

2.4.3 SSL considerations

As discussed, security functions such as SSL are needed to send sensitive data safely if you connect your system to an insecure network such as the Internet. On the other hand, using such security functions has performance impacts, including utilizing additional CPU cycles and degrading Web server performance.

Furthermore, SSL does not satisfy every security requirement. While it protects against eavesdropping and alteration of data, it cannot protect the server from an attacker masquerading as a trusted user. For these security concerns, the risk can be minimized by the use of access controls or firewalls.

To maintain SSL security you have to manage the key carefully, especially when using self-certification, because the whole system environment is affected by the security of the Certificate Authority's key database. On OS/390 the key database or key ring file, including the server key pair, is stored in a directory in the HFS. It may be accessible by those who use the OS/390 UNIX shell. Therefore, you must protect the key database from being viewed or accessed by unauthorized users. The file permissions in the UNIX shell is your first line of defense. However, RACF is an ideal vehicle for this protection.

2.4.4 SSL with OS/390 RACF

Even when used with SSL, user authentication with user ID and password still has a number of vulnerabilities. SSL client authentication provides protection, particularly when combined with RACF authorization checking. This is achieved by RACF class DIGTCERT. By means of this RACF class, you can register your certificate into the RACF database. In other words, you can associate your client certificate with your RACF user ID. At the time of writing, there are a number of products on OS/390 that exploit this RACF function for SSL V3 client authentication. They are:

- IBM HTTP server for OS/390 (Domino Go Webserver 5.0 or later)
- TN3270 server (OS/390 V2R8 or later)
- LDAP server (OS/390 V2R8 or later)
- Digital Certificate Access Server (DCAS) (OS/390 V2R10)
- CICS Web Support in CICS Transaction Server V1R3 for OS/390
- CICS Transaction Gateway in CICS Transaction Server V1R3 for OS/390

This function allows you to check whether a certificate a client submitted is signed by a trusted root Certificate Authority, and to check whether this certificate is associated with a valid RACF user ID.

2.4.5 System SSL

Initially, each application on OS/390 that wanted to support SSL needed to supply the SSL libraries with the product. This requirement leads to unnecessary duplication of code and also has other disadvantages:

- Different applications are using different SSL toolkit levels, which introduces functional differences and also may require the use of different certificate and key management utilities.
- SSL enabling of applications written by the installation is difficult because an SSL toolkit is usually not readily available.

These problems have been solved with *System SSL* on OS/390, a generic SSL toolkit which is generally available and part of the Cryptographic Services Base element of OS/390.

System SSL allows each application to use its own certificate and keyring material for accurate server side authentication.

System SSL enables applications on OS/390 to use both client and server SSL services. On the OS/390 V2R10 level, the following APIs are supplied:

- A C/C++ API to be used with the OS/390 Sockets API with a reliable transport such as TCP.
- A Java class library in file GSKSSL.jar that can be used to create an SSL session and to handle the input and output streams.

System SSL will use the Cryptographic Coprocessor facility on S/390 and the PCICC cards if installed and properly initialized for acceleration of cryptographic functions. The RSA decryption of the key material during the SSL handshake as well as the Triple-DES or DES encryption and decryption of data can be performed by the cryptographic hardware on S/390.

2.4.6 OS/390 key management utilities

As discussed in 2.4.1, “SSL overview” on page 36, SSL connections make use of public/private key mechanisms for authenticating each side of the SSL session and agreeing on bulk encryption keys to be used for the SSL session. To use public/private key mechanisms, public/private key pairs must be generated. In addition, X.509 certificates (which contain public keys) must be created or certificates must be requested, received, and managed.

In OS/390 Release 8, a utility called GSKKYMAN manages keys and certificates. GSKKYMAN replaced the MKKF utility that was used in previous releases. The GSKKYMAN utility creates, fills in, and manages an OS/390 HFS file that contains private keys, certificate requests, and certificates. This OS/390 HFS file is called a key database and, by convention, has a file extension of kdb.

As of OS/390 V2R8, the RACF `RACDCERT` command is also supported by SSL. `RACDCERT` installs and maintains private keys and certificates in RACF. Refer to the *OS/390 Security Server (RACF) Command Language Reference*, SC28-1919, for details on the `RACDCERT` command. For more detailed information regarding how to use the GSKKYMAN utility, see *OS/390 Cryptographic Services System Secure Sockets Layer Programming Guide and Reference*, SC24-5877.

In Chapter 5, "TCP/IP application security" on page 101, we show some examples of SSL communications with IBM HTTP Server for OS/390, TN3270 Server and OS/390 LDAP Server, as well as the GSKKYMAN utility.

2.5 Transport Layer Security protocol (TLS)

SSL 3.0 has outgrown the scope of being a Netscape standard. Continued development of the protocol fell into the hands of the Internet Engineering Task Force in 1996. The result was that SSL 3.0 evolved into the proposed standard for Transport Layer Security, RFC 2246.

TLS is the latest in the continuing evolution of SSL. TLS 1.0 might as readily have been titled SSL 3.1. In fact, when negotiating a TLS handshake, the client and server hello messages will use version specification 3.1 (SSL 3.0 uses version specification 3.0).

Note

At the time of writing, TLS (RFC 2246) is not supported on the TN3270 server. Only IETF Internet-Draft TLS-Based Telnet Security (negotiated Telnet) is supported. RFC 2246 is supported for FTP in z/OS V1R2 and newer.

So, what exactly is new in TLS? Not very much. The protocol syntax and handshake flow remain virtually unchanged. The significant difference being that the hello message for TLS must contain version 3.1. Once it has been agreed by both client and server that 3.1 is to be used, cipher suite exchanges will use a prefix of `TLS_` instead of the SSL 3.0 prefix of `SSL_`.

Finally, TLS 3.1 is a protocol designed with the intent of allowing enhancements for future improvements to privacy over TCP connections.

2.5.1 TLS-based Telnet security

The first S/390-based release to support the Internet Engineering Task Force (IETF) Internet-Draft TLS-based Telnet Security protocol is OS/390 V2R10. The support was implemented at the revision -03 level. At the time of writing, revision level -05 is current.

Negotiated Telnet is a name used (quite aptly) in the z/OS and CS for OS/390 IP manuals. In fact, negotiated Telnet is an implementation of IETF TLS-based Telnet Security. The name TLS-based Telnet Security is a little misleading because this IETF internet draft functions equally well with TLS 1.0 and SSL 3.0. In other words, the actual security protocol used with TLS-based Telnet Security could be SSL 3.0.

What does the TLS-based Telnet Security define? It adds a new IAC (Interpret As Command) option and suboption. The `START_TLS` option allows the client (`WILL START_TLS`) or the server (`DO START_TLS`) to initiate a request for a secure session. Once TLS has been agreed upon, the session immediately drops into negotiation of either SSL or TLS. Negotiation of other Telnet IAC options is suspended until the security negotiation has successfully completed.

If the client and server cannot agree upon the `START_TLS` option, then the Telnet server can opt to drop into native TLS/SSL security negotiation (`CONNTYPE SECURE` in the TCP/IP profile data set).

Why add a new IAC option? The foremost advantage is that this option places the control of session security into the TN3270 world (instead of leaving it up to the transport layer). If a Telnet client won't accept a `DO START_TLS` option, the Telnet server can choose to end the session (`CONNTYPE NEGTSECURE` in the TCP/IP profile data set).

The other significant advantage of placing encryption negotiation into the TN3270 option data stream is that a single port can be used for encrypted and non-encrypted sessions. Prior to negotiated Telnet, a separate port for secure and non-secure sessions had to be used. Since all TN3270 clients default to port 23, this was not an ideal situation.

Other advantages include being able to use session states to determine the status of a session. In addition, placing session security into the domain of TN3270 allows the new AUTH and ENCRYPT Telnet options to be used (if required). Both of these options (RFC 2941, *Telnet Authentication Option*, and RFC 2946, *Telnet Data Encryption Option*, respectively) have not been implemented in z/OS and CS for OS/390 IP. However, they do indicate that the value of negotiated Telnet is also in its future capabilities.

It should be noted that the TLS 1.0 RFC warns against SSL 2.0, saying that is should be done away with as soon as possible. However, the IETF TLS-based Telnet document goes a step further to say that SSL 2.0 is not allowed at all.

2.6 Lightweight Directory Access Protocol (LDAP)

In many organizations, a vast amount of information describing the various users, applications, files, printers, and other resources accessible from a network is often collected into a special database that is sometimes called a directory. As the number of different networks and applications has grown, the number of specialized directories of information has also grown resulting in islands of information that are difficult to share and manage. If all of this information could be maintained and accessed in a consistent and controlled manner, it would provide a focal point for integrating a distributed environment into a consistent and seamless system.

The Lightweight Directory Access Protocol (LDAP) is an open industry standard that has evolved to meet these needs. LDAP defines a standard method for accessing and updating information in a directory. LDAP is gaining wide acceptance as the directory access method of the Internet and is therefore also becoming strategic within corporate intranets. It is being supported by a growing number of software vendors and is being incorporated into a growing number of applications. For example, the two most popular Web browsers, Netscape Navigator/Communicator and Microsoft Internet Explorer, support LDAP as a base feature.

2.6.1 The OS/390 LDAP server

Much critical security information such as security policies, user passwords, and X.509 digital certificates, is maintained in LDAP directories because LDAP is an open standard protocol supported by all the major vendors. This means an LDAP server is required to provide high availability, fault tolerance, efficient resource utilization, and high performance, not to mention a high degree of security for itself. If an LDAP server is down, it could mean that nobody can log on to anything.

The OS/390 LDAP server is intended for use in an environment where high transaction volumes are common, or where maximum availability is required. It can meet the requirements outlined above by exploiting the fault tolerance and dynamic workload management abilities of a Parallel Sysplex, as shown in Figure 24.

These benefits are achieved by the concurrent running of multiple servers that are functionally equivalent and that provide access to the same LDAP data. This is called multiserver mode. Furthermore, connection optimization (provided by the Workload Manager/Domain Name System cooperation in CS for OS/390 IP) achieves workload balancing by distributing connections to the systems with the greatest amount of available resources.

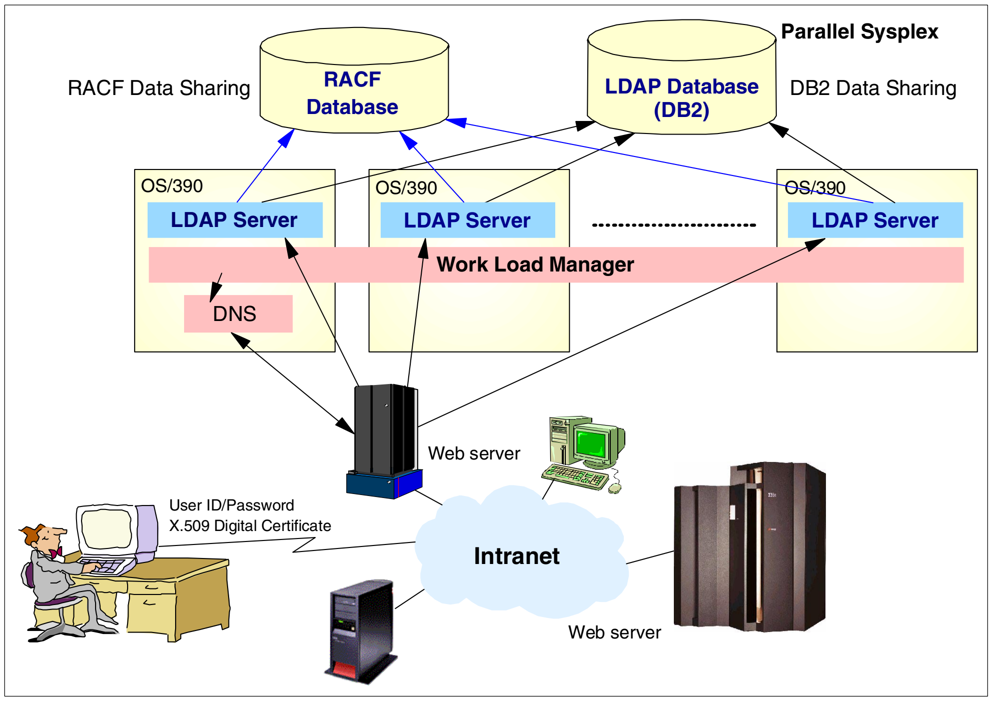


Figure 24. OS/390 LDAP server in a Parallel Sysplex

Figure 24 shows multiple OS/390 LDAP servers in a Parallel Sysplex environment. Each LDAP server has the same capability, and all LDAP servers can share their directory entries in a single DB2 database using DB2 sharing with the coupling facility. For more information about how to set up connection optimization and configure multiserver mode, see *IBM Communications Server for OS/390 IP Configuration*, GC31-8513 and *OS/390 Security Server LDAP Server Administration and Usage Guide*, SC24-5861.

2.6.1.1 Accessing RACF information with LDAP

Some information in the RACF database, especially the data contained in user and group profiles, is information that is very suitable for a directory. To avoid the need for duplication of the data, RACF makes this information available to LDAP.

Using LDAP services and attributes, the information in RACF profiles in classes USER and GROUP can be displayed and modified. It should be noted that the interface between LDAP and RACF converts all LDAP requests into RACF commands. Therefore, only such operations can be performed that are supported with RACF commands and all RACF authorization requirements for commands also apply in the RACF environment.

LDAP accesses all fields with LDAP attribute names. For instance, the user ID is called “racfid” and the user name field is called “racfProgrammerName”. The complete attribute list can be found in the configuration files slapd.at.racf and slapd.oc.racf, which by default are located in directory /usr/lpp/ldap/etc.

2.6.1.2 Authentication with the OS/390 LDAP server

In many directories, a good deal of the information can be accessed by anonymous users without a need for authentication. However, there may be other information of a more confidential nature that can only be accessed by authorized users. Also, any update of data maintained by an LDAP server will usually be only allowed for authorized users.

Checking the authority of a user can only be useful if the user has been reliably authenticated. To fulfill this requirement, the LDAP server supports a number of authentication methods:

- Authentication with username and password stored in the LDAP directory.
- Authentication with an X.509 V.3 client certificate.
- Authentication with the user's RACF user ID and password. The LDAP server will call RACF to authenticate the user's password.

The authentication of the client is performed when the LDAP client binds to the LDAP server. This is the process when the client opens a socket connection with the server.

If passwords are stored in the LDAP directory, a number of different methods are available to prevent unauthorized disclosure of passwords:

- One-way encryption using the SHA-1or MD5 message digest. These methods require the OS/390 Open Cryptographic Services Facility (OCSF) to be installed. The clear-text password cannot be retrieved.
- One-way encryption using the *crypt* method. Crypt is a method based on the DES encryption algorithm that is mostly used in UNIX systems to create one-way encrypted representations of passwords in the passwd file. The clear-text password cannot be retrieved, although there are programs available such as Crack 5.0 by Alec Muffett that can retrieve clear-text passwords in amazingly little time.
- Two-way encryption using DES. This method requires the OS/390 Open Cryptographic Services Facility (OCSF) to be installed. This is the only method that allows the clear-text password to be retrieved. If clear-text passwords need to be retrieved from the LDAP directory, this method should be used.

As an alternative to authentication with passwords, the LDAP server supports user authentication with X.509 V.3 client certificates. After verifying the authenticity of the user's certificate, the LDAP server will match the distinguished name of the certificate owner to the distinguished name of the user's directory entry. The session between the LDAP client and the LDAP server must be an SSL session, or client certificates cannot be used.

For users already defined in RACF, there is no incentive to create and store additional passwords in the LDAP directory since the users would be forced to maintain these passwords or synchronization methods would have to be developed. The LDAP server, if so configured, is able to take the user ID and password specified in the bind request and call RACF for authentication of the user. If the user ID and password supplied by the user are valid, the bind is successful.

User authentication in RACF is required for using LDAP to retrieve and administer information in RACF user and group profiles (see 2.6.1.1, "Accessing RACF information with LDAP" on page 44). If a valid RACF user identity is not established during the bind, requests for RACF information will be rejected by RACF.

2.7 Kerberos-based security system

Kerberos is a network authentication protocol that was developed in Project Athena at the Massachusetts Institute of Technology, in cooperation with IBM and Digital Equipment Corporation in 1980's. DES cryptography is used to provide data privacy, especially for the sensitive data such as password to log into a server.

Kerberos Version 5 is the latest release and has been implemented in SecureWay Security Server Network Authentication and Privacy Service for OS/390, and chosen by Microsoft Corporation as their preferred authentication technology in Windows 2000.

Note: IBM OS/390 implementation of Kerberos Version 4 provided by CS for OS/390 IP is no longer supported in z/OS V1R2.

The Kerberos system is an encryption-based security system that provides mutual authentication between the users and the servers in a network environment. The assumed goals for this system are:

- Authentication to prevent fraudulent requests and/or responses between users and servers that must be confidential and on groups of at least one user and one server.
- Authorization can be implemented independently from the authentication by each service that wants to provide its own authorization system. The authorization system can assume that the authentication of a user/client is reliable.
- Message confidentiality may also be used that provides assurance to a data sender that the message's content is protected from access by entities other than the context's named peer.

The Kerberos authentication is heavily based on the shared secrets, which are passwords stored on the Kerberos server. Those passwords are encrypted with a symmetrical cryptographic algorithm, which is DES in this case, and decrypted when needed. This fact implies that a decrypted password is accessed by the Kerberos server, which is not usually required in an authentication system that exploits public key cryptography. Therefore the servers must be placed in locked rooms that are physically secure to prevent an attacker from stealing a password.

For the complete description about the Kerberos Version 5 protocol, refer to *RFC 1510 - The Kerberos Network Authentication Service (V5)*.

2.7.1 Kerberos protocol overview

The Kerberos system consists of three components: a client, a server, and a trusted third party, which is also known as a Key Distribution Center (KDC). KDC interacts with both a client and a server to accept the client's request, authenticate its identity, and issue tickets to it.

The domain served by a single KDC is referred to as a *realm*. A *principal identifier* is used to identify each client and server in a realm. The principal name is uniquely assigned for all clients and servers by the Kerberos administrator. All principals must be known to the KDC.

Although the Kerberos protocol consists of several subprotocols, three exchanges would be the most interesting for almost readers. See Figure 25. The first phase exchange is taken place between a client and the authentication server (AS), in which a client asks the AS that knows secret keys of all client in the realm to authenticate himself and give it a ticket called *ticket granting ticket* (TGT) to be used to get a secret shared with an application server it wants to access.

Upon receiving TGT, the client sends a request, which contains the TGT, for a service ticket to the ticket-granting server (TGS), and wait for a service ticket returned. Having the session ticket ready, the client is allowed to communicate with the server that is providing a service he wants to use. Optionally the application server can perform further authentication process against the client.

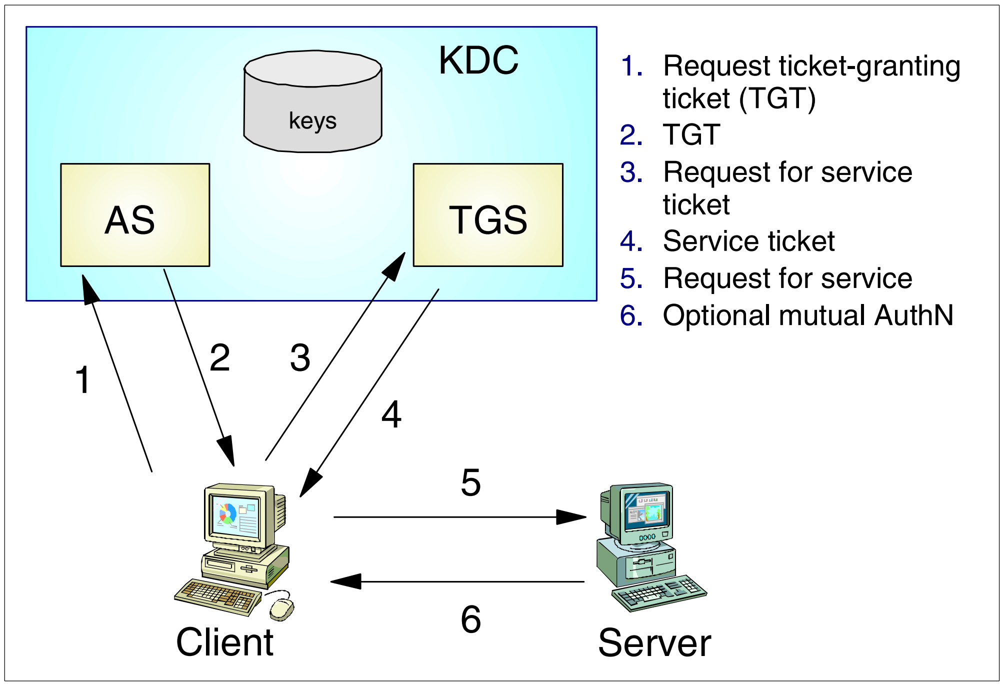


Figure 25. Kerberos protocol overview

Message encoding defined in Kerberos Version 5 is described using the(Abstract Syntax Notation 1 (ASN.1) syntax in accordance with ISO standards 8824 and 8825.

In the following sections, we will discuss the interactions in more detail using the following notations:

- K_x: X’s symmetric encryption key
- $K_{x,y}$: Encryption key shared by X and Y (for example, a session key)
- K_x\{data\}: A message that contains data encrypted with X’s key

2.7.1.1 Phase 1: Authentication service (AS) exchange

The authentication service exchange is initiated by a client when it wants to get authentication credentials for an application server but it currently holds no credentials. Two messages are exchanged between the client and the Kerberos authentication server; then credentials for a ticket-granting server (TAS) are given to the client, which is called the *ticket-granting ticket* (TGT) and will subsequently be used to obtain credentials for other services.

This exchange is used for other services, such as the password-changing service, as well. As noted in Figure 26, the client's secret key is used exclusively in this phase.

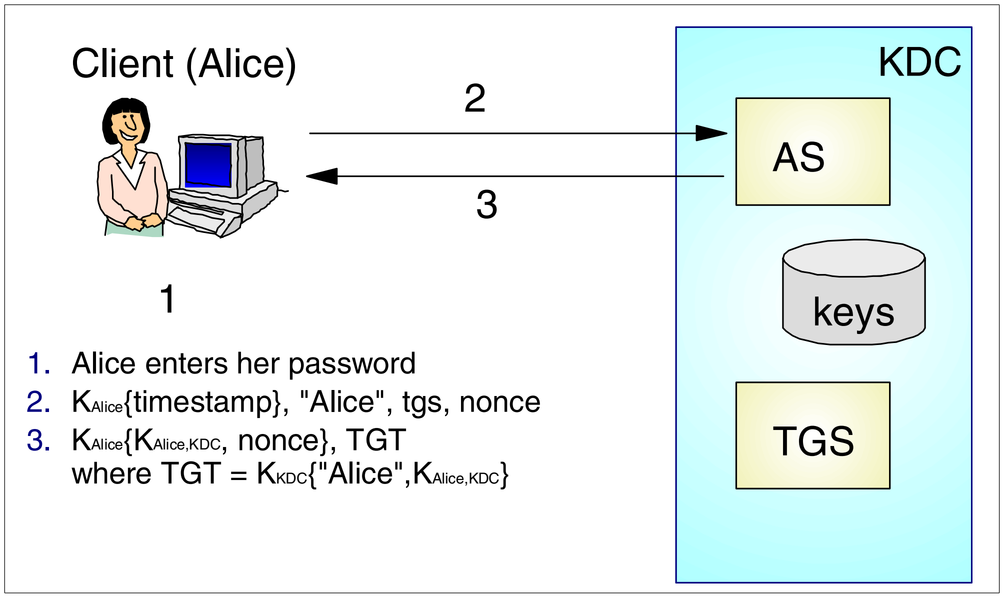

Client (Alice)
KDC
2
AS
3
keys
1
TGS
1. Alice enters her password
2. K_{Alice}{timestamp}, "Alice", tgs, nonce
3. K_{Alice}{$K_{Alice,KDC}$, nonce}, TGT
where TGT = K_{KDC}{"Alice",$K_{Alice,KDC}$}

Figure 26. Simplified authentication service exchange

When a user logs into a client system and enters her password, a client sends the Kerberos authentication server (AS) a message that includes a user name in plain text ("Alice"), the current time encrypted with her secret key, and the identity of the server for which the client is requesting credentials (TGS in Figure 26).

Upon receiving the request from the client, the AS looks up the client name and the service name (the TGS in this case) in the Kerberos database, and then obtains an encryption key of each of them, K_{Alice} and K_{KDC}.

The AS then generates a response back to the client, which contains the TGT and a session key $K_{Alice,KDC}$, which is used in the subsequent secure communication between the client and KDC. The TGT includes the session key $K_{Alice,KDC}$, the identities of the server and the client, lifetime, and some other information. The AS then encrypts the ticket using its own key K_{KDC}. This produces a *sealed ticket*. The session key $K_{Alice,KDC}$ is also encrypted using the client's key K_{Alice} with some other information, such as nonce.

The encrypted current time is also known as the *authenticator*, since the receiver can assure that the sender knows the correct shared secret K_{Alice}, which is the client's encryption key derived from her password (this key is also referred as Alice's *long-term key*), by decrypting it and validating what is inside. Because the AS knows the Alice's secret key, it can evaluate the time decrypted from the received authenticator. As you might have noticed, the clocks on the client system and the KDC must be reasonably synchronized with each other. A network time service may be used for this purpose.

An authenticator is also used to help server detect the message replays.

A *nonce* is information to identify a pair of Kerberos requests and responses. A timestamp or a random number generated by a client may be used.

TGS is the server's identification, which is the Kerberos ticket-granting server (TGS) in this case.

Since K_{Alice} is known exclusively by Alice and KDC, no one but Alice can extract the critical information from the response message, such as the session key $K_{Alice,KDC}$ to be used in the next phase.

When the client receives the AS's response, it decrypts it using its secret key K_{Alice} and checks to see if the nonce matches the specific request. If the nonce matches, the client caches the session key $K_{Alice,KDC}$ for future communications with the TGS.

2.7.1.2 Phase 2: Ticket-granting service (TGS) exchange

The next phase is used for a client to obtain credentials for services that it wants to use. This exchange is also initiated by the client, and two messages are exchanged between the client and the ticket-granting server (TGS). The protocol and message format used in this exchange is almost identical to those for the AS exchange. The primary difference is that the client's key is never used in this exchange, but the session key obtained from the preceding AS exchange is used.

The request message the client sends to the TGS contains several pieces of information including:

- Information to authenticate the client, which includes a new authenticator and the TGT obtained the preceding AS exchange
- Identity of the service for which the client is requesting credentials
- Nonce to identify this request

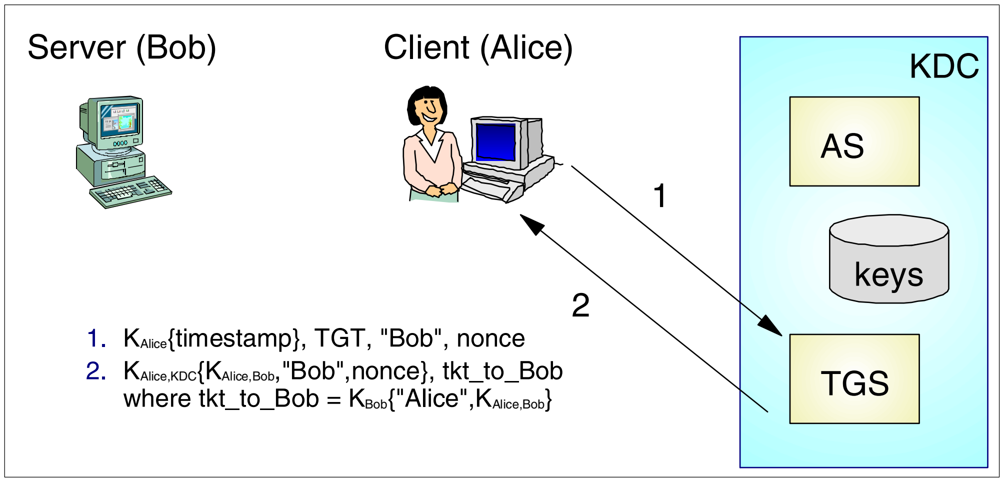

Server (Bob)
Client (Alice)
KDC
AS
keys
TGS
1
2
1. K_{Alice}\{timestamp\}, TGT, "Bob", nonce
2. $K_{Alice,KDC}$\{$K_{Alice,Bob}$,"Bob",nonce\}, tkt_to_Bob
where tkt_to_Bob = K_{Bob}\{"Alice",$K_{Alice,Bob}$\}

Figure 27. Simplified ticket-granting service exchange

When the ticket-granting server (TGS) receives the above message from the client, it first deciphers the sealed ticket using its encryption key K_{KDC}. From the deciphered ticket, the TGS obtains the session-key $K_{Alice,KDC}$. It uses this session key to decipher the authenticator. The validity checks performed by the TGS include:

- If the client name and its realm in the ticket match the same fields in the authenticator.
- If the address from which this message is originated is found in the address field in the ticket, which specifies addresses from which the ticket can be used.
- If the user-supplied checksum in the authenticator matches the contents of the request. This procedure guarantees the integrity of the message.

Finally, it checks the current time in the authenticator to make certain the message is recent. Again, this requires that all the clients and servers maintain their clocks within some prescribed tolerance.

Note: By checking the timestamp in the nanoseconds scale, the replay attacks can be detected.

The TGS now looks up the server name from the message in the Kerberos database, and obtains the encryption key K_{Bob} for the specified service.

The TGS forms a new random session key $K_{Alice,Bob}$ for the benefit of the client (Alice) and the server (Bob), and then creates a new ticket tkt_to_Bob containing:

- The session key $K_{Alice,Bob}$
- Identities of the service and the client
- Lifetime

  Note: The format of the ticket for a particular service is identical to one of the ticket-granting ticket (TGT).

It then assembles and sends a message to the client.

2.7.1.3 Phase 3: The client/server authentication (CS) exchange

The client/server authentication (CS) exchange is performed by the client and the server to authenticate each other. The client must have gotten credentials for the server using the AS or TGS exchange before the CS exchange is initiated.

After receiving the TSG exchange response from the TGS, the client deciphers it using the TGS session key $K_{Alice,KDC}$ that is exclusively known by the client and the TGS. From this message it extracts a new session key $K_{Alice,Bob}$ that is shared with the server (Bob) and the client (Alice). The sealed ticket included in the response from the TGS cannot be deciphered by the client because it is enciphered using the server's secret key K_{Bob}.

Then the client builds an authenticator and seals it using the new session key $K_{Alice,Bob}$. At last, it sends a message containing the sealed ticket and the authenticator to the server (Bob) to request its service.

When the server (Bob) receives this message, it first deciphers the sealed ticket using its encryption key K_{Bob}, which is kept in secret between Bob and the KDC. It then uses the new session key $K_{Alice,Bob}$ contained in the ticket to validate the authenticator in the same way as the TGS does in the TGS exchange.

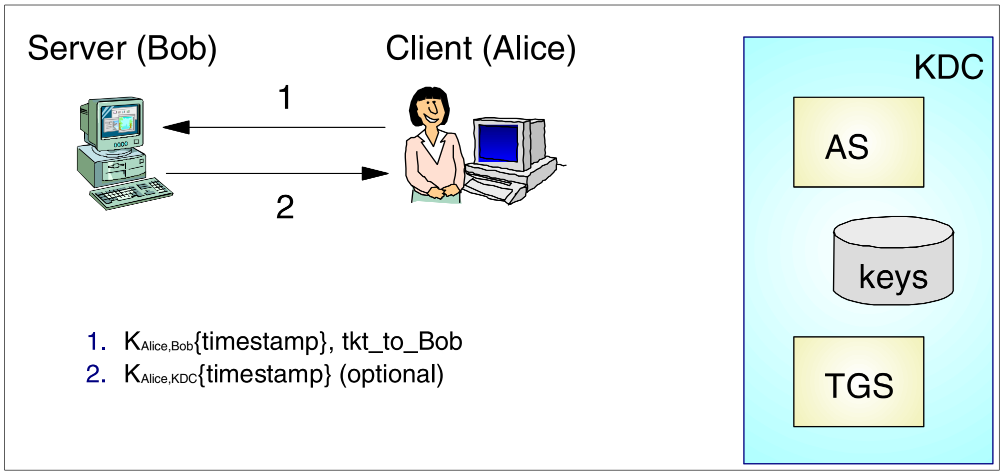

Server (Bob)
Client (Alice)
1
2
KDC
AS
keys
TGS
1. $K_{Alice,Bob}${timestamp}, tkt_to_Bob
2. $K_{Alice,KDC}${timestamp} (optional)

Figure 28. Simplified client/server authentication exchange

Once the server has authenticated a client, an option exists for the client to validate the server (this procedure is called *mutual authentication*). This prevents an intruder from impersonating the server.

If mutual authentication is required by the client, the server has to send a response message back to the client. The message has to contain the same timestamp value as one in the client's request message. This message is enciphered using the session key $K_{Alice,Bob}$ that was passed from the client to the server.

If the response is returned, the client decrypts it using the session key $K_{Alice,Bob}$ and verifies that the timestamp value matches one in the authenticator that was sent by the client in the preceding CS exchange. If it matches, then the client is assured that the server is genuine.

Once the CS exchange has completed successfully, an encryption key is shared by the client and server and can be used for the on-going application protocol to provide the data confidentiality.

2.7.2 Inter-realm operation

The Kerberos protocol is designed to operate across organizational boundaries. Each organization wishing to run a Kerberos server establishes its own realm. The name of the realm in which a client is registered is part of the client's name and can be used by the application server to decide whether to honor a request.

By establishing inter-realm keys, the administrators of two realms can allow a client authenticated in one realm to use its credentials in the other realm. The exchange of inter-realm keys registers the ticket-granting service of each realm as a principal in the other realm. A client is then able to obtain a ticket-granting ticket for the remote realm's ticket-granting service from its local ticket-granting service. Tickets issued to a service in the remote realm indicate that the client was authenticated from another realm.

This method can be repeated to authenticate throughout an organization across multiple realms. To build a valid authentication path to a distant realm, the local realm must share an inter-realm key with the target realm or with an intermediate realm that communicates with either the target realm or with another intermediate realm.

Realms are typically organized hierarchically. Each realm shares a key with its parent and a different key with each child. If an inter-realm key is not directly shared by two realms, the hierarchical organization allows an authentication path to be easily constructed. If a hierarchical organization is not used, it may be necessary to consult some database in order to construct an authentication path between realms.

Although realms are typically hierarchical, intermediate realms may be bypassed to achieve cross-realm authentication through alternate authentication paths. It is important for the end-service to know which realms were transited when deciding how much faith to place in the authentication process. To facilitate this decision, a field in each ticket contains the names of the realms that were involved in authenticating the client.

2.7.3 Some assumptions

The following limitations are applied the Kerberized security environment:

- *Denial-of-service (DoS)* attacks are not addressed by Kerberos. There are places in these protocols where an intruder can prevent an application from participating in the proper authentication steps. Detection and solution of such attacks (some of which can appear to be “usual” failure modes for the system) is usually best left to human administrators and users.

  Note: OS/390 (and z/OS also) has built-in ability to limit the capability of DoS attacks. In addition, some services provided by z/OS CS/390, such as TRMD, may be used to protect against attacks like resource hogging.
- The secret key must be kept in secret by each principal (each client and server). If an attacker steals a principal’s key, it can then masquerade as that principal or impersonate any server of the legitimate principal
- Kerberos does not address *password-guessing* attacks. If a poor password is chosen, an attacker may be able to mount an offline dictionary attack by repeatedly attempting to decrypt messages that are encrypted with a key derived from the user’s password.
- Kerberos assumes a loosely synchronized clock in the whole system. Workstations may be required to have a synchronization tool such as the time server provided.
- Principal identifiers should not be reused on a short-term basis. Access control lists (ACLs) may be used to grant permissions to particular principals.

2.7.4 Kerberos implementation in z/OS

The Kerberos Version 5 server has been introduced in OS/390 V2R10 and implemented in SecureWay Security Server Network Authentication and Privacy Service for OS/390.

Note: The Kerberos server shipped with CS for OS/390 IP supports Kerberos Version 4 and is discontinued in z/OS V1R2.

z/OS V1R2 Communications Server introduces Kerberos support to provide stronger authentication and data confidentiality for the following applications:

- The UNIX Telnet server - authentication support provided by the Kerberos 5 protocol
- The UNIX remote shell execution (rsh) server - authentication support provided by the Kerberos 5 protocol and the GSSAPI protocol
- The FTP client and FTP server - authentication support provided by the GSSAPI protocol

For further information about the Kerberos server in z/OS, refer to *SecureWay Security Server Network Authentication and Privacy Service Administration*, SC24-5896 and *SecureWay Security Server Network Authentication and Privacy Service Programming*, SC24-5897.

2.8 The OS/390 UNIX System Services policy agent

OS/390 SecureWay Communications Server Release 7 introduced a policy agent that determines and applies the level of service to be given to various traffic types. Much of its work is concerned with prioritizing IP traffic and managing resource reservations, but one of its functions is to apply access control. In this respect, it can act in a similar fashion to a firewall. Policies can block unwanted datagrams or control access for connection requests from clients.

For more information on Policy Agent, see 5.10.3, “Policy agent and security” on page 210.

2.8.1 Traffic Regulation Management Daemon (TRMD)

Traffic Regulation Management (TRM) is a function introduced in OS/390 V2R10 as a first step towards an Intrusion Detection Services (IDS) in OS/390. Its purpose is to regulate TCP/IP traffic in a way that a user cannot monopolize the available traffic bandwidth. This makes TRM an excellent tool to limit and defeat Denial-of-Service attacks attempted through flooding.

For more information on TRMD, see 5.11, “Traffic Regulation Management (TRM) security” on page 224.

Chapter 3. UNIX System Services security

This chapter discusses questions related to the security of OS/390 UNIX System Services. For readers familiar with UNIX, we give an overview of RACF and of the differences between UNIX System Services security and traditional UNIX security. We then describe how UNIX System Services can be configured to provide the level of platform security required by your installation.

3.1 OS/390 Security Server (RACF)

The standard access control system used on OS/390 systems is the OS/390 Security Server, particularly the Resource Access Control Facility (RACF) element of the OS/390 Security Server. For convenience, we refer to the product as RACF throughout this book.

For a software access control mechanism to work effectively, it must be able to:

- Identify the person who is trying to gain access to the system
- Authenticate the user by verifying that the user is really that person
- Log and report attempts of unauthorized access to protected resources
- Control the access to resources
- Allow applications to use the RACF interfaces

3.1.1 Identification and authentication

RACF uses a user ID to identify the person who is trying to gain access to the system and a password to authenticate that identity. RACF uses the concept of only one person knowing a particular user ID and password combination to verify user identities and to ensure personal accountability. So, RACF determines:

- If the user is defined to RACF
- If the user has supplied a valid password, pass ticket, or operator identification card (OIDCARD), and a valid group name
- If the user's ID (UID) and group ID (GID) are valid on OS/390 UNIX System Services (UID and GID are explained in 3.2.1, “Traditional UNIX security mechanisms” on page 59)
- If the user ID is in REVOKE status, which prevents a RACF-defined user from entering the system at all or entering the system with certain groups (if the user’s group connection is revoked)
- If the user can use the system on this day of the week and at this time of day (an installation can impose restrictions)
- If the user is authorized to access the terminal (which can also include day and time restrictions for accessing that terminal)
- If the user is authorized to access the application

After authenticating the user's identity, RACF specifies the scope of the user's authorization for the current terminal session or batch job.

3.1.2 Alternatives to passwords

In an OS/390 environment, RACF allows alternatives to passwords to be used for authenticating users. These alternatives can be pass tickets or operator ID cards

(OIDCARDs). OIDCARDs are an obsolete technology that is no longer used. In an OS/390 UNIX System Services environment where users are also identified with numeric user identifiers (UIDs) and group identifiers (GIDs), these too may be used by RACF to control user access to system resources. Unlike user names or group names, these numeric IDs can be shared by more than one user or group, although sharing is not recommended.

In a client/server environment, RACF has the ability to map a client’s digital certificate to a RACF user ID, the digital certificate being stored in the RACF database or mapped by a certificate name filter rule. A digital certificate or digital ID, issued by a certification authority, contains information that uniquely identifies the client.

The IBM HTTP Server for OS/390 (formerly known as the Lotus Domino Go Webserver), for example, authenticates a client using the client's certificate over an SSL-secured session (the so-called SSL client authentication). The HTTP Server passes the client's digital certificate to OS/390 UNIX System Services for validation. UNIX System Services passes the certificate to RACF to retrieve the RACF user ID from a mapping profile in the RACF database. This means that the RACF user ID and password of each client do not need to be supplied when accessing secure Web pages.

3.1.3 Checking authorization

After identifying and authenticating the user, RACF controls the interaction between the user and the resources in the system. RACF is called by resource managers to authorize a user’s access to a resource. This includes the access level, for instance READ (for a user to read, display or copy a resource) or UPDATE (for a user to write, update or rewrite a resource). Access levels are hierarchical in RACF, so UDPATE includes READ and so on.

Figure 29 shows how RACF uses the RACROUTE REQUEST=AUTH call to check authorization. The resource managers issue the other RACF calls in a similar manner.

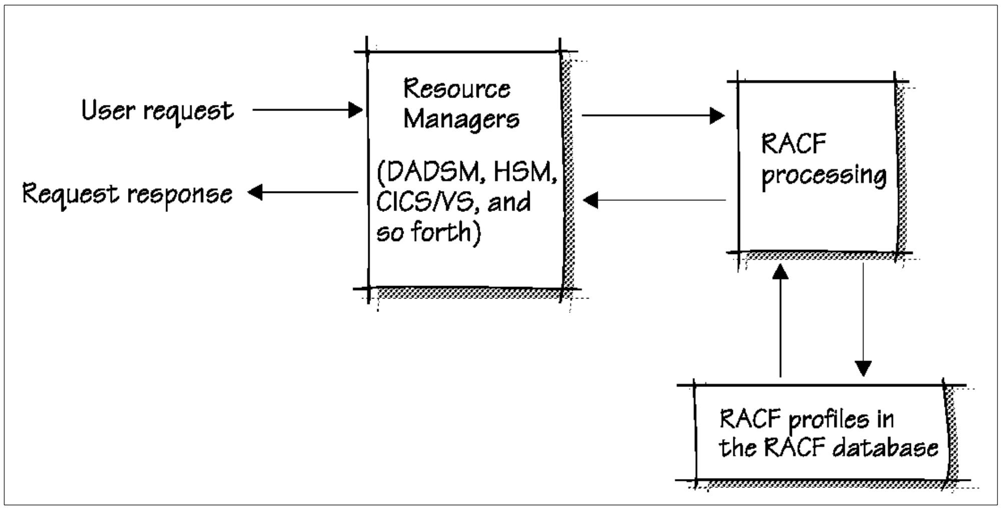

User request
Request response
Resource Managers (DADSM, HSM, CICS/VS, and so forth)
RACF processing
RACF profiles in the RACF database

Figure 29. Example of RACROUTE REQUEST=AUTH processing

Before a user can access a resource, RACF does the following:

1. Checks the profiles to determine whether the user is authorized to access the resource. OS/390 data sets are protected by profiles RACF class DATASET, but HFS files are protected differently. In the OS/390 environment, an HFS itself is an OS/390 data set protected a DATASET profile which controls access to the data set. However, individual files and directories within the HFS are protected by means of permission bits as described in 3.3.4, “Access permission to HFS files and directories” on page 67.
2. Checks the security classification of the user and data.
3. Gives the user access to the resource if any of the following conditions is satisfied:

- The resource is a data set and the high-level qualifier is the user’s user ID.
- The user ID is in the access list with sufficient authority.
- Any of the groups the user is connected to is in the access list with sufficient authority.
- The universal access authority (UACC) or the access granted to ID(*) is sufficiently high.

3.1.4 Logging and reporting

RACF maintains statistical information, such as the date, time, and number of times that a user enters a system and the number of times a specific resource was accessed by any one user. Depending on the auditing options defined, RACF also writes security log records when it detects:

- Unauthorized attempts to enter the system
- Authorized or unauthorized attempts to access RACF-protected resources
- Authorized or unauthorized attempts to enter RACF commands

RACF writes all log records to System Management Facility (SMF), a central recording facility in OS/390, as SMF type 80 records. SMF records all its data into data sets it uses in a round robin fashion. Usually, installations have automated or semi-automated procedures to save and clear the SMF data sets. In the OS/390 UNIX System Services environment, a large number of events are available that can be logged by RACF. Among them are events such as “check access to directory”, “check access to file”, “chmod”, “setuid”,”seteuid”, “mount file system”, and many more.

3.1.5 RACF and OS/390 UNIX System Services

UNIX System Services security functions are implemented in RACF, partially as extensions to existing RACF functions and partially as new RACF functions. The security functions provided include user authentication, file access checking, and privileged user checking.

UNIX System Services users are defined with RACF commands. When a job starts or a user logs on, the user ID and password are verified by existing OS/390 and RACF functions. When an address space requests an OS/390 UNIX function for the first time, RACF:

- Verifies that the user is defined as an OS/390 UNIX user.
- Verifies that the user's current connect group is defined as an OS/390 UNIX group.
- Initializes the control blocks needed for subsequent security checks.

File access, including access to load modules for execution, is controlled based on the user's UNIX identity and the permissions associated with the files in question.

3.2 Security in UNIX systems

UNIX and OS/390 UNIX System Security systems manage user identities differently. Table 2 contrasts some aspects on how user names and identities are handled by UNIX systems and OS/390 UNIX.

Table 2. UNIX security comparison

CATEGORY	UNIX	OS/390 MVS	OS/390 UNIX
User identity	Users are assigned a unique UID: 4-byte integer and user name	Users are assigned a unique user ID: 1 to 8 characters	Users are assigned a unique user ID with an associated UID
Security identity	UID	user ID	UID for accessing traditional UNIX resources and the user ID for accessing traditional OS/390 resources
Login ID	Name used to locate a UID	Same as the user ID	Same as the user ID
Special user	Multiple user IDs can be assigned a UID of 0	RACF administrator assigns necessary authority to users	Multiple user IDs can be assigned a UID of 0 or users can be permitted to BPX.SUPERUSER
Data set access	Superuser can access all files	All data sets controlled by RACF profiles	Superuser can access all HFS files; data sets controlled by RACF profiles
Identity change from superuser to regular user	Superuser can use the `su` command to switch into any UID	APF-authorized program can invoke SAF service to change identity	1.If FACILITY class profile BPX.DAEMON is not defined, the superuser can `su` into any other UID 2. If BPX.DAEMON is defined, the superuser must know the password of the other user ID or have access to SURROGAT class profile BPX.SRV.userid

CATEGORY	UNIX	OS/390 MVS	OS/390 UNIX
Identity change from regular user to superuser	The `su` shell command allows change if user provides root's password	No provision for unauthorized user to change identity	The `su` shell command allows change if the user is permitted to the BPX.SUPERUSER FACILITY class profile or if the user provides the password of a user with a UID(0)
Identity change from regular user to another regular user	The `su` shell command allows change if user provides password	No provision for unauthorized user to change identity	The `su` shell command allows change if user provides password
Terminate user processes	Superuser can kill any process	MVS operator can cancel any address space.	Superuser can kill any process, UNIXPRIV class profile can authorize non-UID(0) users
Multiple logins	Users can log in to a single user ID multiple times	Users can only log on to TSO/E once per user ID	Users can rlogin or Telnet multiple times to a single user ID in the UNIX shell, and log on once to TSO/E at the same time
Login daemons	inetd, rlogind, lm, and telnetd process user requests for login; a process is created with the user identity (UID)	TCAS and VTAM process user requests for logon; a TSO/E address space (process) is created with the user identity (user ID)	Users can log on to TSO/E or log in using one of the login daemons. In all cases, an address space is created with both an MVS identity (user ID) and a UID

3.2.1 Traditional UNIX security mechanisms

UNIX is not just one operating system, there are many different “flavors” of UNIX depending on the vendor, such as AIX by IBM, HP-UX by Hewlett-Packard, Solaris by Sun. Also, some UNIX operating systems are released by not-for-profit groups such as Linux and FreeBSD. Therefore, we have to concentrate on the common characteristics of UNIX systems.

Security in traditional UNIX is largely implemented by means of the UID and GID and the use of permission bits. Additional security can be provided by means of access control lists (ACLs).

3.2.1.1 User ID and group ID

In the UNIX architecture each user who has access to a system has a user name and an identification called a UID. The UID is a numeric value, typically in the range of 0 to 2**(31)-1. Each UID can belong to one or more groups.

Each group is identified by its name and group ID (GID) a numeric value just like the UID. In some UNIX systems, a user can only use one group at a time, and can switch to another group, provided that the user is also listed as a member of this group. Other UNIX systems such as AIX allow a user to be connected to many groups at the same time, somewhat similar to the “list of groups” function in RACF.

Each object (file or directory) in the file system has nine permission bits. The nine permission bits are divided into three sets of three bits each:

- The first set of three bits shows the owning UID’s permission.
- The next set of three bits shows the permission of the other users in the group that owns the file or directory.
- The last set of three bits shows the permission of anyone else with access to the file.

The three bits in each set indicate, respectively, read, write, and execute permission to the file. Read permission to a directory lets you list the files and subdirectories it contains. Write permission to a directory allows you to create, update or delete an object in that directory. Execute permission to a directory lets you search a directory for a specified file.

To be able to access a file, not only the appropriate permission to the file but also permission to search the chain of directories from the root directory down to the file is required.

3.2.1.2 Access Control Lists (ACLs)

ACLs are a facility offered by some versions of UNIX, including AIX. The need for the functionality they provide is as follows.

The POSIX permission bits allow to give a specific set of access permissions to only one group. However, there are cases when, for instance, one group needs read access to a file or directory while another group needs write access.

This is a problem that access control lists (ACLs) can solve. An ACL is a list of permissions attached to an object (file or directory). These permissions permit or deny access to the object. They can specify user IDs or group IDs, and allow or deny them separately read, write or execute rights. The degree of control and flexibility offered by ACLs allows system administrators to answer complicated user requirements where some users fulfill different roles and participate in different workgroups.

An object can have at most one ACL attached to it. If there is no ACL, its permission bits are used to determine its access attributes.

Conversely, the same ACL can be duplicated and applied to several files, to which it will grant identical permissions.

3.2.1.3 Auditing

Several versions of UNIX, including AIX, have full auditing. The system administrator can specify which objects (files or directories) are to be audited, and what kind of access should be audited. Audit log files are produced during the operation of the system.

Ideally, these logs should be reviewed periodically. Unfortunately, there is no standard way of filtering and querying audit log files under UNIX. For this task, most system operators rely on locally written scripts in AWK, Perl or other text-processing languages.

3.3 OS/390 UNIX System Services security

The security mechanisms in OS/390 are integrated with the system. Each component that needs a service from the security product calls it with a standard interface called RACROUTE. The RACF component of the OS/390 Security Server provides the needed services. OS/390 UNIX System Services use the RACROUTE callable services to interface with RACF and receive the needed security services.

Note

Alternative security products are available which are not the focus of this book. This book exclusively refers to RACF, and statements made about OS/390 UNIX System Services security only take RACF into account.

In OS/390, each user who may access the system has an identification called a user ID. For each user ID, a user profile is defined in class USER in the RACF database which contains all information about this user. The data elements for a user that are needed in the OS/390 UNIX environment such as UID or home directory are defined in the *OMVS segment* of the user profile.

For organizational purposes and for ease of administration, users can be connected to groups. Groups are identified by a *group name* and are collections of users. A user who is connected to a group inherits the access rights that have been given to the group. Users can be connected to many groups and can make use of all of their groups' privileges at the same time. This property in RACF is called *list-of-groups checking*.

In UNIX systems, a superuser has the authority to access any files and to do all administration of users and resources. In a system like that, there is no superuser accountability and allowing a superuser to assume any other user identity does not add any security risk to the system.

In an OS/390 system, there is a system of checks and balances between security administrators (users with the RACF attribute SPECIAL), auditors (users with the RACF attribute AUDITOR) and system programmers. In this type of system, allowing a superuser to switch into the identity of any OS/390 UNIX user could be a security risk.

For this environment, additional security functions have been implemented in OS/390 UNIX System Services to ensure that overall system security is not negatively impacted by UNIX functions. These additional security functions are optional, and therefore OS/390 UNIX System Services supports two levels of system security:

- UNIX level (the traditional method)
- OS/390 UNIX level (with added security functions for OS/390)

Both levels are distinguished by the existence of at least one of the following two profiles in class FACILITY: BPX.DAEMON and BPX.SERVER.

3.3.1 UNIX level security

If the BPX.DAEMON and BPX.SERVER profiles in class FACILITY class are not defined, the system has UNIX-level security.

This level of security is for installations where superuser authority has been granted to system programmers and security administrators. These individuals already have permission to access critical data sets such as PARMLIB, PROCLIB, and LINKLIB and they can exert total authority over the system.

Daemon programs run with superuser authority and can issue setuid(), seteuid(), and __spawn() to change the UID of any process to a different UID.

At the UNIX level of security, administrators need to be superusers and daemon programs such as inetd or cron must run with UID(0).

Note

We do not regard UNIX level security to be adequate for an OS/390 UNIX System Services environment and do not recommend to run any system at this level of security. Consequently, the rest of this chapter assumes a system running with OS/390 UNIX System Services security.

3.3.2 OS/390 UNIX System Services level security

A system has OS/390 UNIX System Services level security when at least one of the FACILITY class profiles BPX.DAEMON or BPX.SERVER is defined. At this level of security, additional security functions are active to increase the overall security of the system, especially better control of identity changes and improved superuser control.

This level of security is for customers with stricter security requirements who need to have some superusers maintaining the file system but want to have greater control over the OS/390 resources that these users can access. Although BPX.DAEMON provides some additional control over the capabilities of a superuser, a superuser should still be regarded as a privileged user because of the large range of privileges the superuser is granted.

In a traditional UNIX environment, a user is associated with a UID. If the UID of a process is changed by a service such as setuid(), the identity of the user associated with the process also changes. This is not necessarily true in the OS/390 UNIX System Services environment, where a user is identified by a RACF user ID as well as an OS/390 UNIX UID. When a program or user attempts to alter the user identity of a process, the result will vary depending on what authority is permitted. The following possibilities exist:

- The effective UID of a process is changed to UID(0) but the RACF user ID remains unchanged. This happens in the following cases:
  - When a shell user enters the `su` command without specifying a user ID in order to enter superuser mode.
  - When a TSO ISHELL user uses the **Enter Superuser Mode** menu option.

  - When a program such as SMP/E issues seteuid() to enter superuser mode.

    Note: The RACF user ID associated with the process needs READ access permission to FACILITY class profile BPX.SUPERUSER in all cases above.

- A program or daemon first issues the __passwd() service to authenticate the RACF user ID to be switched to and then issues setuid() or __spawn() with target user ID to change the user ID and UID of the process or the child process, respectively. The caller of __passwd() needs to supply the user ID and the password or PassTicket of the user. If the password is not supplied, the caller needs READ access to profile BPX.SRV.userid in class SURROGAT. Also, the address space of this process needs to be program-controlled (more on this later). Examples for this usage are:
  - A user enters the su command with the userid parameter. The user must enter the password for the user ID or, with appropriate authority in class SURROGAT, the user can use the "-s" switch or press Enter at the password prompt.
  - The FTP server, before changing the identity of the child process to the user ID and UID of the client, authenticates the user with user ID and password. For anonymous FTP, the FTP server needs access to profile BPX.SRV.anonymo in class SURROGAT, where user ID `anonymo` is the default RACF user ID for an anonymous FTP user.
- A daemon issues the setuid() or spawn() with user ID change service and just specifies the UID to be switched to without first authenticating the user. This is typically done by the cron daemon. The user ID of the process doing this needs to have *daemon authority* to be successful.
- A server program issues the service pthread_security_np() to create a thread running under this user ID and UID. The issuer of pthread_security_np() needs access to profile BPX.SERVER in class FACILITY. Also, the address space of this process needs to be program-controlled.

Daemon authority

BPX.DAEMON provides additional controls for the use of kernel services such as setuid(), which changes a issuer's UID in the OS/390 UNIX environment. In OS/390 UNIX, any user can issue a setuid() that follows a successful __passwd() call, which can be used to verify and/or change a user's password, to the target user ID. However, when a user want to change his/her user identity without knowing the target user's password, daemon authority is required.

As you might have noticed, the FTP server does not have to have the access permission to BPX.DAEMON, because it can authenticate a target user with his password, which has been entered by the FTP client. For the anonymous FTP support, however, the server must have daemon authority, because no password is available for an anonymous user. We recommend you use the SURROGAT class profile rather than BPX.DAEMON to prevent possible security exposure.

3.3.2.1 The controlled program environment

In OS/390 UNIX System Services, security critical functions such as authenticating users or switching identities require that the process runs in an address space that is a controlled program environment.

In a controlled program environment (also called a *program-controlled* address space), all programs that are loaded into the address space must either:

- Be loaded from an MVS program library (PDS or PDSE) that is defined in RACF class PROGRAM, or
- Be loaded from an HFS file that has the extended attribute “program controlled” (PROGCTL) turned on.

Any uncontrolled program from the HFS or an MVS library that is loaded into the address space will cause it to become uncontrolled, often called *dirty*. Once an address space becomes uncontrolled, it cannot be reverted into a controlled program environment. (There are fairly complicated methods in TSO/E involving the TSO/E Service Facility that can create a program-controlled task structure in an uncontrolled address space, but these are irrelevant in the OS/390 UNIX environment.)

The requirement for a controlled program environment makes it more difficult for intruders and Trojan horse programs to misuse a security critical process. All attempts to replace or modify code in the address space will cause it to become uncontrolled and the security critical functions will fail.

It should be noted that the requirement to be in a library defined in class PROGRAM or to have the PROCTL extended attribute turned on also applies to dlls and GWAPI plug-ins. A typical example for defining controlled libraries and programs:

```
RDEFINE PROGRAM * ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('cee.SCEERUN'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('imw.SIMWMOD1'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('TCPIP.SEZALINK'//NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

And an example for defining a module in the HFS a program-controlled:

```
BOCHE § SC57:/web/cert $ extattr +p pwapi.so
BOCHE § SC57:/web/cert $
```

3.3.2.2 Daemon authority

As mentioned in 3.3.2, “OS/390 UNIX System Services level security” on page 62, switching into another UID and user ID without first authenticating the user requires the daemon authority. Why is this so potentially dangerous that a special authority from RACF is required? A program that can switch into the identity of any UID has almost the same authority as a superuser in traditional UNIX, this could make an OS/390 system lose accountability, at least as far as user IDs that have a UID defined are concerned. When a process with daemon authority switches into another UID without authenticating the user first, RACF will search for a user ID that this UID belongs to in the pretty much same way it resolves

UIDs to user IDs in an `ls -l` display. The process will then run with the UID and the user ID determined by RACF. As a consequence, a process with daemon authority can switch into any RACF user ID that has an UID defined.

It could be dangerous if a process with daemon authority is allowed to switch into UID(0) without further security control. If RACF would resolve UID(0) into a user ID such as OMVSKERN that has daemon authority by itself, this could allow someone to run a rogue program that assumes other user identities at will. To avoid this situation, the SUPERUSER parameter in member BPXPRMxx in SYS1.PARMLIB was created. It allows to specify a user ID with UID(0) that RACF will use when switching into UID(0) with daemon authority. This user ID (the default is BPXROOT) must not be defined with daemon authority.

For a process to have daemon authority, the following must all be true:

- The user ID associated with the process must be defined with an OMVS segment that specifies UID(0).
- The user ID associated with the process must have READ authority to profile BPX.DAEMON in class FACILITY.
- The process must run in a controlled program environment.

It can easily be seen that daemon authority is far-reaching and offers possibilities for compromising the security and integrity of the system. Therefore, READ access to profile BPX.DAEMON should only be given to as few user IDs in the system as ever possible. Some examples of users/daemons that need READ access to BPX.DAEMON are:

- The UNIX System Services kernel user ID (the default is OMVSKERN)
- cron
- uucpd
- rlogind
- rshd

3.3.2.3 BPX.SERVER authority

As mentioned in 3.3.2, “OS/390 UNIX System Services level security” on page 62, when a server program issues the service pthread_security_np() to create a thread running under this user ID and UID, its user ID needs access to profile BPX.SERVER in class FACILITY.

The access level to BPX.SERVER that the program needs is dependent on a number of conditions:

- Before issuing pthread_security_np(), the server program does not authenticate the user ID using __passwd().
  - If the server’s user ID has UPDATE access to BPX.SERVER, the thread is created and all authorization requests for resources accessed by the thread are checked against the user ID or UID of the thread.
  - If the server’s user ID has READ access to BPX.SERVER, the thread is created and all authorization requests for resources accessed by the thread are checked against the user ID or UID of the thread as well as the user ID or UID of the server.
- Before issuing pthread_security_np(), the server program authenticates the user ID and specifies the correct password.

  - The server's user ID needs READ access to BPX.SERVER; all authorization requests for resources accessed by the thread are checked against the user ID or UID of the thread.
- The user ID of the server program has READ access to profile BPX.SRV.userid in class SURROGAT
  - The server's user ID needs READ access to BPX.SERVER; all authorization requests for resources accessed by the thread are checked against the user ID or UID of the thread.

In addition to the need to have access to BPX.SERVER, the process must also run in a controlled program environment to be able to use the pthread_security_np() service. One of the main users of this service is the IBM HTTP Server for OS/390.

3.3.3 Why is OS/390 UNIX System Services a more secure UNIX?

OS/390 UNIX System Services takes advantage of the inherent strengths of MVS and OS/390, including the security mechanisms of RACF. Working with RACF allows OS/390 UNIX to provide these OS/390-exclusive enhancements to UNIX security:

1. No /etc/passwd file. UNIX System Services relies on RACF for user authentication. This means that user information is not kept in /etc/passwd and /etc/security/passwd, respectively, and that all user administration is performed outside the OS/390 UNIX System Services environment. It is easy to configure OS/390 UNIX in a way that no access to the RACF database is possible from a UNIX process. This prevents brute-force password attacks against UNIX passwords and stops intruders from altering user information from within OS/390 UNIX.
2. Granularity of auditing and reporting. UNIX System Services and RACF provide comprehensive auditing, allowing numerous events to be audited. The auditing information is written to SMF data sets which can be made inaccessible even to superusers. Reporting is based on an open architecture that allows the use of practically any reporting package. This provides better detection of suspicious events.
3. Protection of daemon programs from modification and misuse. Programs that perform security-critical functions must run in a controlled program environment. A modification to a module in such an address space or an attempt to load a module from another, uncontrolled library will cause the program to fail.
4. Superuser granularity control, a function that allows non-superusers authorized by RACF to use services that would otherwise require the user to be a superuser. This can reduce the number of superusers required in a system.
5. The storage keys implemented in the S/390 and z900 hardware together with the concept of address in OS/390 isolates applications from each other and from the operating system. A program that exceeds its allocated storage fails with a storage exception rather than overwriting storage areas of the operating system or other applications. This greatly reduces the possible damage that could be done by buffer overflows and similar problems.

6. TCP/IP stacks, ports, and network addresses can be protected with RACF to prevent unauthorized users and programs from using them.
7. The ability to assign different user identities (user IDs and UIDs) not only to processes but also to threads within a process allows access control in OS/390 UNIX to be performed on a much more fine-grained basis than is possible in traditional UNIX environments. Specifically, programs such as the IBM HTTP Server for OS/390 can run each thread under the client's identity. This allows the access authority of the client to be checked for all requests.

3.3.4 Access permission to HFS files and directories

To be able to access files in the HFS, a UID needs to be assigned to the process or thread that tries to open the file. This will be the case if the RACF user ID has a valid OMVS segment with an OMVS UID or if a default user with a valid UID has been defined.

When a process issues its first call to an OS/390 UNIX service, it is said to be *dubbed*; it is given an OS/390 UNIX identity that is based on the OMVS segments of the associated RACF user and group profiles.

The basic access algorithm can be deduced by reference to Figure 30.

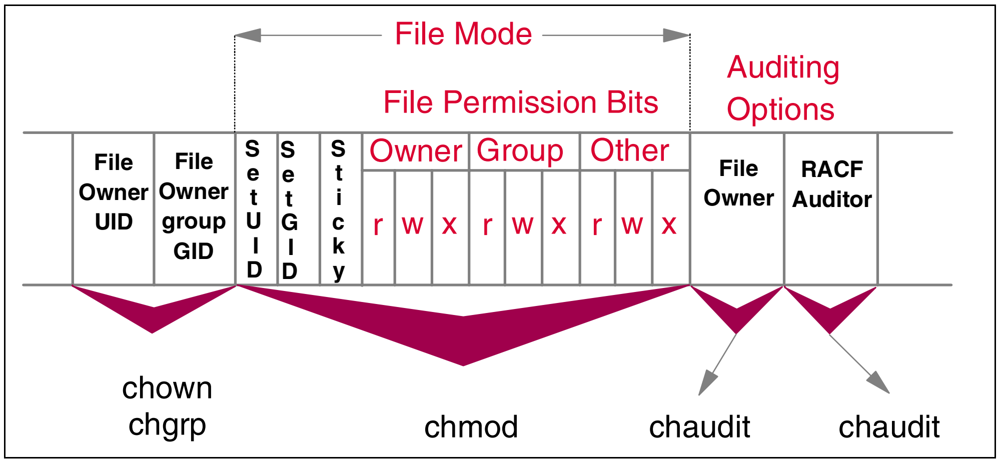


Figure 30. Hierarchical File System file security packet

Every file in the Hierarchical File System has a file security packet (FSP) assigned to it. This FSP contains the identification of the owning UID and GID along with information about what level of access (read, write, execute or rwx in short) is granted to the three user categories that are considered to exist in this environment:

- The owner permissions

  Any processes with an effective UID that matches the UID of the file owner can access the file under the defined permission. Please note that the owner of a file may restrict him or herself to read and execute (r-x) access. A superuser can change the file owner by issuing a `chown` shell command.

- The owning group permissions

  Any processes with an effective GID that matches the GID of the file group can access the file under the defined permission. A superuser or the file owner can change the GID of the file by issuing a `chgrp` shell command.

- The other permissions

  This permission class is applied to any process that is not the file owner, nor a member of the file owner's group. It is generally known as world access.

Note: A process running with an effective UID(0) can always read any file, even if not specifically permitted to do so by the permission bits.

See Table 3 for an overview of types of access and the permissions granted by the accesses:

Table 3. File access types and permission bits

Access Type	Permission for File	Permission for Directory
Read	Permission to read or copy the contents	Permission to read, but not search, the contents
Write	Permission to change, add to, or delete from the contents	Permission to change, add, or delete directory entries, that is, HFS files, links, or subdirectories
Execute	Permission to run the file; this permission is used for executable files	Permission to search the directory

Notes:

1. To access an HFS file, search permission to all directories in the path name of the file is required. Read permission is required for some options of some commands.
2. To create new HFS files or directories, write permission to the directory in which they will be created is required.

To change the permission bits for a file, the file owner or superuser can use one of the following:

- The UNIX System Services ISPF shell
- The `chmod` shell command
- The `chmod()` API from a program

When accessing a file in the Hierarchical File System, the file system looks up the effective UID and GID of the current process and compares these to the owning UID and GID. If this fails, the check proceeds under the assumption of world access. Access checking is performed by the OS/390 UNIX System Services kernel by using the appropriate callable service.

If RACF denies access, an error message will be logged to SYSLOG as shown in Figure 31:

```
ICH408I USER(BOCHE   ) GROUP(SYS1    ) NAME(ULRICH BOCHE IBM GER)
  /u/boche/.sh_history
  CL(FSOBJ   ) FID(01D7C4C7D6C5F2000320000000260000)
  INSUFFICIENT AUTHORITY TO OPEN
  ACCESS INTENT(RW-)  ACCESS ALLOWED(GROUP ---)
```

Figure 31. Resource access authorization error

This message is the usual error message issued by RACF for access violations, but a short explanation of the different elements may be helpful:

- The first line identifies the user associated with the process or thread. USER is the RACF user ID, GROUP the current connect group, and NAME is the username in the user profile.
- The second line identifies the failing resource, in this case the full path name of an HFS file.
- In the third line, CL identifies the RACF class, in this case FSOBJ. This is one of the OS/390 UNIX classes which are only used to hold auditing options, so don't look for any profiles in this class. FSOBJ means *file system objects*, in other words, files. DIRACC would mean read/write access to a directory while DIRSRCH would appear if a directory search failed. For a description of all classes used, see Appendix A in *OS/390 Security Server for OS/390 RACF Security Administrator's Guide*, SC28-1915. The FID parameter is normally only of interest to the IBM service organization for debugging purposes.
- The forth line contains the reason for the failure.
- In the fifth and last line, ACCESS INTENT shows the access modes requested by the program. "`RW-`" in this case means the program tried to open the file for both reading and writing. ACCESS ALLOWED shows the access level that would have been allowed: GROUP means that one of the groups the user is connected to matched the owning group, so the group permissions were used, and they were "`---`" which means no access was allowed.

See Appendix B, "Default permissions for HFS files in OS/390 UNIX" on page 317 for the default settings of the permission bits using different applications to create files or directories in the UNIX System Services.

3.3.5 Displaying files and directories

The shell command `ls` is used to list the contents of directories. The following example shows the output of the command `ls -l`:

```
-rw-r--r--   1 OMVSKERN SYSPROG       9 Feb 28 14:58 syslog.pid
-rwxr-xr-x   1 OMVSKERN SYSPROG     157 Feb  4  1999 syslogd.start
-rwxr-xr-x   1 OMVSKERN SYSPROG     547 Feb 10  1999 t03dns.boot
-rw-r--r--   1 OMVSKERN SYSPROG     201 Aug 10  1999 tcpipa.data
-rw-rw-rw-   1 OMVSKERN SYSPROG       0 Feb 11  1998 telnetd.stderr
-rw-r--r--   1 OMVSKERN SYSPROG   20334 Feb 22  2000 testu03n
-rw-r--r--   1 OMVSKERN SYSPROG      62 Jun 22  2000 trmd.r2615c.env
-rwxr-xr-x   1 OMVSKERN SYSPROG     119 Apr 27  1998 u.map
-rw-r--r--   1 OMVSKERN SYSPROG     792 Mar  6 14:25 utmpx
-rw-r--r--   1 OMVSKERN SYSPROG   15520 Feb  4 13:25 yylex.c
-rw-r--r--   1 OMVSKERN SYSPROG   22559 Feb  4 13:25 yyparse.c
drwxr-xr-x   2 OMVSKERN SYSPROG    8192 Jan 29  1998 zoneinfo
ULRICH @ RA03:/etc>
```

The listing shows the user ID of the file owner and the group name of the owning group although the information in the HFS only contains the UID and GID, respectively. The cross-reference between a UID and a user ID and a GID and a group name is done by a RACF service. Resolving a UID to a user ID (or a GID to group name) can be ambiguous if there is more than one user with the same UID assigned (or more than one group with the same GID).

3.3.5.1 Class UNIXMAP and VLF classes IRRUMAP and IRRGMAP

For the cross-referencing, RACF uses profiles in class UNIXMAP. The profile names for UIDs are Unnnn where nnnn is the UID number; for GIDs the profile names are Gnnnn. The user IDs associated with this UID are entered in the access list of the profile. If class UNIXMAP is used exclusively, RACF will always return the same user ID for a UID.

For performance reasons, VLF classes IRRUMAP and IRRGMAP are used. In the data space associated with IRRUMAP, UIDs and corresponding user IDs are stored at the time when this user ID is first used for an OS/390 UNIX process after an IPL or after the table is rebuilt. The same is true for VLF class IRRGMAP and groups. When IRRUMAP and IRRGMAP are active, the user ID returned by RACF for a given UID will depend on the sequence in which user IDs have created processes. Therefore, the selection made by RACF among the user IDs sharing a common UID may appear rather arbitrary.

3.3.5.2 Application Identity Mapping (AIM)

In OS/390 V2R10, RACF introduces Application Identity Mapping (AIM) as a replacement for class UNIXMAP and VLF classes IRRUMAP and IRRGMAP. With AIM, RACF resolves UIDs to user IDs and GIDs to group names with the help of an alternate index into the RACF database. Making full use of this function requires all systems sharing a RACF Database to be at the OS/390 V2R10 level or higher.

When AIM is used, RACF will always return the same user ID for a UID and the same group name for a GID, respectively.

3.3.6 UID/GID assignment to a process

As mentioned earlier, the first use of an OS/390 UNIX service causes the OS/390 address space to be “dubbed” into an OS/390 UNIX process. The User Security Packet (USP) is an important control block created by dubbing. Among the information it holds are the UIDs relevant to the process:

Real UID: At process creation, the real UID identifies the user who has created the process.

Effective UID: Each process also has an effective UID. The effective UID is used to determine the owner access privileges of a process.

Normally this value is the same as the real UID. It is possible, however, for a program that resides in the Hierarchical File System to have a special flag set that, when this program is executed, changes the effective UID of the process to the UID of the owner of the program. A program with this special flag set is said to be a set-user-ID program. This feature provides additional permissions to users while the set-user-ID program is being executed.

Saved UID: Used to save the effective UID of a process. When an OS/390 UNIX user tries to change the UID, the saved_UID field is checked to see if the UID is the original one.

Real GID: At process creation, the real GID identifies the group of the user for which the process was created.

Effective GID: Each process also has an effective group. The effective GID is used to determine the group access privileges of a process. Normally this value is the same as the real GID. Some programs, however, have a special flag set that, when the program is executed, changes the effective GID of the process to the GID of the owner of this program. A program with this special flag set is said to be a set-group-ID program. Like the set-user-ID feature, this provides additional permission to users while the set-group-ID program is being executed.

Saved GID: Used to save the effective GID of a process. When an OS/390 UNIX user tries to change the GID, the saved GID is checked to see if the GID is the original one.

The real UID/GID tell who really owns the process; the effective UID and GID are used for file access permission checks. The saved values of UID and GID are stored by the exec() function.

3.3.7 Defining UNIX System Services users

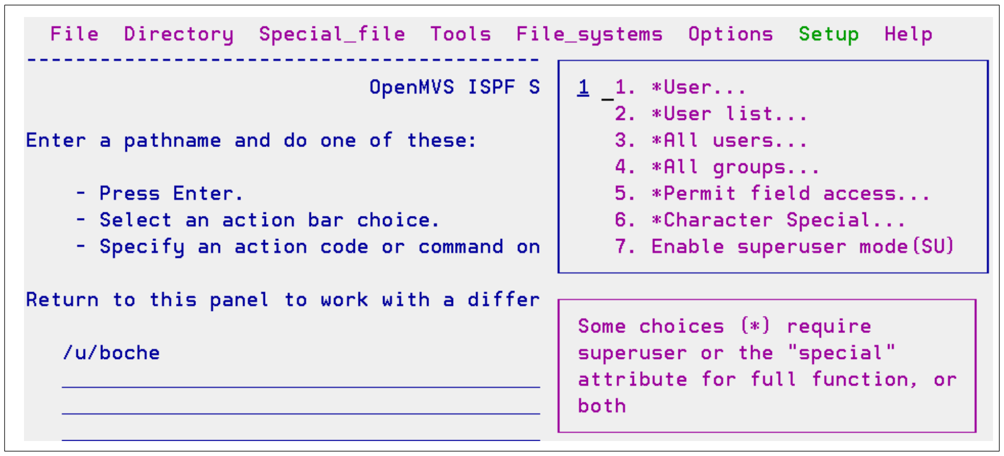
To define new UNIX System Services users, use the RACF command `ADDUSER`. The new user needs to have an OMVS segment with a UID. Also, the users default group needs to have a GID assigned. If needed, a new group can be defined using the ADDGROUP command.

```
ADDGROUP usrgrp OMVS(GID(10))
```

Then define the new user:

```
ADDUSER user01 DFLTGRP(usrgrp) OMVS(UID(20) HOME('/u/user01') -
PROGRAM('/bin/sh'))
```

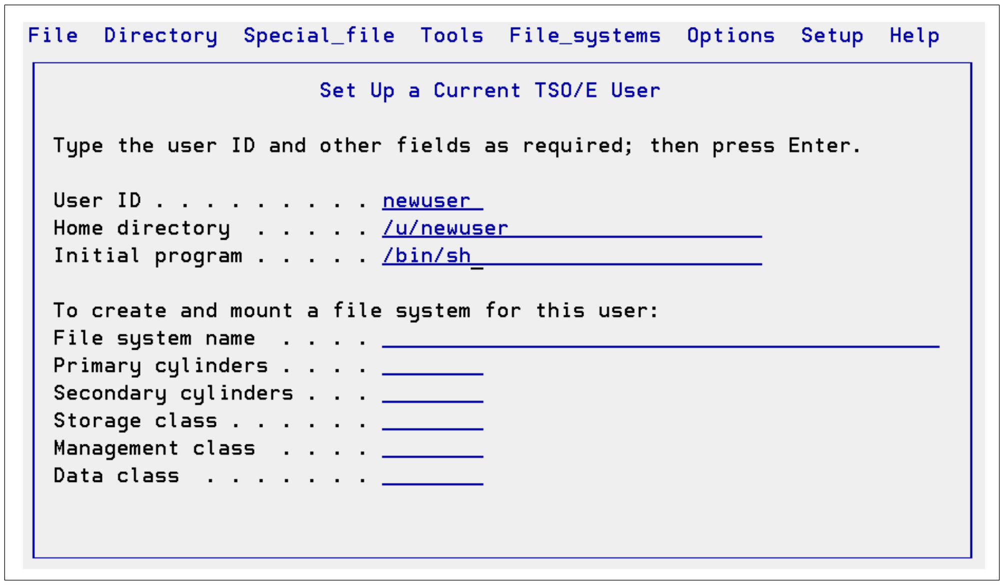
Note: For the UID, any value up to 2,147,483,647 can be used. However, UID(0) should only be assigned to superuser IDs. To avoid the task of manually keeping track of all the UIDs that are in use we recommend using standard procedures to define users without OMVS segments, and using the TSO ISHELL to add the OMVS segment to the user ID. The TSO ISHELL will keep track of the UIDs it has issued and will assign the next available UID to a new user. Figure 32 shows how to invoke the appropriate ISHELL function:



```
 File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
------------------------------------------
                           OpenMVS ISPF S   1 _1. *User...
                                               2. *User list...
Enter a pathname and do one of these:          3. *All users...
                                               4. *All groups...
    - Press Enter.                             5. *Permit field access...
    - Select an action bar choice.             6. *Character Special...
    - Specify an action code or command on     7. Enable superuser mode(SU)

Return to this panel to work with a differ  Some choices (*) require
                                            superuser or the "special"
   /u/boche                                 attribute for full function, or
   ______________________________________   both
   ______________________________________
   ______________________________________
```

Figure 32. The setup drop-down menu in IDPF ISHELL

Selecting **Setup** -> **1. *User...** brings up the window shown in Figure 33 that will assign the next free UID to an existing user and also allows to set the other OMVS segment parameters:



```
File  Directory  Special_file  Tools  File_systems  Options  Setup  Help

                     Set Up a Current TSO/E User

 Type the user ID and other fields as required; then press Enter.

 User ID . . . . . . . . . newuser_
 Home directory  . . . . . /u/newuser____________________
 Initial program . . . . . /bin/sh_______________________

 To create and mount a file system for this user:
 File system name  . . . . ____________________________________________
 Primary cylinders . . . . ________
 Secondary cylinders . . . ________
 Storage class . . . . . . ________
 Management class  . . . . ________
 Data class  . . . . . . . ________
```

Figure 33. ISHELL window to assign a UID and other OMVS segment parameters to a user

For a user to be a UNIX System Services user, the user's default group must be a UNIX System Services group, that is, the group must be associated with a valid OS/390 UNIX group ID.

For the new user01 to be able to log in to the UNIX System Services shell environment, you should create the home directory /u/user01 and give the user at least:

- Execute permission to the "/" and "/u" directories.
- Read, write and execute permission to the /u/user01 directory.

If you do not give user01 sufficient permission, any attempt to log in to the OS/390 UNIX shell will fail. Note that defining /u/userid as the home directory for a user is just a frequently used convention; any valid directory name can be defined as a user's home directory.

3.3.8 Default user

Since OS/390 V2R5, TCP/IP itself is an OS/390 UNIX application. As a consequence, any user of TCP/IP services needs to be an OS/390 UNIX user. This means, for instance, that any FTP user who logs in with a RACF user ID needs to have a UID assigned or the FTP session will fail. The same is true for a user who points his or her Web browser to a URL that is protected and requires the user to authenticate with a user ID and password or a client certificate.

For various reasons, some installations do not want to assign individual UIDs to all these users. Some do not want all their users to be able to use the OMVS shell or the ISPF ISHELL; others do not want to spend the administrative overhead involved with defining OMVS segments for all users. These installations can define a default UID and GID for use with those user IDs that do not have an OMVS segment defined.

When you define an OMVS segment for the default user, you should consider the following entries:

UID

You can use any value for default user and default group. Administrators often like to use a value for the default UID that makes it stand out from other, regular UNIX System Services users. Of course, assigning UID(0) to the default user would create a major security exposure.

HOME

The home directory is the initial HFS directory for users who enter the TSO command OMVS or use Telnet to enter an OS/390 UNIX shell. Generally, users of the default UID should not be shell users. Therefore, the following alternatives seem appropriate:

- Define a home directory where the default user does not have write permission, such as the root (/) directory.
- Define the HOME directory as the /tmp directory which is designed for temporary files created by different users.

PROGRAM

The shell program is executed when a user enters the OS/390 UNIX shell. The default shell program is `/bin/sh`. If you do not want users running in a shell environment with the default UID and GID, then define the PROGRAM parameter for the user's OMVS segment as:

```
PROGRAM('/bin/false')
```

This will cause any attempt to enter the shell to terminate. Note, however, that this does not restrict TSO users from using the ISHELL.

An example of the setup required for a default user and group follows:

1. Activate the FACILITY class and define the OS/390 UNIX default user profile, BPX.DEFAULT.USER:

```
RDEFINE FACILITY BPX.DEFAULT.USER APPLDATA('UDSSUSR/USSGRP')
```

2. Define a default GROUP and USER:

```
ADDGROUP USSGRP OMVS(GID(17))
ADDUSER USSUSR DFLTGRP(USSGRP) NAME('USS DEFAULT USER') -
OMVS(UID(88) HOME(/) PROGRAM('/bin/false'))
```

Note: You can use any value for UID and GID, except zero for the UID.

3. Refresh the FACILITY class profile:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

The format of the application data is exactly as shown when a default is being set up for both USER and GROUP OMVS segments. To set up a default for the USER OMVS segment only, the format is:

```
APPLDATA('USSUSR')
```

There is no option to set up a default GROUP OMVS segment without a user.

Note: Because many OS/390 UNIX processes now run with the default user's UID, the number of allowed processes per UID as defined in BPXPRMxx with the MAXPROCUSER statement may be too low. Note that processes executing under the default user's UID do not get counted toward MAXPROCUSER unless APAR OW36890 has been applied (OS/390 V2R5 - OS/390 V2R8).

Profile BPX.DEFAULT.USER in class FACILITY is used as follows:

1. A user requests a UNIX service and the kernel dubs the user.
2. The kernel calls the security product to extract the UID, GID, HOME, and PROGRAM information.
3. The security product attempts to extract the OMVS segment associated with the user. If it is not defined, it tries to extract and use the OMVS segment for the default user that was listed in the BPX.DEFAULT.USER profile.

Note: If you allow users to access UNIX System Services resources using the default UID and GID, be aware that these users can access all "world readable" HFS files and directories. In an HFS environment, it is very often infeasible to avoid world readable files because of the limited possibilities offered by the POSIX permission bits. Therefore, the use of the default UID and GID requires careful planning and risk assessment.

Also, if you have trusted or privileged started tasks defined in the STARTED class or in ICHRIN03 these started tasks will be superusers if they use an OS/390 UNIX service even if they do not have an OMVS segment defined.

3.3.9 Superuser

In any UNIX system, a user with a UID of zero has special privileges. This user is said to be a superuser. Superusers are very powerful users, therefore, the number of users with UID(0) should be kept to a minimum and these IDs should be handled with great care.

At installation, you have to define at least one superuser and a group to which the superuser will belong. The group could be an existing group as long as it has a GID assigned.

```
ADDGROUP OMVSGRP OMVS(GID(1))
```

Note: You can use any group ID for the group to which the superuser will belong.

Then define the superuser:

```
ADDUSER OMVSKERN DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/') PROGRAM('/bin/sh'))
```

In this case, the superuser is named OMVSKERN and its default group is OMVSGRP.

In most UNIX systems, a UID of zero gives unlimited rights to access and manipulate files in the HFS and to change the identity of the process that is running with a UID of zero. In most UNIX systems this means that the system administrator normally has a UID of zero, and any server programs that need to change the identity of forked processes run with a UID of zero.

In an OS/390 UNIX system running with OS/390 UNIX System Services security, limitations to the authorities of a superuser apply, such as:

- Using a service such as setuid(), even a superuser can only switch into another UID without specifying the password if the user ID has READ access to profile BPX.DAEMON in class FACILITY.
- The use of the `su` command to switch into another user ID requires the user ID's password or READ access to profile BPX.SRV.userid in class SURROGAT.
- Extended attributes for HFS files such as program-controlled (PROGCTL), APF-authorized (APF), and shared library (SHARELIB) can only be set with READ authority to the appropriate RACF profile in class FACILITY (BPX.FILEATTR.PROGCTL, BPX.FILEATTR.APF, or BPX.FILEATTR.SHARELIB, respectively).
- Use of the ptrace function of dbx to debug programs running with APF authority or with BPX.SERVER authority requires READ access to profile BPX.DEBUG in class FACILITY.

3.3.9.1 Superuser granularity

Even with the restrictions imposed on superusers in OS/390 UNIX System Services, a superuser still has far-reaching authority. Therefore, it should be each installation's goal to keep the number of users with superuser authority to an absolute minimum. This is an important contribution to the overall security and accountability of the OS/390 system.

A number of system programmers and administrators do not need full superuser authority. They just need to perform a few selected functions that cannot normally be executed without superuser authority. In this situation, the number of superusers (or users able to switch into superuser mode through access to BPX.SUPERUSER) that are needed in the system can be reduced by using a function called *superuser granularity*.

Superuser granularity allows a non-superuser to successfully execute a function that would normally require superuser authority if the user has access to a certain resource in RACF class UNIXPRIV. For example, a file system can normally only be mounted by a superuser. However, a user with a non-UID(0) user ID can successfully mount file systems if the user ID has access to profile SUPERUSER.FILESYS.MOUNT. READ access will allow file systems with the nosetuid option, while UPDATE access will also allow the user to mount file systems with the setuid option.

Table 4 shows the resource names that are used in class UNIXPRIV and lists the privileges associated with each resource.

Table 4. Resource names in the UNIXPRIV class for OS/390 UNIX privileges

Resource name	**OS/390 UNIX privilege**	**Access required**
CHOWN.UNRESTRICTED	Allows all users to use the `chown` command to transfer ownership of their own files.	NONE
SUPERUSER.FILESYS.CHOWN	Allows user to use the `chown` command to change ownership of any file.	READ

Resource name	OS/390 UNIX privilege	Access required
SUPERUSER.FILESYS [1]	Allows user to read any HFS file and to read or search any HFS directory.	READ
	Allows user to write to any HFS file and includes privileges of READ access.	UPDATE
	Allows user to write to any HFS directory and includes privileges of UPDATE access.	CONTROL (or higher)
SUPERUSER.FILESYS.MOUNT	Allows user to issue the `mount` command with the `nosetuid` option and to unmount a file system mounted with the `nosetuid` option.	READ
	Allows user to issue the `mount` command with the `setuid` option and to unmount a file system mounted with the `setuid` option.	UPDATE
SUPERUSER.FILESYS.QUIESCE	Allows user to issue the `quiesce` and `unquiesce` commands for a file system mounted with the `nosetuid` option.	READ
	Allows user to issue the `quiesce` and `unquiesce` commands for a file system mounted with the `setuid` option.	UPDATE
SUPERUSER.FILESYS.PFSCTL	Allows user to use the `pfsctl()` callable service.	READ
SUPERUSER.FILESYS.VREGISTER [2]	Allows a server to use the `vreg()` callable service to register as a VFS file server.	READ
SUPERUSER.IPC.RMID	Allows user to issue the `ipcrm` command to release IPC resources.	READ
SUPERUSER.PROCESS.GETPSENT	Allows user to use the `w_getpsent` callable service to receive data for any process.	READ
SUPERUSER.PROCESS.KILL	Allows user to use the `kill()` callable service to send signals to any process.	READ
SUPERUSER.PROCESS.PTRACE [3]	Allows user to use the `ptrace()` function through the dbx debugger to trace any process. Allows users of the `ps` command to output information on all processes. This is the default behavior of `ps` on most UNIX platforms.	READ
SUPERUSER.SETPRIORITY	Allows user to increase own priority.	READ

Notes:
1. Authorization to the SUPERUSER.FILESYS resource provides privileges to access only local Hierarchical File System (HFS) files. No authorization to access Network File System (NFS) files is provided by access to this resource.
2. The SUPERUSER.FILESYS.VREGISTER resource authorizes only servers, such as NFS servers, to register as file servers. Users who connect as clients through file server systems, such as NFS, are not authorized through this resource.
3. Authorization to the resource BPX.DEBUG in the FACILITY class is also required to trace processes that run with APF authority or BPX.SERVER authority. For more information about administering BPX profiles, see *OS/390 UNIX System Services Planning*, SC28-1890.

SETROPTS RACLIST needs to be issued for class UNIXPRIV to make the profiles resident in storage. Therefore any addition or modification to profiles in this class requires the profiles to be refreshed using the command:

```
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

3.3.10 Started task user IDs

Starting with RACF 2.1, the association of a user ID with a started procedure can be done with profiles in resource class STARTED. This allows for a dynamic definition of started procedures and their associated user IDs as opposed to the use of the ICHRIN03 table which required an IPL of the system.

Typical entries in this resource class originating from UNIX System Services are the procedures for TCP/IP, FTPD, InetD, syslogd, the HTTP Server and a number of other applications.

The following example shows how to define a resource profile in class STARTED for the OS/390 HTTP Server.

```
SETROPTS GENERIC(STARTED)
SETROPTS CLASSACT(STARTED) RACLIST(STARTED)
```

The above two commands are needed only if you have not configured the RACF STARTED class in your OS/390 system. The next command (`RDEFINE`) associates the user WEBSRV in group USSGRP with the started task WEBSERV:

```
RDEFINE STARTED WEBSERV.* STDATA(USER(WEBSRV) GROUP(USSSGRP) -
        PRIVILEGED(NO) TRUSTED(NO) TRACE(NO))
SETROPTS RACLIST(STARTED) REFRESH
```

The profiles of RACLISTed classes are cached in a data space. Therefore, you need to refresh class STARTED every time you have made changes to any profiles.

To display the definition for a particular started class profile, use the RACF GENERAL RESOURCE SERVICES - DISPLAY panel or issue the following command:

```
RLIST STARTED WEBSERV.* STDATA
```

Figure 34 is a sample of the output produced by the `RLIST` command.

```
CLASS      NAME
-----      ----
STARTED    WEBSERV.* (G)
LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----  --------   ----------------  -----------  -------
00     SYS1           NONE              NONE     NO
............................
N
STDATA INFORMATION
------------------
USER= WEBSRV
GROUP= USSGRP
TRUSTED= NO
PRIVILEGED= NO
TRACE= NO
```

Figure 34. RLIST output for STARTED class profile

When you start the Web server with the operator command `START WEBSRV`, you will see the following messages on the OS/390 console:

```
IEF695I START WEBSERV WITH JOBNAME WEBSERV IS ASSIGNED TO
USER WEBSERV, GROUP OMVSGRP
```

3.3.11 FACILITY class profile BPX.SUPERUSER

This profile can be used to allow users with a nonzero UID to change the effective UID of their process into a UID of zero. If you have users who occasionally, as part of their daily system work, need to have superuser privileges, you can assign them a nonzero UID and permit them use of BPX.SUPERUSER. The alternative is to define all such users with a permanent UID of zero, which means that they perform all their work as superusers, even work that does not require them to be superusers. By using this profile you can limit the amount of time any user runs in superuser mode. Remember that in superuser mode, the user can, perhaps by mistake, delete the root file system of your Hierarchical File System. If a user who is permitted access to BPX.SUPERUSER logs on to the OS/390 UNIX interactive shell, the user can type in a command to switch into superuser mode, and an exit command to return to his or her normal nonzero UID environment.

Below are examples of definitions for users who can switch to superuser status.

Before defining these users, you need to create profile BPX.SUPERUSER in class FACILITY. We assume the FACILITY class is already defined and activated. Enter the `RDEFINE` command shown in the following example to define this profile:

```
RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)
```

Permit all users who need superuser authority to this profile. We give the TSO/E user ID SYSPROG permission to use `su` to obtain superuser authority. It is assumed that the default group for SYSPROG is set up with a GID.

Use the RACF commands shown in the following example:

```
ALTUSER SYSPROG OMVS(UID(7) HOME('/u/sysprog') PROGRAM('/bin/sh'))
PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(SYSPROG) ACCESS(READ)
```

Another alternative is to define a group, say SUPERUSR. You would then add users that need superuser permission to the group. To add the user ID SYSPROG to this group and then permit this group to the BPX.SUPERUSER profile, enter:

```
CONNECT (SYSPROG) AUTH(USE) GROUP(SUPERUSR) OWNER(SYS1)
PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(SUPERUSR) ACCESS(READ)
```

A user who does not have READ permission to the FACILITY class profile BPX.SUPERUSER and wants to change into superuser status will receive the following error message:

```
OZECH @ RA03:/u/ozech>su
FSUM5011 su: User not authorized to obtain superuser authority
```

A daemon program that wants to make use of the authority to BPX.SUPERUSER to gain superuser status needs to invoke the seteuid() service. Therefore, using a non-UID(0) user ID with access to BPX.SUPERUSER instead of a superuser user ID for a daemon or server program is a solution only if the program does indeed support this function. An example of such a program is OS/390 SMP/E in OS/390 V2R7 and later.

3.3.12 FACILITY class profile BPX.DAEMON

The profile BPX.DAEMON is very important to the concept of OS/390 UNIX level security and the overall system security in OS/390. We have discussed this concept extensively in 3.3.2, "OS/390 UNIX System Services level security" on page 62. If you are not familiar with identity switching, daemon authority, and the controlled program environment, please return to this section.

To enable OS/390 UNIX level security, define profile BPX.DAEMON in class FACILITY:

```
RDEFINE FACILITY BPX.DAEMON UACC(NONE)
```

Note: Activating OS/390 UNIX level security in a system that did not use this level of security before could cause some daemon programs to fail. Before defining this profile or the FACILITY class profile BPX.SERVER, you should make sure that all daemons that use security-critical services are loaded from a controlled program environment and that appropriate authority to SURROGAT class profiles has been established. For details see 3.3.2.1, "The controlled program environment" on page 64.

If this is the first FACILITY class profile that your installation is going to use, you need to activate the FACILITY class with the following commands:

```
SETROPTS CLASSACT(FACILITY) GENERIC(FACILITY) AUDIT(FACILITY)
SETROPTS RACLIST(FACILITY)
```

Depending on the technique you use to start your server programs, some of them may inherit their initial user identity from the kernel address space. This will be the case if you start server programs from the /etc/rc shell script that is executed during startup of UNIX System Services. In that case, the user ID of the kernel address space must be authorized to the BPX.DAEMON resource:

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(OMVSKERN) ACCESS(READ)
```

If you start server programs via OS/390 start commands or from shell scripts that execute after startup of OS/390 UNIX, you may need to give some of these user IDs the access to the BPX.DAEMON FACILITY class resource.

3.3.12.1 User ID BPXROOT

In 3.3.2.2, "Daemon authority" on page 64, we explained the use of user ID BPXROOT. This user ID should not be used to log on to any interactive services, so it is appropriate to define it as a protected user ID (using the NOPASSWORD parameter). To define the BPXROOT user ID, enter:

```
ADDUSER BPXROOT DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/') -
PROGRAM('/bin/sh')) NOPASSWORD
```

In SYS1.PARMLIB member BPXPRMxx, the SUPERUSER parameter can be used to specify a user id to be used instead of BPXROOT.

Note: Make sure you do not give the BPXROOT user ID or its replacement READ (or higher) access to profile BPX.DAEMON in class FACILITY. As explained in 3.3.2.2, "Daemon authority" on page 64, this would be a security exposure.

3.3.13 Additional BPX.* FACILITY class profiles

For security reasons, you may need to define these additional FACILITY class profiles:

3.3.13.1 BPX.DEBUG

Users with read access to profile BPX.DEBUG in class FACILITY can use the ptrace function of dbx to debug programs that run APF authorized or authority to profile BPX.SERVER in class FACILITY.

3.3.13.2 BPX.FILEATTR.APF

This controls which users are allowed to set the APF authorized attribute for an HFS file. This authority allows the user to create a program that will run APF authorized. This is similar to the authority of allowing a programmer to update SYS1.LINKLIB or SYS1.LPALIB.

3.3.13.3 BPX.FILEATTR.PROGCTL

This controls which users are allowed to set the program-controlled attribute for an HFS file. Programs marked with this attribute can execute in program-controlled address spaces.

3.3.13.4 BPX.FILEATTR.SHARELIB

OS/390 V2R9 introduced the concept of user and system-shared library objects. Shared libraries allow multiple processes to share subroutines in object libraries more efficiently.

Programs in a user-shared library (filenames with a suffix of ".so") are loaded into the kernel-shared library region data space on a page boundary. Programs in a system-shared library are loaded into the kernel-shared library region data space on a megabyte boundary, use a single-page table for all address spaces that use them (similar to the LPA) and do not interfere with the virtual storage used by the application program.

A program in the HFS that has the shared library extended attribute set is loaded as a system-shared library program. To prevent misuse of system-shared libraries, profile BPX.FILEATTR.SHARELIB in class FACILITY controls who can set this attribute.

These are sample RACF definitions for all three BPX.FILEATTR profiles in class FACILITY:

```
RDEFINE FACILITY BPX.FILEATTR.APF UACC(NONE)
PERMIT BPX.FILEATTR.APF CLASS(FACILITY) USER(userid) ACCESS(UPDATE)
RDEFINE FACILITY BPX.FILEATTR.PROGCTL UACC(NONE)
PERMIT BPX.FILEATTR.PROGCTL CLASS(FACILITY) USER(userid) ACCESS(UPDATE)
RDEFINE FACILITY BPX.FILEATTR.SHARELIB UACC(NONE)
PERMIT BPX.FILEATTR.PROGCTL CLASS(FACILITY) USER(userid) ACCESS(UPDATE)
SETROPTS CLASS(FACILITY) REFRESH
```

3.3.13.5 BPX.SERVER

This FACILITY class profile has been explained in detail in 3.3.2.3, "BPX.SERVER authority" on page 65.

3.3.13.6 BPX.SMF

In the non-UNIX OS/390 environment, a program needs to be APF authorized or in supervisor state or system key to be able to cut an SMF record. This protects the System Management Facilities from misuse by rogue programs. Such programs might purposefully create large amounts of SMF records to fill up all SMF recording data sets and either attempt a denial-of-service attack on the system (if an SMF full conditions halts the system) or to cover up illicit activity that might otherwise have been logged to SMF.

In OS/390 UNIX, it is not appropriate to require daemons that need to cut SMF records to be APF authorized. To minimize the exposure that rogue UNIX programs might attack System Management Facilities, READ authority to FACILITY class profile BPX.SMF is required for OS/390 UNIX to be able to cut SMF records.

3.3.13.7 BPX.STOR.SWAP

This controls which users can make address spaces nonswappable. Users permitted with at least READ access to BPX.STOR.SWAP can invoke the __mlockall() function to make their address spaces either nonswappable or swappable.

When an application makes an address space nonswappable, it may cause additional real storage in the system to be converted to preferred storage.

Because preferred storage cannot be configured offline, using this service can reduce the installation's ability to reconfigure storage in the future. Any application using this service should warn the customer about this side effect in its installation documentation.

3.3.13.8 BPX.WLMSERVER

This controls access to the WLM server functions _server_init() and _server_pwu(). It also controls access to these C language WLM interfaces:

- QuerySchEnv()
- CheckSchEnv()
- DisconnectServer()
- DeleteWorkUnit()
- JoinWorkUnit()
- LeaveWorkUnit()
- ConnectWorkMgr()
- CreateWorkUnit()
- ContinueWorkUnit()

A server application with read permission to this FACILITY class profile can use the server functions, as well as the WLM C language functions, to create and manage work requests.

3.3.14 Programs in the Hierarchical File System

The implementation of UNIX services in OS/390 introduces a new location for executable programs: the Hierarchical File System. In UNIX terms, a program is an executable file, or for short, an executable. In MVS, the non-UNIX OS/390 environment, we work with load modules that reside in load libraries.

A search for an MVS load module is performed in a predefined sequence of steps that include already loaded modules, STEPLIB or JOBLIB libraries, LPA resident modules, and modules that reside in libraries that are included in the system link list as specified in the LNKLSTxx member of SYS1.PARMLIB. There are techniques in standard MVS to work with APF authorized load modules, which are programs that use certain authorized functions in an MVS system, for example, changing the MVS identity of an address space or a task in an address space with the RACROUTE REQUEST=VERIFY macro.

As the HFS is a UNIX-style file system, OS/390 UNIX uses UNIX-style techniques when searching for executable files in the HFS. The order of search is specified via an environment variable called PATH. Because every user may set his or her own PATH variable, it is relatively simple for a user to specify that the user's own programs should execute instead of system programs or commands. The user can, for example, assign the following value to the PATH environment variable:

```
.:/bin:/u/hdm:/usr/lpp/internet/www/Admin
```

The leading dot (.) specifies that the search for an executable file should begin in the current directory.

If the executable file is not found in the current directory, then the search continues through the /bin, /u/hdm, and finally /usr/lpp/internet/www/Admin directories.

Be aware

A superuser or a user with authority to BPX.SUPERUSER who has entered superuser mode should not use a PATH that contains the current directory. If it is absolutely necessary to use the current directory, it should be placed at the end of the search path.

As you can see from the above example, it is quite simple for a user to execute a different program with the same name in place of a system command or system program.

Processes that use security-critical functions need to run in a controlled program environment (for details see 3.3.2.1, “The controlled program environment” on page 64) to make sure that all programs in the address space are loaded from a trusted source.

3.3.14.1 The sticky bit

The sticky bit was invented with early versions of UNIX where the process of loading an executable file into memory was very resource consuming. By setting the sticky bit in the file system for the executable file, the executable file was kept in storage even after it had finished executing, ready to be used by other users in the UNIX system. It was kept in storage until the system was shut down. Nowadays systems are not as constrained in storage as they used to be, so the original meaning of the sticky bit has vanished and it is now often used for other purposes.

UNIX System Services uses the sticky bit to bypass loading of an executable from the Hierarchical File System. If an executable file has the sticky bit turned on as shown below, the executable file in the HFS will not be used, but OS/390 will instead turn to the standard MVS search order to look for a copy of the executable file in an MVS load library.

```
/usr/lpp/internet: >ls -l
total 40
-rw-r--r-- 2 WEBADM IMWEB envvars
-rw-r--r-- 2 WEBADM IMWEB httpd_msg.cat
drwxr-xr-x 2 WEBADM IMWEB IBM
-rwxr--r-T 2 WEBADM IMWEB IMWCGIBN
Drwxr-xr-x 2 WEBADM IMWEB logs
Drwxr-xr-x 3 WEBADM IMWEB Samples
Drwxr-xr-x 10 WEBADM IMWEB ServerRoot
/usr/lpp/internet: >
```

Figure 35. Sticky bit

Above is an example of a sticky bit setting for an executable file. The T in the file security packet for the file called IMWCGIBN indicates that the sticky bit has been set.

When the OS/390 UNIX program loader finds an executable file with the sticky bit on, it does not load the program from the Hierarchical File System, but passes the load request on to the MVS contents supervisor, which then searches for the requested module in one of the following libraries:

1. STEPLIB, if it has been allocated to the current process
2. The LPA
3. Link libraries as they are defined in the LNKLSTxx or PROGxx members of SYS1.PARMLIB

If a process needs to run in a controlled program environment, all STEPLIB and LNKLSTxx libraries where modules for this address space are loaded from have to be defined in class PROGRAM. The LPA is always considered a controlled environment, so LPA libraries do not need to be defined in class PROGRAM.

3.3.14.2 Extended attributes

Some files in the Hierarchical File System need special permissions over and above those normally used for UNIX files, such as attributes that:

- Mark an executable file as program controlled
- Mark an executable file as APF authorized
- Allow a program to be loaded into a shared address space
- Mark a program as a system-shared library object

In the Hierarchical File System, *extended attributes* can be set to define these special permissions with the `extattr` shell command.

- `extattr +p` marks an executable file as program controlled. See 3.3.13.3, “BPX.FILEATTR.PROGCTL” on page 80 for the authority needed to set this attribute.

- `extattr +a` marks an executable file as APF authorized. Note that the program needs to be linked with AC=1 if it is to be invoked as a job step program. See 3.3.13.2, “BPX.FILEATTR.APF” on page 80 for the authority needed to set this attribute.
- `extattr +l` marks an executable file as a system-shared library object. The program will be loaded into the kernel-shared library region data space. See 3.3.13.4, “BPX.FILEATTR.SHARELIB” on page 80 for the authority needed to set this attribute.
- `extattr +s` marks an executable file as a program that may be loaded into a shared address space. Whether the program will actually be loaded into a shared address space depends on the setting of the environment variable _BPX_SHAREAS.

To specify any of the attributes, the issuer of the command needs to be the file owner or a superuser in addition to any RACF authority that may be needed. As soon as an executable file is modified, it will lose all its extended attributes.

The following example shows how the program-controlled attribute for a file can be set:

```
extattr +p /usr/sbin/inetd
```

3.3.15 OS/390 UNIX kernel address space

The OS/390 UNIX kernel address space is the owner of every process it forks on behalf of /etc/rc definitions. That means the user ID of the kernel address space must be defined to BPX.DAEMON with READ access.

In our sample setup, the kernel user ID is OMVSKERN. If you start InetD and syslogd via the /etc/rc shell script, they will execute under the OMVSKERN user ID.

The output of the `D OMVS,A=ALL` command for InetD looks like:

```
OMVSKERN INETD1   0076  687865905          1 1FI   14.48.00      1.637
  LATCHWAITPID=         0 CMD=/usr/sbin/inetd /etc/inetd.conf
```

3.3.16 OS/390 UNIX security considerations for TCP/IP

The following topics discuss security considerations for TCP/IP address spaces, operator commands, and a number of TCP/IP applications.

3.3.16.1 TCP/IP address spaces

As mentioned before, TCP/IP is an OS/390 UNIX application itself. Consequently, the user ID a TCP/IP address space is running under needs to have a UID associated to it, and it must be UID(0). The following example shows how to assign an OMVS segment to an existing user ID for a TCP/IP address space. This user ID should also be a protected user ID (a user ID that has no password and cannot be used to log on to any interactive service):

```
ALU tcpip_user NOPASSWORD OMVS(UID(0) HOME(/) PGM(/bin/sh))
```

The TCP/IP address space is started as a started procedure, so a profile in class STARTED is required for it to pick up its user ID. The following example shows a definition:

```
RDEFINE STARTED TCPIP.** STDATA(USER(TCPIP1) GROUP(SYS1) TRUSTED(NO))
SETROPTS RACLIST(STARTED) REFRESH
```

3.3.16.2 VARY TCP/IP command

The `VARY TCPIP` command is used to stop and restart TCP/IP address spaces, make changes to system operation and network configuration, start and stop devices, trace data, and control certain TCP/IP applications.

If the installation has activated class OPERCMDS in RACF, profiles should be defined to control the use of the `VARY TCPIP` command. The resource names shown in Table 5 are in use.

Table 5. RACF OPERCMDS resources for the VARY.TCPIP command

VARY TCPIP command options	RACF resources
VARY TCPIP,,DATTRACE	MVS.VARY.TCPIP.DATTRACE
VARY TCPIP,,DROP	MVS.VARY.TCPIP.DROP
VARY TCPIP,,OBEYFILE	MVS.VARY.TCPIP.OBEYFILE
VARY TCPIP,,PKTTRACE	MVS.VARY.TCPIP.PKTTRACE
VARY TCPIP,,START	MVS.VARY.TCPIP.START
VARY TCPIP,,STOP	MVS.VARY.TCPIP.STOP

The following example shows the definition of profiles for DROP and OBEYFILE. Users should be authorized with access level CONTROL. The class needs to be refreshed after the changes.

```
RDEFINE OPERCMDS (MVS.VARY.TCPIP.DROP) UACC(NONE)
RDEFINE OPERCMDS (MVS.VARY.TCPIP.OBEYFILE) UACC(NONE)
PERMIT MVS.VARY.TCPIP.DROP ACCESS(CONTROL) CLASS(OPERCMDS) ID(uid)
PERMIT MVS.VARY.TCPIP.OBEYFILE ACCESS(CONTROL) CLASS(OPERCMDS) ID(uid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

In many cases, it will not be necessary to define individual profiles because the different commands will be issued by the same operators. In this case it is adequate to define a single generic profile that covers all commands. The following example shows this; CONTROL access is given to group NETOPER:

```
RDEFINE OPERCMDS (MVS.VARY.TCPIP.**) UACC(NONE)
PERMIT MVS.VARY.TCPIP.** ACCESS(CONTROL) CLASS(OPERCMDS) ID(NETOPER)
```

3.3.17 IBM-supplied daemons

A number of daemon programs are supplied with OS/390. These daemons require some security considerations, which we will discuss in the following sections. All the daemons discussed here are OS/390 UNIX programs so the user ID they are running under must have a UID assigned. Please note that some daemons also have an MVS, non-UNIX equivalent which will not be discussed here.

OS/390 UNIX supplies these daemons:

- InetD, the Internet daemon
- rlogind, the remote login daemon
- cron, the batch scheduler
- lm, the Communications Server login monitor
- uucpd, the UUCP daemon

The inetd, rlogind, cron, lm, and uucpd programs reside in the HFS directory /usr/sbin. They have the sticky bit turned on (see 3.3.14.1, "The sticky bit" on page 82).

3.3.17.1 InetD

InetD is a generic listener program that is responsible for starting server programs at the client's request. The operation of InetD is controlled through its configuration file /etc/inetd.conf.

If InetD is forked by /etc/rc, it will inherit the user ID of the BPXOINIT process which usually should be OMVSKERN. This user ID is a superuser and has access to BPX.DAEMON in class FACILITY. If InetD is started from the UNIX shell, it needs to be started by a superuser.

If InetD is started as a started procedure (starting with OS/390 V2R7, program BPXBATCH must be used since InetD is no longer a module in SYS1.LINKLIB), a profile in class STARTED is needed to assign a user ID to InetD.

When InetD is running and receives a request for a connection, it processes that request according to the port number over which it received the request. If, for example, a user tries to log in from a remote system using rlogin, InetD receives the request at port number 513. The correlation between port numbers and service names as they are specified in /etc/inetd.conf is established through your /etc/services file, where you define which port number is associated with a service name. After having received the connection request, InetD issues a fork(), and the forked process issues an exec() to start the requested server program (in this case, the rlogind program) which then runs as the server.

The user ID under which the new process will initially start is defined in /etc/inetd.conf. The figure below shows the definitions of the services in /etc/inetd.conf. We have extracted the Telnet and rlogin definition lines:

```
=====================================================================
service | socket | protocol | wait/ | user | server  | server program
name    | type   |          | nowait|      | program |   arguments
=====================================================================
otelnet  stream tcp nowait TCPIP3   /usr/sbin/otelnetd otelnetd -l
login    stream tcp nowait OMVSKERN /usr/sbin/rlogind rlogind -m
```

In this example, we have assigned the user ID OMVSKERN to rlogind and TCPIP3 to telnetd.

3.3.17.2 telnetd, RSHD, REXECD and rlogind

The most often-used services that are managed by InetD are Telnet, RSH, REXEC, and rlogin. We will briefly describe one service here, the Telnetd daemon.

InetD listens for Telnet connection requests usually on port 23, although a different port can be used. Immediately after such a request arrives, InetD uses the fork() service to create a child process and starts the Telnetd server program as specified in /etc/inetd.conf.

The Telnetd server program enters the initial session setup window with the Telnet client and requests a user ID and password from the client end user. In this phase, the user ID and password are checked. After these are accepted by the current phase of Telnetd, Telnetd restarts itself via a new exec() call with a new set of parameters and in addition, a shell process is started.

Both the Telnetd process and the shell process execute under the end user ID, and not the original daemon user ID. The shell process has the Telnetd process as parent, the Telnetd process has the InetD process as parent, and InetD in turn has the highest level process of all, the BPXOINIT process, as parent.

Processing for the other interactive services, such as RSHD, REXECD, and rlogind, is similar.

All servers need to run under the superuser authority. Furthermore, RSHD and REXECD require daemon authority to change the user identity of its child processes. If they are executed without daemon authority, a rsh/rexec client program will fail and the error messages similar to the following will show up in the MVS console:

```
ICH408I USER(TCPIP3  ) GROUP(OMVSGRP ) NAME(TCPIP TASKS         )
  BPX.DAEMON CL(FACILITY)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

Because all those servers are invoked by InetD, the user ID associated with InetD should have the access to BPX.DAEMON in most installations. The user ID OMVSKERN, which is the default user ID associated with OS/390 UNIX kernel, would be recommended.

If you choose to have a daemon program run with a user ID other than OMVSKERN, the selected user ID needs to be defined with UID(0) and it needs READ access to profile BPX.DAEMON in class FACILITY.

3.3.17.3 Routing daemons

On OS/390 IP there are two routing daemons:

- ORouteD

  ORouteD is an IP routing daemon that implements RIP-1 and RIP-2. It creates and maintains network routing tables. ORouteD provides an alternative to static routing protocols. ORouteD determines if a new route has been established or if a route is temporarily unavailable.

1. OpenEdition Multiprotocol Route Application (OMPROUTE)

   OMPROUTE is an IP routing daemon that supports RIP-1, RIP-2 and OSPF protocols. OMPROUTE was introduced in eNetwork Communications Server for OS/390 V2R6 IP and enhanced in eNetwork Communications Server for OS/390 V2R7 IP as the recommended routing daemon application for OS/390 IP. It provides the technology to determine the most efficient route for use in the autonomous system. OSPF is the preferred protocol in OMPROUTE.

Note: ORouteD and OMPROUTE will not run on the same TCP/IP stack.

RACF control of who can start OMPROUTE or ORouteD was integrated into CS for OS/390 V2R8 IP. If the RACF profile is not defined, no additional security checks are performed.

To enable ORouteD and OMPROUTE RACF authorization for a particular user ID, enter the following commands with the ID of the user or with the ID of the started task to be authorized:

```
RDEFINE OPERCMDS (MVS.ROUTEMGR.OMPROUTE) UACC(NONE)
PERMIT MVS.ROUTEMGR.OMPROUTE ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
RDEFINE OPERCMDS (MVS.ROUTEMGR.OROUTED) UACC(NONE)
PERMIT MVS.ROUTEMGR.OROUTED ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

If neither the user nor the started task is RACF authorized for ORouteD, the following message is displayed:

```
EZZ5020E  "User not RACF authorized to start OE RouteD"
```

If neither the user nor the started task is RACF authorized for OMPROUTE, the following message is displayed:

```
EZZ5020E  "User not RACF authorized to start OMPROUTE"
```

3.3.18 MVS sockets server applications

In CS for OS/390 Release 8, the only server that ran in a traditional MVS environment was the TN3270 server, which ran within the TCP/IP address space. All others needed an OMVS segment defined in the RACF database, except those that use the PASCAL API. The PASCAL servers are LPD, SMTP and DNS.

The following application servers are covered in Chapter 5, "TCP/IP application security" on page 101:

- IBM HTTP Server
- TN3270 server
- FTP server
- TFTP server
- NFS server
- UNIX rlogin/rsh/rexecd
- SNMP

3.3.19 Summary

The following table gives a summary of all the definitions you have to create to run UNIX System Services servers with MVS-like security.

Table 6. Overview of processes, UIDs and RACF FACILITY class profiles

Process/Job	UID	BPX.SERVER access	BPX.DAEMON access
OMVS	0	NONE	READ
TCP/IP system address space	0	NONE	NONE
InetD	0	NONE	READ

Process/Job	**UID**	**BPX.SERVER access**	**BPX.DAEMON access**
syslogd	0	NONE	NONE
FTPD	0	NONE	NONE
Client User	Not 0	NONE	NONE
Web Server	0	READ	NONE
Web User	Web User not 0, not UID of administrator, not in same group	NONE	NONE

Note: Please remember that as soon as you have defined BPX.DAEMON, program control must have been set up correctly and must be active. If not, all server requests are going to fail with various error messages. Whenever something does not work as expected, make it a rule to look at your OS/390 system log to see if there are any security violation messages that might indicate a security definition problem.

You may also wish to investigate a couple of sample jobs that will assist you in performing some of these tasks. They are:

- EZARACF, shipped in hlq.SEZAINST from CS for OS/390 V2R8, contains sample RACF commands for most TCP/IP-related security functions.
- BPXISEC1, shipped in SYS1.SAMPLIB as part of UNIX System Services from OS/390 V2R6, contains sample commands for most OS/390 UNIX-related RACF commands.

Chapter 4. TCP/IP stack security

Several security features relating to the TCP/IP stack itself exist in z/OS and CS for OS/390 IP. They are:

- Network access control
- Stack access control
- Port access control (enhanced in OS/390 V2R10)
- Stack affinity

All of these features can be used, either separately or together, to enhance the security of CS for OS/390 IP. This chapter provides an overview of these features.

4.1 Network access control

Network access control is, as its name suggests, a method of controlling access to a specific network or subnetwork. However, it does not control which networks TCP/IP is able to receive data from; it controls which networks are reachable on an outbound (send data to) basis. This is a subtle distinction and not one that would generally make a difference in the security world. For example, let’s say we have resource A, which is not permitted outbound access to network a.b.c.d. If a network attack were to occur from network a.b.c.d, there is nothing to stop or disallow this inbound request. Thus, a denial-of-service SYN attack could still occur from a.b.c.d. However, under no circumstances could network a.b.c.d gain any return information from resource A. Of course, if another resource, B, is permitted to this network, this is an entirely different situation and outbound communication could occur.

Network access control is a granular approach to network access and a considerably powerful one. Some installations might want to permit only a few users to have access to the Internet, while other users have only local access. In other situations, an installation might want only the SMTP server (or sendmail) to have access to the Internet. The user ID that SMTP runs under could be permitted, while all end users’ user IDs would have intranet access only. There are numerous possibilities.

4.1.1 How network access control works

CS for OS/390 V2R10 IP allows an installation to disallow the access of certain users to certain networks. It can disallow communication between a particular user or group of users on the OS/390 host and a particular network by specifically disallowing the users the ability to send IP data packets to the destination network or host. TCP/IP stacks in CS for OS/390 V2R10 IP provide the ability to associate a destination network with a Security Access Facilities (SAF) resource. During TCP connection establishment and UDP/TCP/RAW sends, it determines if a particular user has the authority to send to that destination.

Figure 36 is an excerpt from a TCP/IP profile data set.

```
PROFILE.TCPIP:
NETACCESS
192.168.100.0/24 ADM1
9.100.11.1/32 WS01
172.16.232.39/32 RA39
ENDNETACCESS
```

Figure 36. Example of NETACCESS statement

Focusing on the first statement, we have a subnet work on the first three octets (24 bits) giving a subnetwork of 192.168.100.0. If an end user wanted to be able to reach the 192.168.100.0 subnetwork, this user's user ID would need to be permitted to the ADM1 SAF resource (SERVAUTH class).

Some sample RACF commands to complete the NETACCESS control are illustrated in Figure 37. Note that the complete profile name used for the RACF definition is of the format:

```
EZB.NETACCESS.systemname.stackname.resourcename
```

In this example, the resources are defined with access `NONE`, which means all users must be explicitly permitted in order to access any of these networks. The only user that would be able to send data to the 192.168.100.0 subnetwork would be USER2. This is a direct result of USER2 having been permitted to the resource:

```
EZB.NETACCESS.RA03.TCPIPB.ADM1
```

```
SETROPTS CLASSACT(SERVAUTH) REFRESH
RDEFINE SERVAUTH (EZB.NETACCESS.RA03.TCPIPB.ADM1) UACC(NONE)
RDEFINE SERVAUTH (EZB.NETACCESS.RA03.TCPIPB.RA39) UACC(NONE)
RDEFINE SERVAUTH (EZB.NETACCESS.RA03.TCPIPB.WS01) UACC(NONE)
PERMIT EZB.NETACCESS.RA03.TCPIPB.WS01 ACCESS(READ) CLASS(SERVAUTH) ID(USER1)
PERMIT EZB.NETACCESS.RA03.TCPIPB.WS01 ACCESS(READ) CLASS(SERVAUTH) ID(USER3)
PERMIT EZB.NETACCESS.RA03.TCPIPB.ADM1 ACCESS(READ) CLASS(SERVAUTH) ID(USER2)
PERMIT EZB.NETACCESS.RA03.TCPIPB.RA39 ACCESS(READ) CLASS(SERVAUTH) ID(USER2)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Figure 37. Example of RACF definitions to use NETACCESS control

If a network is not identified in a NETACCESS statement, all TCP/IP users will by default be permitted access.

Obviously, end users are not the only users of TCP/IP to be affected by NETACCESS resources. Any IP applications (servers) that run under their own user ID would need SAF access to the resource for the desired networks. For example, a server such as the FTP daemon would need to be permitted to any networks that are expected to be able to make an FTP connection to the FTP daemon.

There is another subtlety to be considered. Imagine you have a user out on the network that wants to FTP into your FTP server. The user ID that this user logs on with must have been permitted to NETACCESS.

User IDs associated with an IP application or an end user logging in from a remote location and attempting to access a network (and the corresponding SAF resource) would be presented with a RACF error message when attempting to connect.

In Figure 38, the network control as defined by the commands above is illustrated graphically.

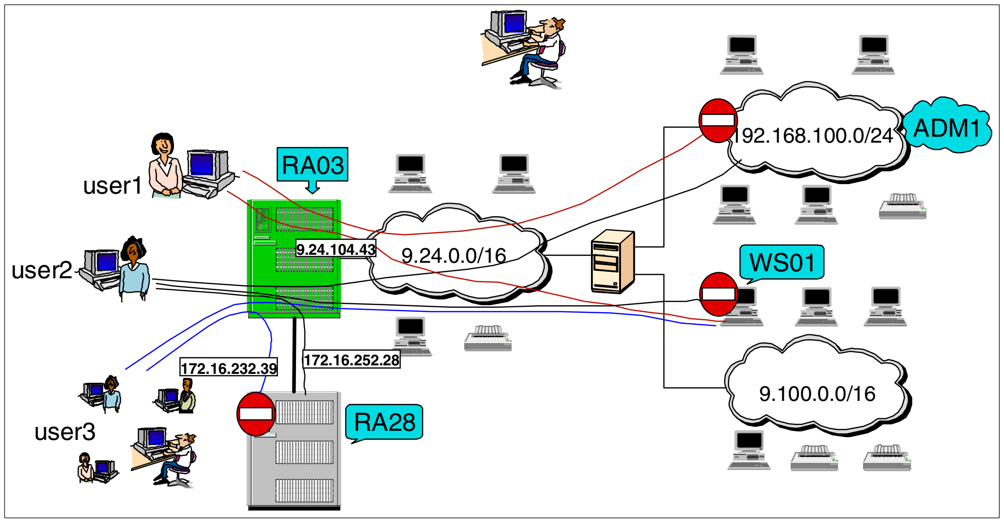


Figure 38. Example of network access control

4.2 Stack access control

There are situations when, for any number of reasons, an installation might need to run more than one TCP/IP stack within a single image. One security issue that arises from this scenario is the difficulty of controlling which users are allowed to use which TCP/IP stack. Stack access was formally controlled only via the TCPIPJOBNAME parameter within a TCPIP.DATA (/etc/resolv.conf) file. It was trivial for a user to either change one of these files or allocate their own modified copy, giving them access to any TCP/IP stack within the image.

As of OS/390 V2R10, an installation can prevent users from getting access to any or all TCP/IP stacks through a TCP or UDP socket. To minimize performance impacts, the stack access check is performed only on the socket() call. Additionally, the security checking will be bypassed for socket() calls originating in the TCP/IP, UNIX System Services or VTAM address spaces. If SERVAUTH stack access profiles are enabled, all TCP/IP code must be running under a user ID that is permitted to the SAF resource:

```
EZB.STACKACCESS.sysname.tcpipname
```

where:

- `sysname` is the name of the MVS image in the sysplex
- `tcpipname` is the TCP/IP job name to be controlled

If the system administrator does not define the SAF resource to protect the stack, stack access control is not performed and all users have access to the stack. If there are multiple stacks in the OS/390 system, each stack can be enabled for stack access or not.

Figure 39 is an example of stack access control in action.

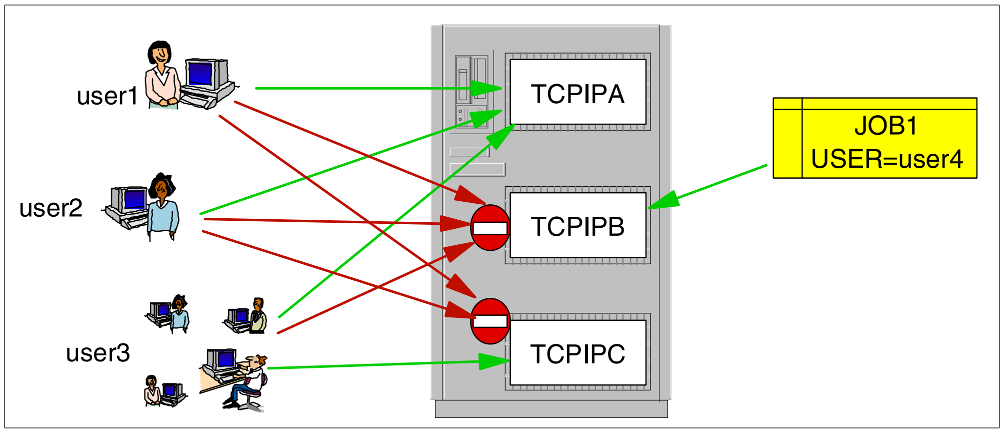


Figure 39. Stack access control

User IDs that are not explicitly SAF authorized will be unable to access a stack. Although this example shows end users, the principle applies equally to user IDs associated with any TCP/IP application.

A word of caution

AnyNet (IP sockets over SNA) cannot be protected via the stack access resource definition. In addition, IP services provided via non-IBM companies may not be protected (OEM TCP/IP stacks).

4.3 Port access control

There are two improvements to port access control available with OS/390 V2R10:

- SAF resources for port access control
- Reserved keyword

These items have a significant impact on improving the security of a CS for OS/390 IP implementation.

4.3.1 SAF resource for port access control

Port access control allows a CS for OS/390 V2R10 IP installation the ability to use SAF resource control to authorize port numbers to applications. A SAF resource name is coded on the PORT or PORTRANGE statement. Any application attempting to use the specified port number requires SAF access to the resource.

The SAF keyword can be specified along with all other valid options on PORT and PORTRANGE statements. Matching an application to a port reservation entry is done by the job name. The RACF profile would be defined as:

```
EZB.PORTACCESS.sysname.tcpproc.saf_resname
```

where:

- `sysname` identifies the OS/390 system name
- `stackname` identifies the TCP/IP stack name
- `saf_resname` identifies the SAF resource name defined in the PORT (or PORTRANGE) statement in TCPIP.PROFILE

When a SAF resource name is used, the binding process's user ID must be permitted to that resource by the security product.

In order to facilitate the usage of SAF resource port access, a new wildcard capability has been added to the TCP/IP profile PORT statement. The wildcard is, as one would expect, a method of allowing any application to access a port. By itself, the wildcard would obviously be of no benefit.

To get a better idea of how SAF-based port access functions, let’s look at the example in Figure 40.

```
PORT
   20 TCP * SAF FTP20B
   21 TCP * SAF FTP21B
   23 TCP INTCLIEN
  600 TCP * SAF MYAPP
```

Figure 40. Example of PORT statement with SAF resource control

In this example, the TCP port 600 reserved with the SAF keyword would have the following SERVAUTH resources (assuming the system name is MVS1 and the TCP/IP stack name is TCPIP):

```
EZB.PORTACCESS.MVS1.TCPIP.MYAPP
```

This form of port reservation can be used when a reserved low port must be accessed by many potential users via a client program that is not APF-authorized. All users needing the ability to run this program could be permitted to this RACF resource.

The port access function for FTP is a little tricky since the bind to the data connection port (20) is done under the identity of the end user not the FTP daemon. Therefore, if the PORT statements for port 20 is configured as shown in Figure 40, all end users who could potentially perform FTP need to be permitted to the port 20 RACF resource, EZB.PORTACCESS.MVS1.TCPIP.FTP20B. In the meantime, the FTP daemon is the only user that should access the port 21 RACF resource, EZB.PORTACCESS.MVS1.TCPIP.FTP21B. (The FTP server could function with access only to FTP21B, but data transfers, including DIR and LS commands, would fail with a SAF error.)

Now that all users are able to bind to port 20 because they have been authorized by RACF, this opens up the possibility that they could bind to port 20 from some other program. To prevent this security exposure, we recommend that you use RESTRICTLOWPORTS for port 20, instead of RACF controls. This combination says that you do not have to be authorized through RACF to the port but you do have to be APF authorized or superuser to a low port that the FTP server uses (even though we are running under the end user's identity).

Recall that use of TCPCONFIG RESTRICTLOWPORTS and UDPCONFIG RESTRICTLOWPORTS is encouraged to enhance security. Then low ports (<1024) can only be bound when at least one of the following is true:

- The bind issued from superuser (UID 0) user ID
- The bind issued from APF (AC=1) application
- The port is reserved for application by job name, which may include *, OMVS or TSO user ID
- If a SAF resource name is used, the binding process's user ID must be permitted to the resource by the security product

Figure 40 illustrates port access control used for the FTP and TN3270 servers. It also shows an example of a RESERVED port 111 (see 4.3.2, "Reserved keyword" on page 97).

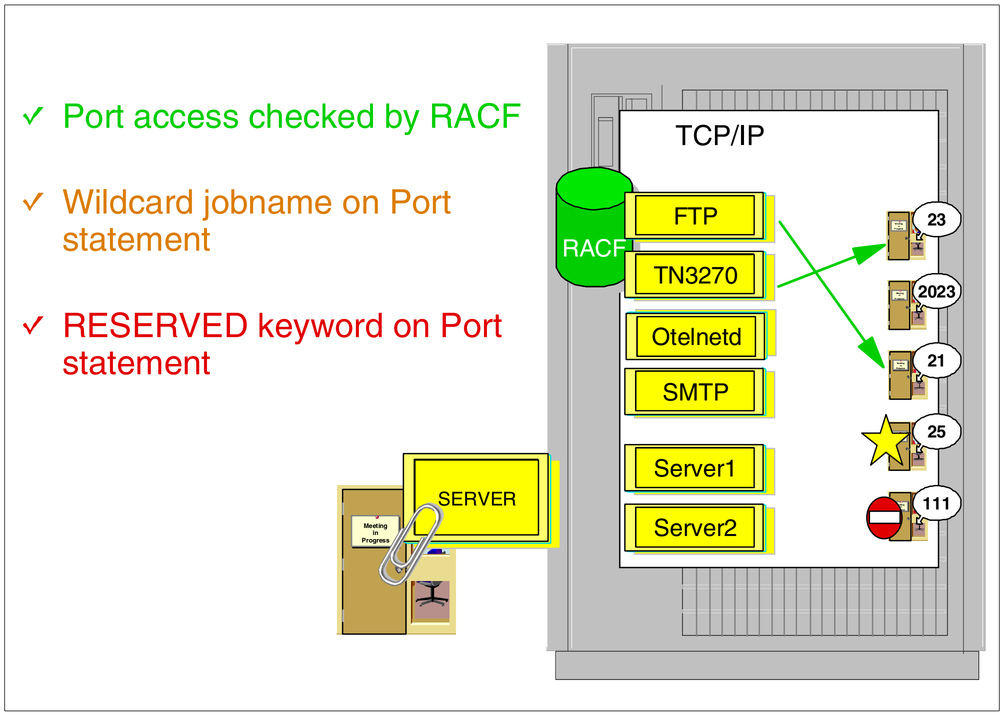

✓ Port access checked by RACF
✓ Wildcard jobname on Port statement
✓ RESERVED keyword on Port statement
TCP/IP
RACF
FTP
TN3270
Otelnetd
SMTP
Server1
Server2
SERVER
23
2023
21
25
111

Figure 41. Port access control

Formerly, the only protection available for a port was to reserve it for a job name (started task name). It was conceivable that a sysprog hacker could run a job using the same name to the detriment of the OS/390 IP host. When using SAF resource control, only the user ID for FTPD itself would be allowed access to these ports. This is a much more difficult hurdle to leap for the sysprog hacker.

Note

OS/390 V2R10, LPR, LPRM, LPQ and RSH all run as APF-authorized programs. This means that even when TCPCONFIG RESTRICTLOWPORTS has been coded in the TCP/IP profile data set, these applications will still be able to use the required ports.

4.3.2 Reserved keyword

Protection of local ports has always been possible with CS for OS/390 IP. This was done by assigning a “secret” job name to a `PORT` or `PORTRANGE` statement. The exposure of this method is obvious: if someone determined this job name, a port could be compromised.

In CS for OS/390 V2R10 IP a new keyword, `RESERVED`, was added. The `RESERVED` keyword will shut down a port from being used by any job name at all. The keyword can also be specified on the PORTRANGE statement. This readily allows an installation to clamp down very tightly on usage of ports if such control is desired.

4.4 Transport affinity

The OS/390 platform (that is, an MVS image) is capable of being configured to support more than one concurrent instance of TCP/IP. In order to support more than one TCP/IP stack, the Physical File System must be configured to support Common INET (C-INET).

The C-INET environment, however, has some subtle implications for the certain IP applications. Some servers function as generic applications. What this means is that when the server sets up a listen() socket call, this listen is effective across all active TCP/IP stacks within the MVS image.

The FTP daemon is one such application. For example, if we start the FTP daemon and there are three active TCP/IP stacks, all three stacks will be able to handle incoming FTP connection requests. See Figure 42 for a pictorial representation.

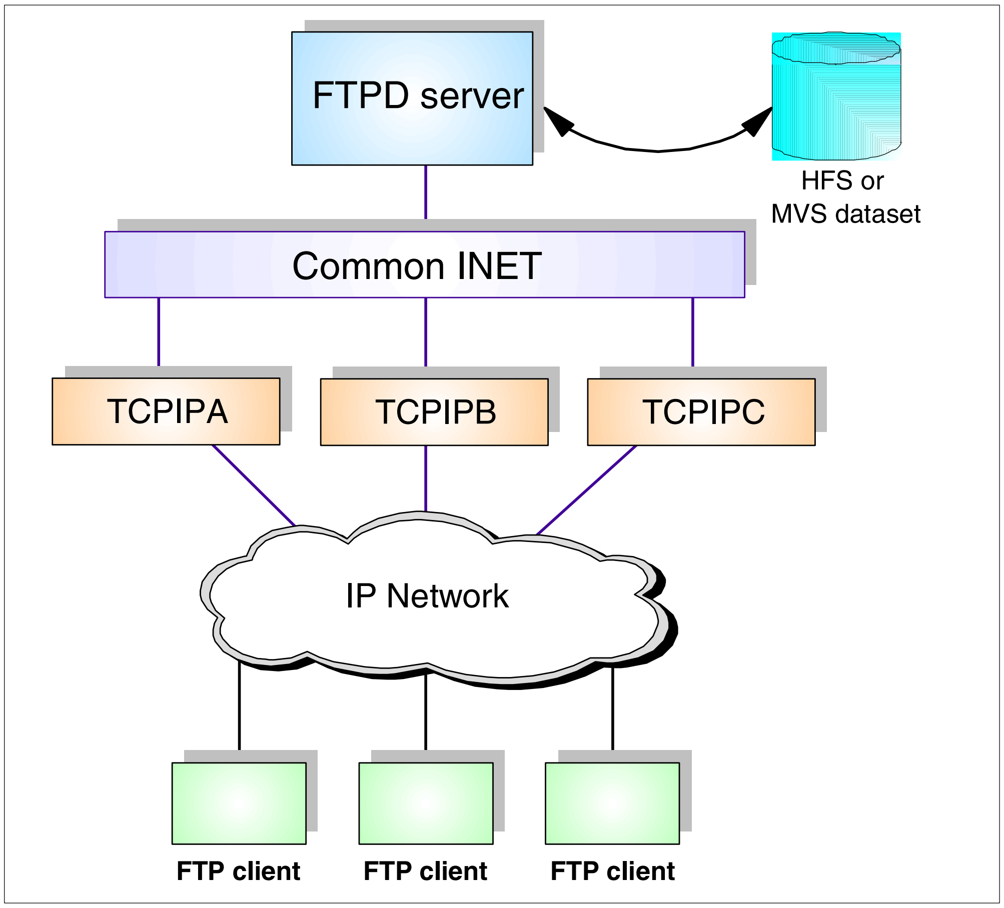


Figure 42. FTP server without affinity

Note that pointing to a particular TCPIP.DATA file and a TCPIPJOBNAME statement usually (see clarification below) does not provide any relief for a generic application. Any active stack can accept an incoming request for a generic server and pass it via C-INET to the appropriate application. This could be a security exposure for customers who do not want an application's services available on all TCP/IP stacks. Or, customers may want FTPA connected to, say, TCPIPA and FTPB connected to TCPIPB.

Clarification

Indirectly, an application can use the __iptcpn() C language function call to determine the TCPIPJOBNAME and then use setibmopt() to bind to the transport.

The assumption at this stage is that a customer is using STACKACCESS (see 4.2, "Stack access control" on page 93) to enforce which stack a generic application is allowed to use. The problem is, in a C-INET environment, a generic server essentially attempts to access all active stacks. If STACKACCESS isn't enabled for the application's RACF user ID to access all of these stacks, a RACF error will occur. Some method is required to ensure that a server is directed to connect to a specific instance of TCP/IP. The mechanism to enable this is referred to as *transport affinity* (sometimes, the term *stack affinity* is also used).

What, exactly, is a generic application? A generic application is one which issues a bind to inaddr_any. When a bind to inaddr_any is issued, the transport provider can be any running instance of TCP/IP. Of course, if an application issues a bind to a specific IP address, or uses setibmopt() to choose a transport provider, then the application is no longer considered generic.

For some installations, a generic application may be exactly the desired effect. However, from a security perspective, this is something we would want to control. There are two ways this can be controlled.

Transport affinity can be set via the ENVAR environment variable _BPXK_SETIBMOPT_TRANSPORT for C language and POSIX-based applications. This variable should be set to the job name of the TCP/IP stack to which you want this instance of the application to bind. Continuing our FTP example, the FTP server with this variable set will function as illustrated in Figure 43.

Essentially, using _BPXK_SETIBMOPT_TRANSPORT causes the LE run-time environment to issue a setibmopt() call on behalf of the executing program.

For other programs, the BPXTCAFF program (as a job step) can be executed to set transport affinity for an entire address space. For more details on BPXTCAFF (and other transport options), see *OS/390 UNIX System Services Planning*, SC28-1890.

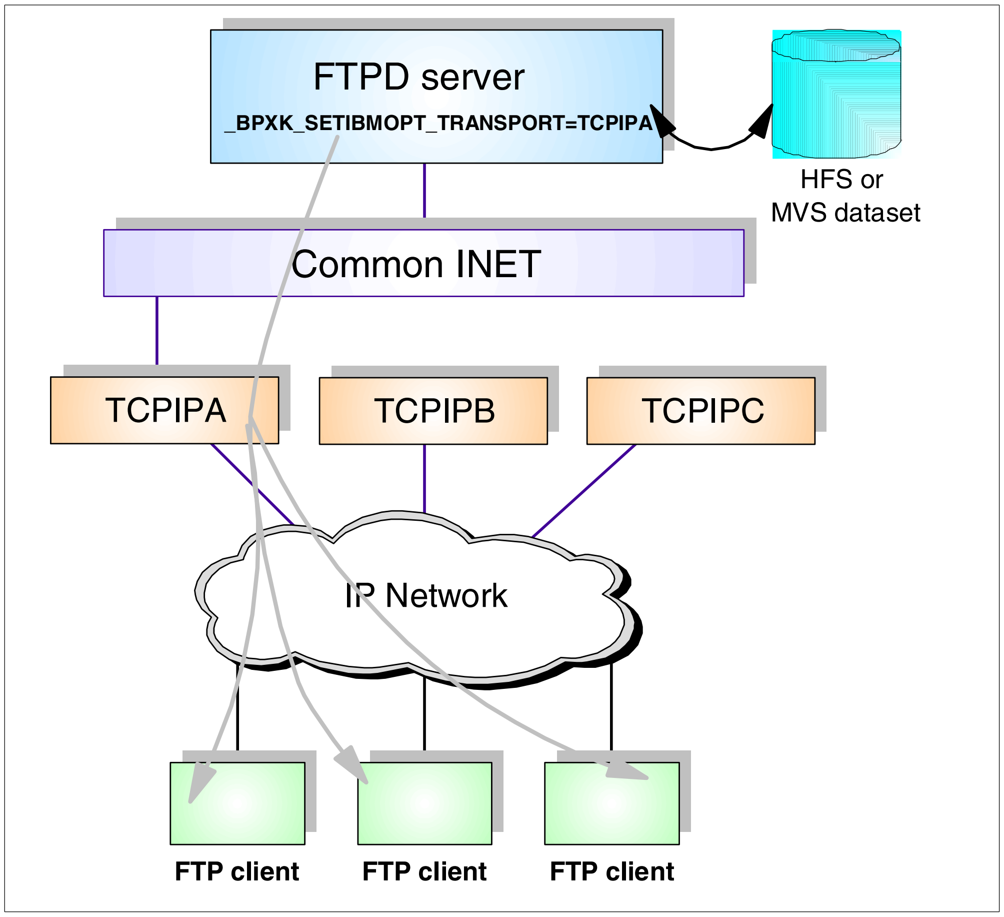


Figure 43. FTP daemon with transport affinity

An example of using this parameter in the FTP started task is as follows:

```
//FTPD PROC MODULE='FTPD',PARMS='TRACE'
//FTPD EXEC PGM=&MODULE,REGION=7M,TIME=NOLIMIT,
//       PARM=('POSIX(ON)ALL31(ON)',
//       'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPA")',
//       '/&PARMS')
```

Using this JCL would instruct the FTP daemon to bind to the TCPIPA stack only.

A second and more granular option for indirectly controlling the transport affinity is the BIND keyword on a PORT statement in the TCP/IP profile data set. For example:

```
PORT
   23 TCP INETD1 BIND 9.67.113.3 ;OE telnet server
```

This statement forces otelnetd, a generic application, to bind to IP address 9.67.113.3, effectively establishing transport affinity. Note that using the BIND keyword is a much more granular control than transport affinity. When binding to a specific IP address, only that IP address can be used as an endpoint of communication. When using setibmopt(), any available IP address in the TCP/IP stack can be a communication endpoint.

The following applications supplied with CS for OS/390 IP are generic applications:

- FTPD
- OS/390 UNIX RSHD
- OS/390 UNIX REXECD
- OS/390 UNIX TELNETD
- OS/390 UNIX SENDMAIL
- OS/390 UNIX POPPER
- TFTPD
- TIMED
- OS/390 UNIX Portmap

4.5 Possible uses of stack security

One of the greatest advantages of stack access control is that it provides an installation with the ability to run multiple independent TCP/IP stacks on a single image. For example, a service provider might run TCPIPA for Company A and TCPIPB for Company B. By using stack access control, user IDs for each company can be prevented from accessing the other company's TCP/IP stack. In the past, prior to OS/390 V2R10, this level of separation could only be achieved by placing the TCP/IP stacks on separate LPARs. Further security can be added using port access and the RESERVED keyword as required.

Although it might not be an ideal situation, it is possible to use stack access to split a single MVS image into a secure TCP/IP stack and a non-secure one. Conceivably, one stack could even be running as a firewall. The connection between the two stacks would be facilitated via an IUTSAMEH type connection.

Chapter 5. TCP/IP application security

In this chapter we talk about the security considerations of the major TCP/IP applications on OS/390. We discuss some concerns and we show how to implement the security functions available on each application. We cover the following applications:

- 5.1, “IBM HTTP Server for OS/390” on page 101
- 5.2, “TN3270 server” on page 136
- 5.3, “UNIX System Services Telnet server” on page 144
- 5.4, “OS/390 FTP server” on page 146
- 5.5, “OS/390 TFTP server” on page 163
- 5.6, “OS/390 NFS server” on page 164
- 5.7, “UNIX System Services rlogind/rshd/rexecd” on page 173
- 5.8, “OS/390 SNMP” on page 177
- 5.9, “OS/390 LDAP server” on page 181
- 5.10, “OS/390 policy agent security” on page 202
- 5.11, “Traffic Regulation Management (TRM) security” on page 224
- 5.12, “OS/390 and z/OS syslog daemon security” on page 232
- 5.13, “Express Logon Feature (ELF)” on page 241
- 5.14, “Routing protocol security” on page 246

5.1 IBM HTTP Server for OS/390

IBM made its first World Wide Web server (Web server) for MVS/ESA available in December 1995. That product was the IBM Internet Connection Server for MVS/ESA. Before becoming IBM HTTP Server, it was also called Domino Go Webserver.

There are still people who cannot believe that S/390 systems could run Web servers, but anybody who takes a more thorough look at the matter should quickly discover the virtues of Web serving with S/390 systems. For most installations, there is significant value in putting Web servers on OS/390. Your S/390 already holds significant amounts of your business data, which could be made available to the people who need it by using Internet technologies. Furthermore, since OS/390 is a critical business system for so many enterprises, it has built-in strengths such as scalability, availability, security, and integrity. These strengths can be applied to Internet work just as they are applied today to transaction processing, very large databases, and batch processing.

In general, Web server security consists of the following elements:

- **Identification and authentication**

  Authentication is the process of verifying the validity of a claimed identity.

- **Access control**

  Assurance that each user or computer that uses the service is permitted to do what the user asks. The term authorization is often used as a synonym for access control, but authorization also includes granting the accessor the right to perform some actions based on access rights.

- **Data confidentiality**

  Sensitive information must not be revealed to parties for which it was not meant. Data confidentiality is often also referred to as privacy.

- **Data integrity**

  Data integrity ensures that the data is not altered, rearranged or truncated in an unauthorized manner.

- **Non-repudiation**

  Assurance that a sender cannot deny being the source of a message and that the recipient cannot deny the receipt of a message.

- **Availability**

  The availability over time, or in terms of response time, of a system. While this is normally not a security-related term, it may become so when dealing with denial-of-service attacks that may intentionally slow down or completely paralyze a system.

- **Security audit**

  A process where an independent party checks the security level of an organization or computer system.

- **Key management**

  Key management deals with the secure generation, distribution, and storage of keys used in cryptography.

IBM HTTP Server for OS/390 provides functions corresponding to all of the above security elements. See Table 7 for a summary.

Table 7. Security functions of IBM HTTP Server for OS/390

Web security element	**Corresponding functions of IBM HTTP Server for OS/390**
Authentication	RACF user ID, password file, client certificate, LDAP definition
Access Control	Protection/Protect/DefProt directive, ACL file, LDAP information
Data Confidentiality	SSL
Data Integrity	SSL
Nonrepudiation	SSL, Association of client certificate with RACF user ID
Availability	Scalable Web server with WLM
Security Audit	Various log files and reporting tools, SNMP subagent
Key Management	GSKKYMAN utility or RACF keyring

In this redbook, we focus on the following security topics related to the IBM HTTP Server for OS/390:

- How to authenticate Web clients
- How to control access from clients
- How to make HTTP communications secure

5.1.1 Server security structure

IBM HTTP Server for OS/390 creates a worker thread within its process (address space) for each client's request. IBM HTTP Server can associate each thread with the identity of the user using a proprietary pthread_security_np() service. Other UNIX systems do not have this concept of thread security.

The pthread_security_np() service enables IBM HTTP Server to establish a more strict and yet more dynamic security environment. A RACF user ID that a client inputs, or a surrogate user ID that is defined in the IBM HTTP Server configuration file (/etc/httpd.conf) in advance, is assigned to each worker thread.

If your installation needs to control sensitive information served by the HTTP Server, it is advisable that you force browsers to enter a user ID and password. However, to force all clients on the intranet or Internet to input their user IDs and passwords whenever they access public information is not only uncommon, it is tedious to the end users. In such a case, it is better to use a surrogate user ID, allowing the HTTP Server to bring up public access files without requiring the user to enter a user ID.

5.1.2 Setting up SAF control

In the Web server environment, we define a resource as anything that can be delivered (served) by the Web server. For example, Web pages and CGI programs are both resources. When we define a surrogate user ID for accessing some resources, the RACF password is not specified on the pthread_security_np() service invocation. However, an additional check is made to ensure that the server is authorized to act as the client. UNIX System Services uses profiles defined to the RACF SURROGAT class to authorize the Web server to act as a surrogate of a client. Profiles defined to the SURROGAT class are of the form:

BPX.SRV.<userid>

in which <userid> is the RACF user ID of the user on behalf of whom the server will act as a surrogate.

The first step is to create a BPX.SERVER profile in class FACILITY. BPX.SERVER restricts the use of the pthread_security_np() service. Then, give the Web server's user ID (normally WEBSRV) READ authority to BPX.SERVER in order to authorize a server to act as a surrogate for client user IDs. The following is an example of the commands:

```
RDEFINE FACILITY BPX.SERVER UACC(NONE)
PERMIT BPX.SERVER CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

For more detailed information regarding BPX.SERVER, see 3.3.2.3, "BPX.SERVER authority" on page 65.

You should also confirm that the server and language environment load libraries have program control turned on (with NOPADCHK). To check this, use the following RACF command:

```
RLIST PROGRAM *
```

For the IBM HTTP Server to process requests for thread-level security without passwords, follow the steps shown below. The steps below are for IBM HTTP

Server with RACF user ID WEBSRV with the ability to support user ID ANONYMO without a password.

1. First, you must activate the SURROGAT class in RACF:

   ```
   SETROPTS CLASSACT(SURROGAT)
   ```

   This has to be done only once on the system. The SURROGAT class may already have been set up on your system. If a daemon or server you are running will be using the SURROGAT support heavily, consider using the `RACLIST` command to keep the SURROGAT profiles in storage. The following example shows how to cache the SURROGAT profiles in storage:

   ```
   SETROPTS RACLIST(SURROGAT)
   ```

2. If the SURROGAT class has been made resident using `SETROPTS RACLIST`, any changes to the SURROGAT profiles must be followed by a refresh of the in-storage tables. To create the SURROGAT class profile for user ANONYMO, issue:

   ```
   RDEFINE SURROGAT BPX.SRV.ANONYMO UACC(NONE)
   SETROPTS RACLIST(SURROGAT) REFRESH
   ```

   A similar SURROGAT profile is required for each user ID that a server uses without specifying a password.

3. To permit server WEBSRV to create a thread-level security environment for user ANONYMO, issue the `PERMIT` command:

   ```
   PERMIT BPX.SRV.ANONYMO CLASS(SURROGAT) ID(WEBSRV) ACCESS(READ)
   SETROPTS RACLIST(SURROGAT) REFRESH
   ```

For more information regarding thread-level security, see *OS/390 UNIX System Services Planning*, SC28-1890.

5.1.3 How to protect resources

HTTP provides an authentication mechanism to allow servers to define access permissions for clients and the resources they want to access. The authentication method can be one of the following:

- Basic authentication

  Basic authentication is based on a user ID and password. In this authentication scheme, the server will permit the connection only if the user ID and password are validated. In basic authentication, user IDs and passwords are not encrypted. They are encoded in base64 format.

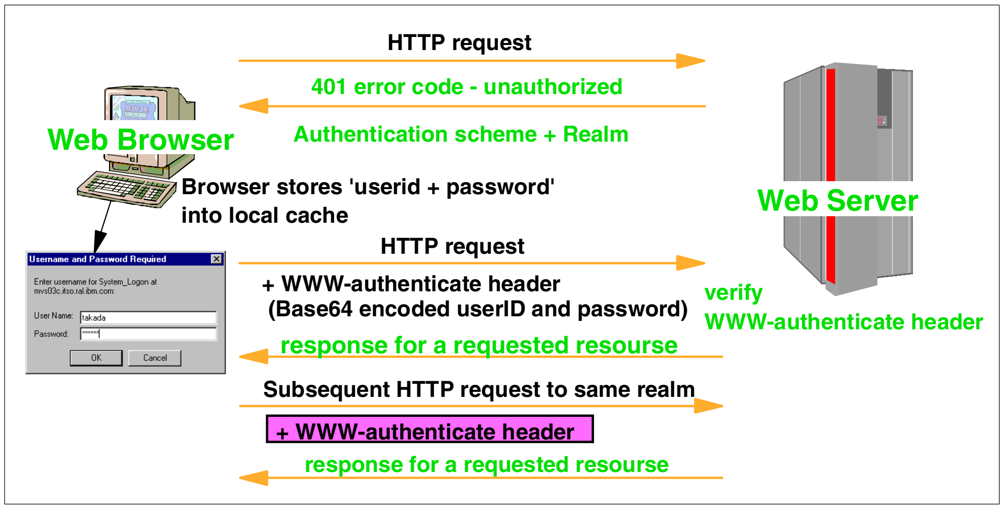

HTTP request
401 error code - unauthorized
Web Browser
Authentication scheme + Realm
Browser stores 'userid + password' into local cache
Web Server
Username and Password Required
Enter username for System_Logon at mvs03c.itso.ral.ibm.com:
User Name: takada
Password: ******
OK
Cancel
HTTP request
+ WWW-authenticate header (Base64 encoded userID and password)
verify WWW-authenticate header
response for a requested resourse
Subsequent HTTP request to same realm
+ WWW-authenticate header
response for a requested resourse

Figure 44. Overview of basic authentication

When the browser requests a protected resource, the Web server returns a 401 error code (unauthorized) and includes the WWW-Authenticate header.

The WWW-Authenticate header tells the browser which authentication schemes the Web server supports and the name of the realm that contains the protected resource. If the user has not already entered a valid user ID and password for this Web server and realm, the password window is presented (see Figure 44). In subsequent requests, the browser can find the information from its internal cache or password file. The browser then requests the same document, but this time with an Authentication header that includes the user's ID and password. Thus, each *logon* (supply of user ID and password) is applied to multiple requests for the same Web server and realm. The password and user ID normally remain available until the browser is recycled or its memory cache is flushed.

- HTTP digest access authentication

  The digest authentication scheme is an extension of HTTP and described in RFC 2617. In this authentication scheme, 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 network is not simply an encrypted form of the password, which could be decrypted if you had the correct key, but is a one-way hash value of the password, which cannot be decrypted. So digest authentication provides a higher level of security than the Base64 encoded password. Currently, IBM HTTP Server for OS/390 does not support digest authentication. However, digest authentication is not supported by many Web browsers and other Web servers, either.

The remainder of this section assumes basic authentication is being used with the IBM HTTP Server for OS/390.

5.1.3.1 Access control directives

Access control directives specify the user name the server changes to before accessing files. It can be set globally (default setting) via the UserID directive. It can also be set locally (for a specific realm) within a Protection, Protect or

DefProt setup. The server does not normally serve data from its own WEBSRV user ID because of its level of authority (normally, WEBSRV has UID=0).

The ability to change to a different z/OS UNIX identity is controlled on OS/390 via the pthread_security_np() service.

The server user ID, usually WEBSRV, must be given surrogate authority for any specified surrogate user ID. It also needs to have READ access to profile BPX.SERVER. In SYS1.SAMPLIB, member IMWJSEC can be used to automate the creation of the security environment.

IBM HTTP Server uses four types of access control user IDs: %%CLIENT%%, %%SERVER%%, %%CERTIF%%, and surrogate user ID.

- %%CLIENT%%

  This special string tells the server to require that the requester have a local OS/390 user ID and password. The requester's user ID is used to access the data. The user is prompted for a valid password.

- %%SERVER%%

  This special string tells the server to use its own user ID to access data.

Note

Be extremely cautious when using %%SERVER%%. Because your server is normally running as a superuser, all users have superuser authority.

Note, however, that the HTTP Server's running under the superuser authority is no longer required. Refer to APAR PQ41777 for the complete information.

- %%CERTIF%%

  This special string tells the server to treat SSL connection certificate data in a special way. The Web server, when presented with an SSL session with client certificate data present, calls RACF to map the client certificate to a local OS/390 user ID and password. If the session is not using SSL, there is no certificate present, the underlying support is not available, or the certificate cannot be mapped, then the request is treated as if %%CLIENT%% had been specified.

- Surrogate user ID

  As mentioned in 5.1.1, "Server security structure" on page 103, this is generally the user ID for public access. If this user ID is specified on a UserID directive, no security checking is performed. All realms are accessible unless specifically protected via a DefProt or Protect directive.

  For example, if you specify ANONYMO as a surrogate user ID in the UserID directive, you have to define a RACF SURROGAT class profile, BPX.SRV.ANONYMO. Then you also have to give READ access authority for this profile to the Web server's user ID (normally WEBSRV):

```
ADDGROUP EXTERNAL OMVS(GID(999))
ADDUSER ANONYMO DFLTGRP(EXTERNAL) OMVS(UID(5) HOME('/') PROG('/bin/sh'))
RDEFINE SURROGAT BPX.SRV.ANONYMO UACC(NONE)

PERMIT BPX.SRV.ANONYMO CLASS(SURROGAT) ID(WEBSRV) ACCESS(READ)
SETROPTS RACLIST(SURROGAT) REFRESH
```

Figure 45 shows the authentication flow when a Web client accesses IBM HTTP Server. After receiving an HTTP request from the browser, IBM HTTP Server verifies which security scheme is specified in the UserID directive of HTTP Server's configuration file (normally, /etc/httpd.conf).

If a surrogate user ID is specified, IBM HTTP Server assigns the UID of the surrogate user ID to a thread for this request using the pthread_security_np() service. For example, if user ID EMPLOYEE with UID=100 is specified as a surrogate user ID, a worker thread runs with UID=100. If %%SERVER%% has been specified, OS/390 HTTP Server assigns the HTTP Server's UID, that is UID=0 in most implementations, to a thread. In either case, OS/390 does not require any more authentication information, such as a user ID/password or a client X.509 digital certificate.

Note: IBM HTTP Server can run under the authority other than superuser. See APAR PQ41777 for detail information.

On the other hand, if %%CLIENT%% or %%CERTIF%% is specified, the client is required to submit security information. In the case of %%CLIENT%%, the Web browser displays a window such as that shown in Figure 46 on page 108, which prompts the user to enter his or her user ID and password. Then IBM HTTP Server creates a worker thread with a UID of this RACF user ID.

If %%CERTIF%% is specified, client authentication is required. A client must submit his or her X.509 client certificate and this certificate must be associated with his or her RACF user ID. After authentication is completed successfully, IBM HTTP Server creates a worker thread with the UID of this RACF user ID.

In the case of %%CLIENT%% or %%CERTIF%%, the worker thread's user ID is the RACF user ID that a user enters in a window or is associated with a client certificate, and accessing resources such as HTML files, GIF images, and CGI programs is controlled by HFS file attributes (unless already restricted via a DefProt or Protect statement).

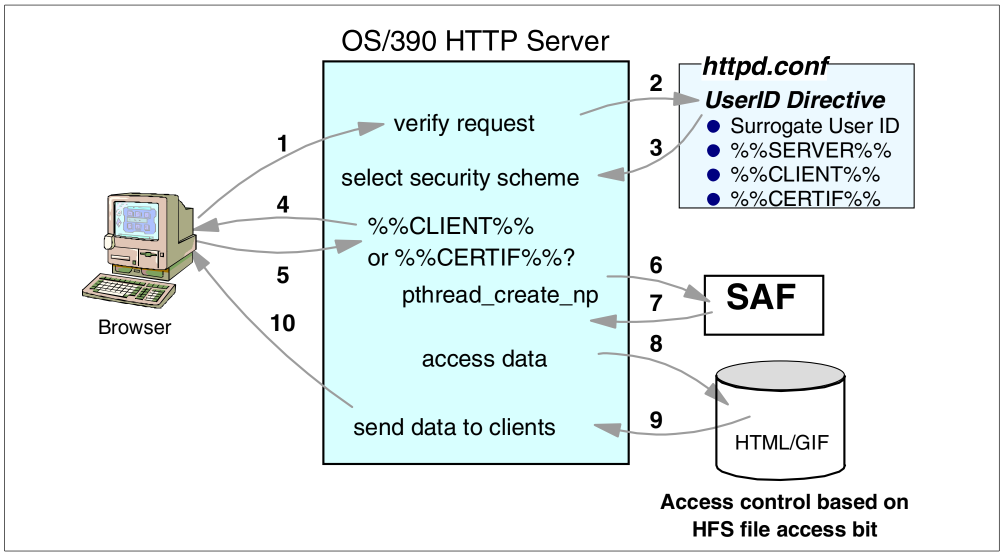


Figure 45. Authentication flow when accessing Web server

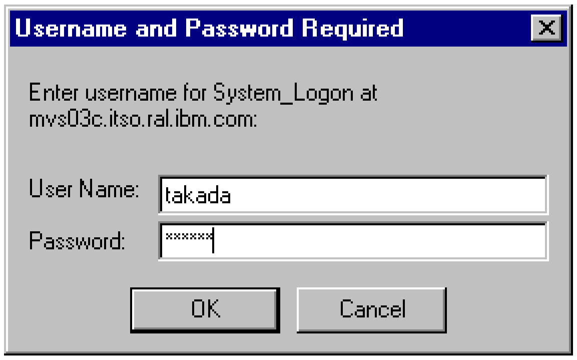
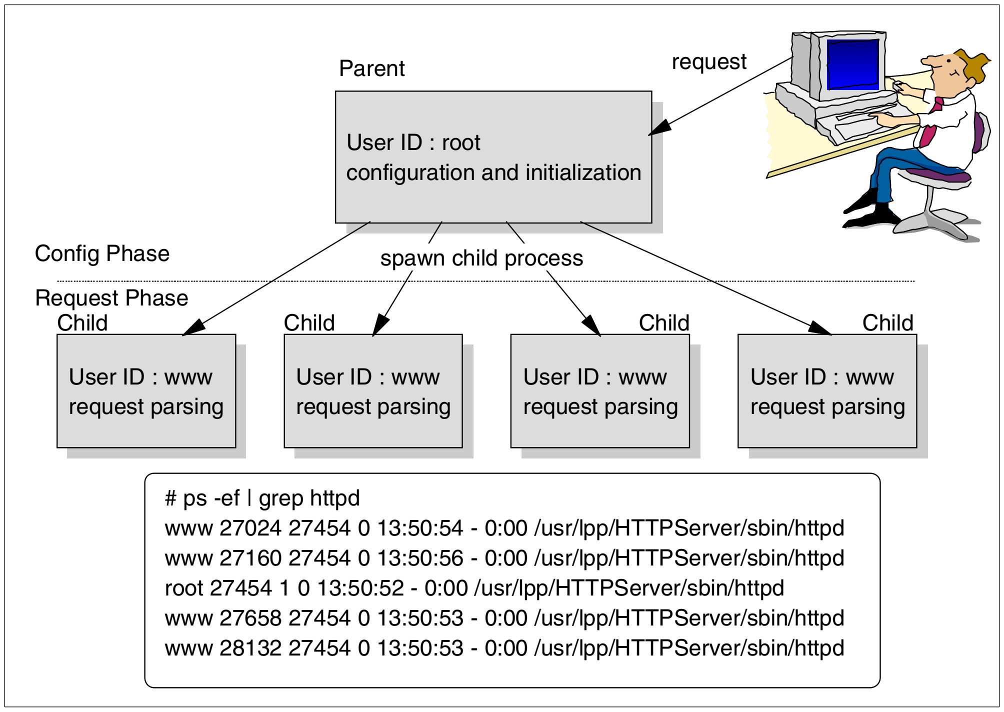

Username and Password Required
Enter username for System_Logon at mvs03c.itso.ral.ibm.com:
User Name: takada
Password: ******
OK
Cancel

Figure 46. Netscape Navigator's pop-up window for the user ID and password

The ability to have a different UID for each thread within a single process is considered superior to the schemes used by other UNIX-based Web servers. In other UNIX environments, Web servers cannot have threads with different UIDs within a process. Web servers must normally be assigned root authority in order to have sufficient privileges to open port 80 (the default HTTP protocol port). Thus, if worker threads are running in the same process as a Web server, all worker threads acquire root authority. Obviously, the fewer processes running under root authority, the more secure the system.

One solution to this is that used by the Apache Web server. It runs a main process under root, which in turn spawns child processes that change their user ID entity to whatever user and group are configured in the server's configuration file (see Figure 47). In this figure, these child processes run under the user www:



```
# ps -ef | grep httpd
www 27024 27454 0 13:50:54 - 0:00 /usr/lpp/HTTPServer/sbin/httpd
www 27160 27454 0 13:50:56 - 0:00 /usr/lpp/HTTPServer/sbin/httpd
root 27454 1 0 13:50:52 - 0:00 /usr/lpp/HTTPServer/sbin/httpd
www 27658 27454 0 13:50:53 - 0:00 /usr/lpp/HTTPServer/sbin/httpd
www 28132 27454 0 13:50:53 - 0:00 /usr/lpp/HTTPServer/sbin/httpd
```

Figure 47. Alternative Web server process structure

In other UNIX platforms, Web administrators have to define access authority for requests from clients in advance. As a consequence every access authority has the same security information. On the other hand, in the UNIX System Services environment, a different access authority for each request can be created dynamically. In addition to the scalable server function, which allows IBM HTTP Server to run in different address spaces and define different performance goals for each workload type (simple HTML, CGI, Server API, Servlet and so forth), thread level security is one of the biggest advantages of the OS/390 solution.

In critical Web environments such as online retailing, Internet banking and so on, the functions of changing access authority dynamically and workload management are indispensable. Because IBM HTTP Server has these abilities, it is the most appropriate Web server for a high-volume and secure Web environment. For detailed information about scalable Web server function, see *OS/390 Workload Manager Implementation and Exploitation*, SG24-5326.

5.1.3.2 Protection directives

The use of Protection directives allows the Web server to control and limit access to certain resources. Protection directives can also be used to authenticate a user or to change the default user ID for the access of certain resources. In this section we will show you some examples of Protection directives and how they work.

Protection directives work based on the URL definition and can protect a specific file, resource, or protect everything beyond a certain directory. Because Protection directives work on the URL level and we have the ability to map real resources or files to the URL using Pass or Exec configuration directives, we are able to set up different Protection directives for the same resource or file. This can be achieved by accessing the real resource using two or more different URLs and mapping them accordingly.

Security exposure

When using Fast Response Cache Accelerator (FRCA), all pages that are placed in cache are accessible to any browser - even pages that originally required authentication! Consequently, any documents that need to be protected and have a Protection or Protect directive that specifies a Map value of “All” or “Anybody” should be listed on a `FRCANoCaching` directive. Of course, documents served over an SSL connection are never cached. See APAR PQ40310 for more information.

When Protection directives require verification, they may use the same schemes and keywords as described in 5.1.3.1, “Access control directives” on page 105. They can also extend this possibility to groups and group files and certain parts of digital certificates, and can check passwords against RACF as well as against webmaster-maintained password files.

In the following examples, we show some typical access control definitions.

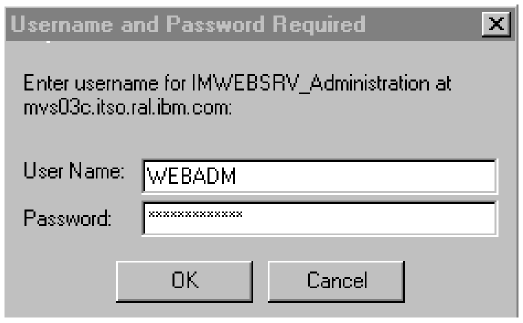
```
Protection IMW_Admin {
        ServerId        IMWEBSRV_Administration
        AuthType        Basic
        PasswdFile      %%SAF%%
        Mask            WEBADM,webadm,TAKADA,takada
}

Protect /admin-bin/* IMW_Admin WEBADM
Protect /reports/*   IMW_Admin WEBADM

Pass    /admin-bin/webexec/*  /usr/lpp/internet/server_root/admin-bin/webexec/*
Exec    /admin-bin/*          /usr/lpp/internet/server_root/admin-bin/*
Pass    /reports/javelin/*    /usr/lpp/internet/server_root/pub/reports/javelin/*
Pass    /reports/java/*       /usr/lpp/internet/server_root/pub/reports/java/*
Pass    /reports/*            /usr/lpp/internet/server_root/pub/reports/*
```

Figure 48. RACF user ID/password authentication

The definition shown in Figure 48 is an example of how to protect access to URL `http://hostname/admin-bin/*` or `http://hostname/reports/*`. Pass and Exec directives map a URL to an HFS file path. The Protect directive defines which URLs should be authenticated and with whose user authority this request should be processed. In this example, when a client accesses the URL `/admin-bin/*` or `/reports/*`, the authentication specified in the Protection directive with label name 'IMW_Admin' is done and this request is processed with WEBADM's UID. Because user ID WEBADM is a surrogate user ID, you must define RACF SURROGAT class profile 'BPX.SRV.WEBADM' and give IBM HTTP Server's user ID READ authority to this profile.

The Protection directive defines the authentication method (AuthType and PasswdFile), realm (Server ID) and who is permitted to access the resources protected by this Protection directive (Mask). In this case, authentication is done by a SAF interface, that is, RACF. A client must input the user ID WEBADM or TAKADA and its valid password. Note that the user IDs are case-sensitive. As a precaution, webadm and takada are also explicitly specified in the Mask subdirective.

Note also that the user IDs are used for the purpose of authentication only and are not related to the user ID assigned to the user request. As you can see in Figure 49, the Server ID specified in the Protection directive is displayed in the window to ask the client's user name and password.


Username and Password Required

Enter username for IMWEBSRV_Administration at mvs03c.itso.ral.ibm.com:

User Name: WEBADM

Password: **************

Figure 49. Netscape Navigator's user authentication window

Once the authentication for this server ID is successful, when accessing a realm with the same server ID, you do not need to input a user ID and password again as long as you do not shut down the browser or clear the browser's memory cache.

Group files (GroupFile subdirective) are not compatible with RACF user IDs and group names. Further, password files (PasswdFile subdirective) are not advisable when RACF is available to handle the equivalent function. In addition, the maintenance of separate user lists is not advisable. As such, group and password files are not recommended.

What about LDAP with a password file? While this is a possibility, authentication would need to occur on each single resource request. As such, LDAP would need to be consulted every time you wanted to view a new URL. The performance impact would be so great that this option is not advisable.

In summary, use your local SAF product for user ID and password control.

The Protection, Protect and DefProt directives in the httpd.conf file have more subdirectives than we can show in this redbook. For more information about the Protection, Protect and DefProt directives in the httpd.conf file, see *IBM HTTP Server for OS/390 HTTP Server Planning, Installing, and Using*, SC31-8690.

5.1.4 Accessing back-end applications

If you want users to access back-end applications such as DB2, IMS, or CICS through the HTTP Server, you need to take the authority to access these resources into consideration.

Figure 50 shows how a user ID from a browser is passed to Net.Data through the IBM HTTP Server. This user ID is authenticated by IBM HTTP Server based on User ID, Protection or Protect directives. This user ID is used to check the authority to access DB2 resources. Therefore, to access DB2 resources via the HTTP Server, your user ID needs the authority to access not only the HTTP Server resources but also DB2 resources.

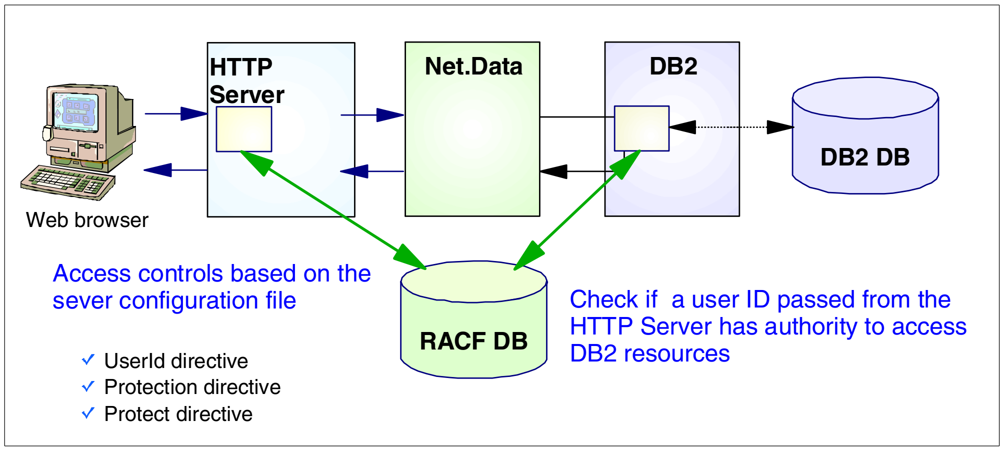

HTTP Server
Net.Data
DB2
DB2 DB
Web browser
Access controls based on the sever configuration file
✓ UserId directive
✓ Protection directive
✓ Protect directive
RACF DB
Check if a user ID passed from the HTTP Server has authority to access DB2 resources

Figure 50. User ID passed to Net.Data from IBM HTTP Server

Figure 51 is an example of the access control needed to query DB2 data through IBM HTTP Server and Net.Data. If clients access the resource with a label

DB2_Query on the Protect directive (1), a Protection directive with the same label (2) is used to decide how the access control for this resource should be done. In this case, you have to enter your RACF user ID and password because %%SAF%% is specified in the PasswdFile subdirective (3). The Mask subdirective specifies All (4) which means that all users defined in the password file (the RACF database) who are successfully authenticated will be allowed to execute the Net.Data connector. At that time, the user ID associated with the Net.Data request is the client's user ID because %%CLIENT%% has been specified in the UserID subdirective (5). Therefore, each client will need sufficient access to the DB2 objects needed to execute the query.

```
Protection DB2_Query {     2
      ServerId      NetData_DB2_Access
      AuthType      Basic
      UserID        %%CLIENT%%5
      PasswdFile    %%SAF%%        3
      Mask          All            4
}

Protect /netdata-cgi/db2www DB2_Query 1
```

Figure 51. Controlling access to DB2 resources through the HTTP Server

For more information about Net.Data, see the following URL:

`http://www.ibm.com/software/data/net.data/`

5.1.5 SSL-related features in the IBM HTTP Server for OS/390

The SSL protocol was first implemented in the IBM Internet Connection Secure Server for OS/390 (ICSS), introduced in September 1996. Since then, many enhancements have been made to the SSL support in Domino Go Webserver and IBM HTTP Server for OS/390, including SSL V3 support, support for the S/390 cryptographic hardware, and the possibility to associate RACF user IDs with client certificates used in SSL V3 client authentication.

5.1.5.1 Encryption support

Computer products containing encryption technology are subject to export and import controls by various national governments. Both Web servers and Web browsers are subject to these restrictions, depending on your geographic location.

For customers in most countries, U.S. Government export regulations are not a problem nowadays. Since October 2000, practically no restrictions for the export of cryptographic products into the EU countries and eight more countries exist. For most other countries, retail products can be exported to any customer. As a consequence, customers in all countries (except of some restricted countries) should order the North America Security Feature.

The IBM HTTP Server V5.3 for OS/390 requires the specification of one of the following security features:

1. North America Security Feature (level 3):

   - Generate asymmetric keys from 512-1024 bits; encrypt data with symmetric keys from 40-168 bits; sign data with keys from 512-1024 bits; check signatures with keys from 512-1024 bits.

- SSL V2 cipher specifications: Triple-DES; RC4 (128-bit); RC2 (128-bit); DES (56-bit); RC4 (40-bit); RC2 (40-bit).
- SSL V3 cipher specifications: Triple-DES SHA; RC4 SHA (128-bit); RC4 MD5 (128-bit); DES SHA (56-bit); RC4 MD5 (40-bit); RC2 MD5 (40-bit).

2. Export Security Feature (level 2):

- Generate asymmetric keys with a key size of 512 bits; encrypt data with symmetric keys from 40-56 bits; sign data with 512-bit keys; check signatures with keys from 512-1024 bits.
- SSL V2 cipher specifications: DES (56-bit); RC4 (40-bit); RC2 (40-bit)
- SSL V3 cipher specifications: DES SHA (56-bit); RC4 MD5 (40-bit); RC2 MD5 (40-bit).

The following table shows the feature codes for the IBM HTTP Server encryption features. Before OS/390 V2R10, three different encryption features were available; the third feature (France Security Feature, level 1) had a maximum symmetric key size of 40 bits.

Table 8. Cryptographic support FMIDs for IBM HTTP Server for OS/390

Product level	NA security	Export security	France security
HTTP Server 5.1 (HIMW510)	JIMW511	JIMW512	JIMW513
HTTP Server 5.2 (HIMW520)	JIMW521	JIMW522	JIMW523
HTTP Server 5.3 (HIMW530)	JIMW531	HIMW530	N/A

5.1.5.2 Global Server IDs

Before January 2000, U.S. Government export regulations did not allow the export of North America Security versions of Web browsers such as Netscape Communicator or Microsoft Internet Explorer to end users outside the United States and Canada. These Web browsers which would normally only use Export Security ciphers, could be enabled to use strong cryptographic algorithms in sessions with a Web server authorized to use strong encryption.

The U.S. Government had authorized VeriSign, Inc. and Thawte to issue special server certificates called *Global Server IDs* to those customers eligible to use strong encryption with servers outside the U.S. and Canada. These certificates are recognized by export versions of Microsoft Internet Explorer Version 3.02 and above, and by Netscape Navigator/Communicator Version 4 and above. When the Web browser recognizes the special certificate, it enables strong encryption routines such as RC4 with 128-bit keys or Triple DES with 168-bit keys. This process is sometimes also called Server Gated Cryptography (SGC).

For more information about Global Server IDs, see *Global Server Certificate Usage with OS/390 Webservers*, SG24-5623.

The only reason Global Server IDs are still an issue worth considering is the fact that a large number of users outside the U.S. and Canada are still using export versions of the Web browsers, so a server with a Global Server Certificate can enable them to use strong encryption algorithms in communications with this server. Downloading a new Web browser can require more than 15 MB to be downloaded which might frighten off users with only a modem connection.

5.1.5.3 Crypto hardware support for SSL

The Cryptographic Coprocessor Feature is available as a feature on the eServer z900, the S/390 Enterprise Servers (9672) and S/390 Multiprise Servers.

To use this hardware feature, the OS/390 Integrated Cryptographic Service Facility (ICSF) is needed. ICSF provides a cryptographic API that interfaces with the cryptographic hardware.

There is no specific setup for the SSL function to enable this support; once the server detects the presence of the crypto hardware feature, the encryption and decryption functions will be performed automatically using this feature.

The use of this hardware feature may help offload OS/390 work for SSL. In an SSL handshake, the private key decryption process can take over 70 percent of the CPU cycles necessary to process the handshake. By using the cryptographic coprocessors, an S/390 G5 or G6 processor can service about 150 full SSL handshakes per second while still leaving most of the CPU power to the applications. If this is insufficient, PCI Cryptographic Coprocessor cards (PCICC) can be added to 9672 G5 and G6 systems and 2064 systems which can increase the ability of the system to handle SSL handshakes considerably. A 9672 system equipped with eight PCICC cards can handle about 1000 full SSL handshakes per second while a 2064 system equipped with eight PCICC features (16 cards) can handle about 2000 full SSL handshakes per second.

5.1.6 SSL scenario

As mentioned in 2.4, “Secure Sockets Layer” on page 36, there are two options available when establishing an SSL connection. They are server authentication (SSL V2 and V3) and client authentication (SSL V3). While server authentication is mandatory, client authentication is optional. In this section, we discuss how to establish server authentication between IBM HTTP Server and Netscape Communicator 4.6.

5.1.6.1 Server authentication

The steps to start your IBM HTTP Server as a secure server are:

1. Establish a key database or RACF keyring with a server certificate

   The IBM HTTP Server for OS/390 5.3 can use a key database established with the GSKKYMAN utility or it can use a RACF keyring. If a key database (a file in the HFS) is used, the password to the key database must be specified in a stash file to allow the server to use the key database. The password is scrambled but not encrypted which can be a security exposure because any superuser can read the stash file. A RACF keyring does not require a password, it can only be accessed by the server’s user ID or a SPECIAL user. For examples on creating key databases or keyrings, see Appendix D, “OS/390 certificate management using RACF” on page 321.

2. Customize SSL-related directives in the HTTP Server configuration file

   To make OS/390 HTTP Server enable server authentication, you have to specify at least three directives in the httpd.conf file:

```
sslmode    on 1
sslport    443 2

keyfile  /u/takada/sslkey/server.kdb
keyfile  WEBRING  SAF              3
```

Figure 52. The SSL-related directives in the HTTP configuration file

1 Use this directive to turn on SSL support.

2 Use this directive to set the port for SSL security (HTTPS). The default is 443.

3 Use this directive to specify the key database file. If you do not specify the name of your key database, IBM HTTP Server cannot bind to a secure port. Alternatively, you can specify the name of a RACF keyring.

3. Restart the HTTP Server

   After the SSL-related directives in /etc/httpd.conf file shown in Figure 52 are modified, you stop and restart the HTTP Server. The following result of the `onetstat -c` command shows the HTTP Server listening on a normal HTTP port 80 and a secure HTTPS port 443.

```
TAKADA @ RA03:/u/takada>onetstat -p TCPIPB -c
MVS TCP/IP onetstat CS V2R8       TCPIP Name: TCPIPB           15:39:15
User Id  Conn     Local Socket           Foreign Socket         State
-------  ----     ------------           --------------         -----
ICAPCFGS 000000F8 0.0.0.0..1014          0.0.0.0..0             Listen
INETD1   00000020 0.0.0.0..109           0.0.0.0..0             Listen
INETD1   0000001E 0.0.0.0..2023          0.0.0.0..0             Listen
INETD1   0000012B 9.24.104.33..2023      9.24.106.155..1100     Establsh
INETD1   0000001F 0.0.0.0..110           0.0.0.0..0             Listen
TCPIPB   00000011 127.0.0.1..1026        127.0.0.1..1025        Establsh
TCPIPB   00000016 0.0.0.0..8823          0.0.0.0..0             Listen
TCPIPB   00000012 127.0.0.1..1025        127.0.0.1..1026        Establsh
TCPIPB   00000015 0.0.0.0..7723          0.0.0.0..0             Listen
TCPIPB   00000014 0.0.0.0..6623          0.0.0.0..0             Listen
TCPIPB   00000017 0.0.0.0..9923          0.0.0.0..0             Listen
TCPIPB   0000000C 0.0.0.0..1025          0.0.0.0..0             Listen
TCPIPB   00000013 0.0.0.0..23            0.0.0.0..0             Listen
WEBSRV   00000178 0.0.0.0..443           0.0.0.0..0             Listen
WEBSRV   00000177 0.0.0.0..80            0.0.0.0..0             Listen
ICAPIKED 000000FA 9.24.104.33..500       *..*                   UDP
ICAPSLOG 000000F7 0.0.0.0..514           *..*                   UDP.
```

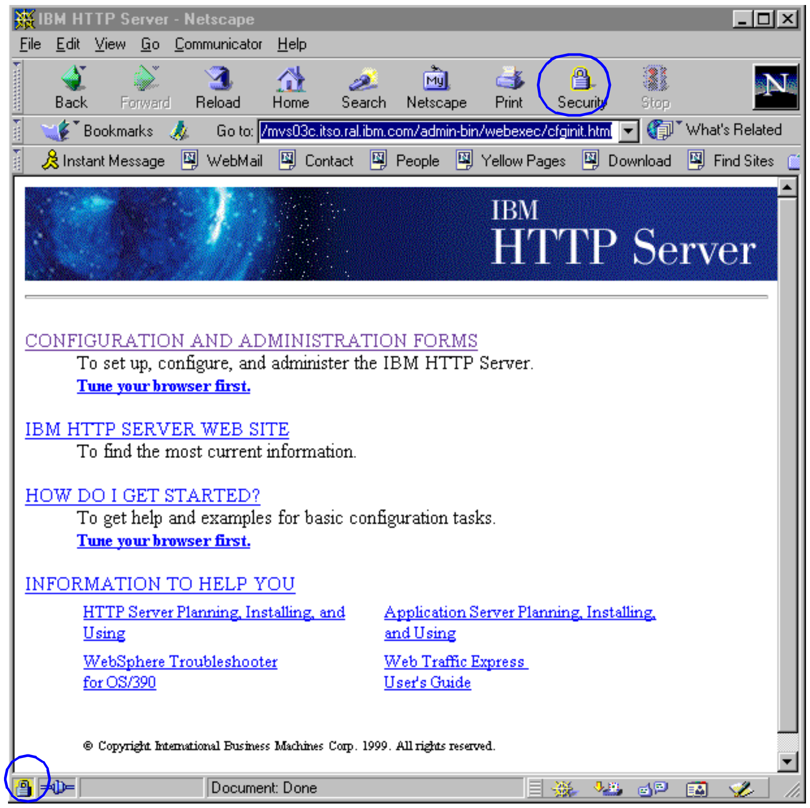
Now you can establish the SSL connection between IBM HTTP Server and a browser. You use the special URL format `https://hostname:port/` to request an SSL connection to an HTTP Server. If you do not specify a port number, the browser requests a connection to port 443 as a default.

Note

While it is possible to use other ports than 443 for SSL, this is likely to cause problems if clients will be accessing the server through firewalls with HTTP proxy servers. The reason is that a proxy server must be enabled for SSL tunneling by port number. Most proxies are only enabled for port 443 so they will block all SSL traffic to other ports without returning any error messages.

In Figure 53, we show an example where Netscape establishes an SSL connection with our secure server.

To view the information in the server certificate, click **Security** on the menu button of the Netscape Navigator window. After the security information window is displayed, click **View Certificate**. Then the window that shows the server certificate information appears (see Figure 54).



CONFIGURATION AND ADMINISTRATION FORMS
To set up, configure, and administer the IBM HTTP Server.
Tune your browser first.

IBM HTTP SERVER WEB SITE
To find the most current information.

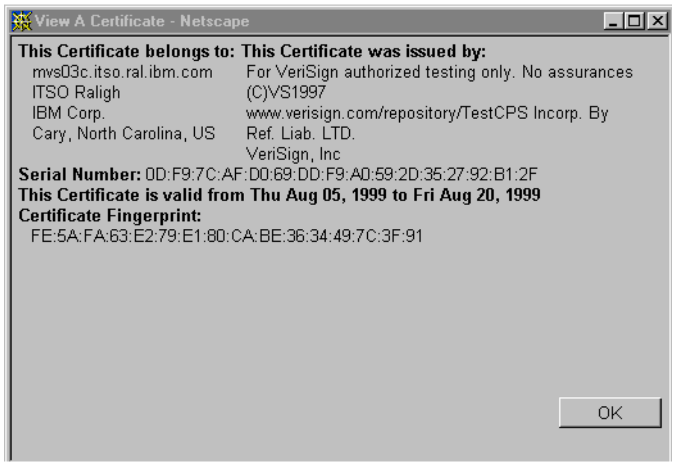
HOW DO I GET STARTED?
To get help and examples for basic configuration tasks.
Tune your browser first.

INFORMATION TO HELP YOU
HTTP Server Planning, Installing, and Using
Application Server Planning, Installing, and Using
WebSphere Troubleshooter for OS/390
Web Traffic Express User's Guide

© Copyright International Business Machines Corp. 1999. All rights reserved.

Document: Done

Figure 53. Example of the Netscape security screen with an SSL connection



This Certificate belongs to:
mvs03c.itso.ral.ibm.com
ITSO Raligh
IBM Corp.
Cary, North Carolina, US

This Certificate was issued by:
For VeriSign authorized testing only. No assurances
(C)VS1997
www.verisign.com/repository/TestCPS Incorp. By
Ref. Liab. LTD.
VeriSign, Inc

Serial Number: 0D:F9:7C:AF:D0:69:DD:F9:A0:59:2D:35:27:92:B1:2F
This Certificate is valid from Thu Aug 05, 1999 to Fri Aug 20, 1999
Certificate Fingerprint:
FE:5A:FA:63:E2:79:E1:80:CA:BE:36:34:49:7C:3F:91

Figure 54. Server certificate information

5.1.6.2 Client authentication

SSL Version 3 is required for client authentication. If you set up your server for SSL client authentication, the server requests a certificate from all clients making an https request.

We obtained the VeriSign trial client certificate from VeriSign's Web site and used that as the client certificate for our client authentication SSL test. In the following discussion, we show how to establish an SSL V3 connection for this purpose.

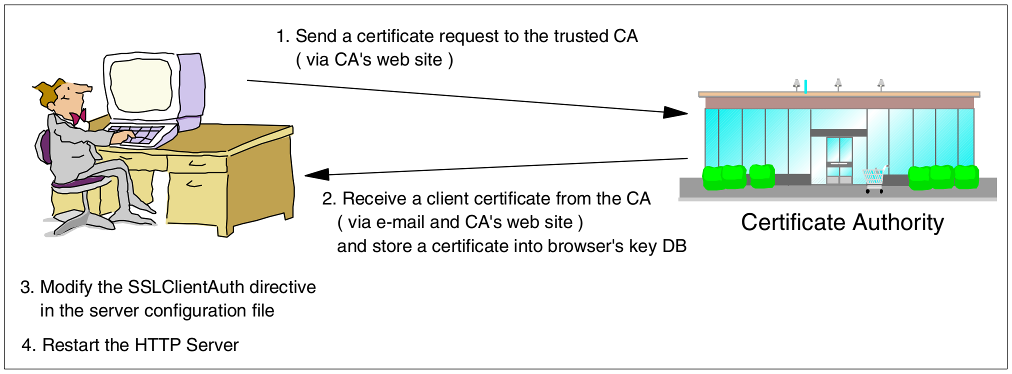
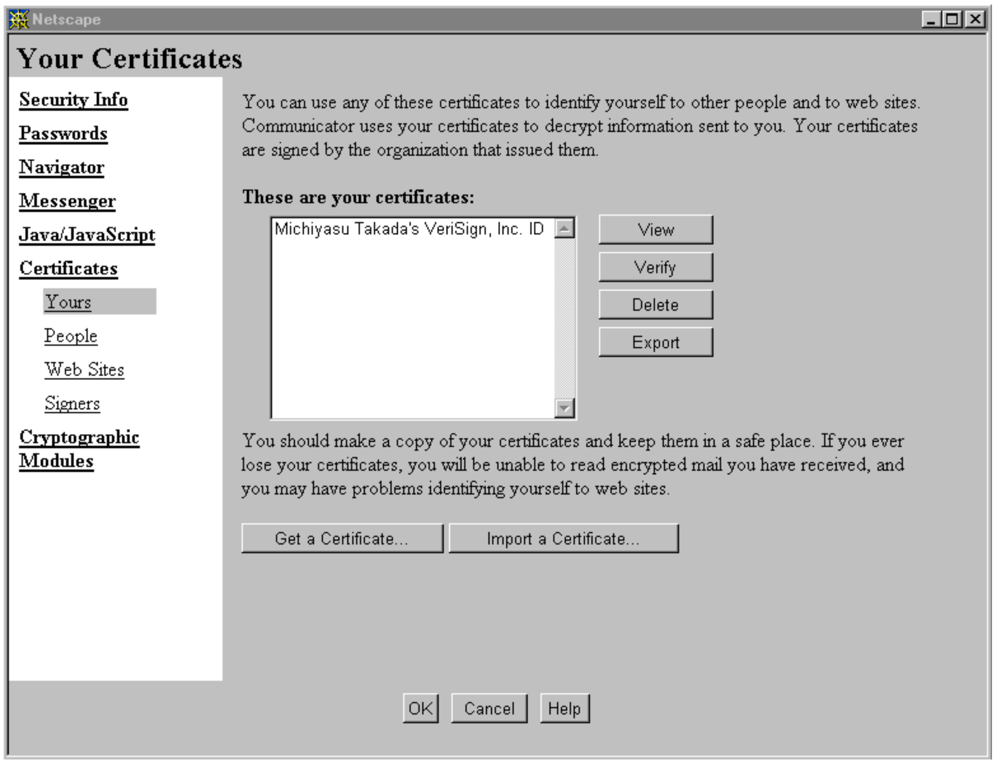

1. Send a certificate request to the trusted CA (via CA's web site)
2. Receive a client certificate from the CA (via e-mail and CA's web site) and store a certificate into browser's key DB
Certificate Authority
3. Modify the SSLClientAuth directive in the server configuration file
4. Restart the HTTP Server

Figure 55. Flow to implement SSL client authentication

Figure 55 shows the steps that are usually needed to request a client certificate from a certificate authority (CA) and store the certificate in the browser's key database. The certificate can be sent by e-mail or downloaded from the CA Web site. To enable the use of client certificates in the Web server, the `SSLClientAuth` directive needs to be set to "local" and the Web server needs to be restarted.

To view your client certificate information, click the **Security** button on the Netscape Navigator's primary window, click **Yours** in the Certificates category, select the certificate you desire from the These are your certificates: scroll box, and click **View**. Then the window that shows your certificate information appears (see Figure 57).



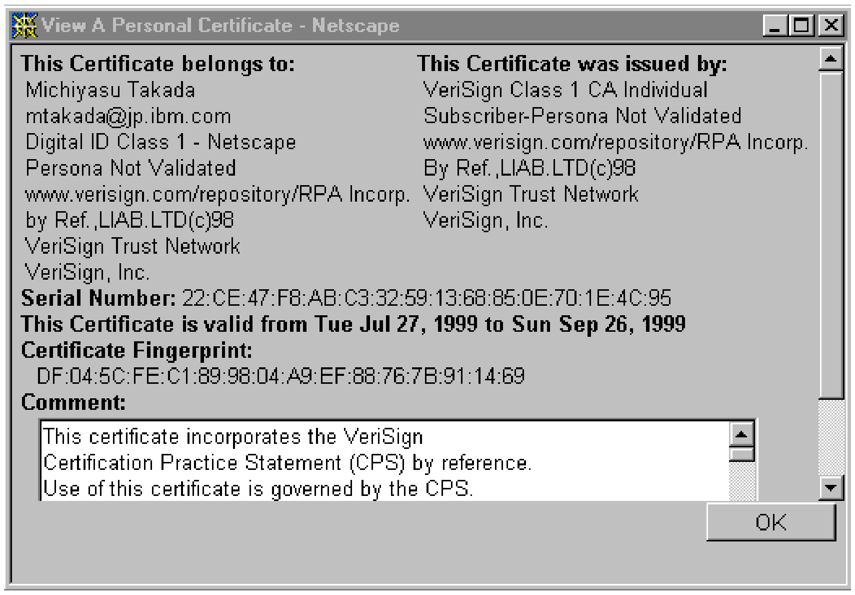
You can use any of these certificates to identify yourself to other people and to web sites. Communicator uses your certificates to decrypt information sent to you. Your certificates are signed by the organization that issued them.

These are your certificates:

Michiyasu Takada'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 56. Security information window



This Certificate belongs to:	This Certificate was issued by:
Michiyasu Takada	VeriSign Class 1 CA Individual
mtakada@jp.ibm.com	Subscriber-Persona Not Validated
Digital ID Class 1 - Netscape	www.verisign.com/repository/RPA Incorp.
Persona Not Validated	By Ref.,LIAB.LTD(c)98
www.verisign.com/repository/RPA Incorp.	VeriSign Trust Network
by Ref.,LIAB.LTD(c)98	VeriSign, Inc.
VeriSign Trust Network	
VeriSign, Inc.	

Serial Number: 22:CE:47:F8:AB:C3:32:59:13:68:85:0E:70:1E:4C:95
This Certificate is valid from Tue Jul 27, 1999 to Sun Sep 26, 1999
Certificate Fingerprint:
DF:04:5C:FE:C1:89:98:04:A9:EF:88:76:7B:91:14:69
Comment:
This certificate incorporates the VeriSign Certification Practice Statement (CPS) by reference. Use of this certificate is governed by the CPS.

Figure 57. Client certificate information

With client authentication, the server can control access to resources by using the client certificate information instead of challenging for a user ID/password. This function is enabled by coding SSL authentication parameters in the right places.

First of all, you have to set the SSLClientAuth directive to on (the default is off). The SSLClientAuth directive has the following three options for client authentication:

- `local`

  The OS/390 HTTP Server uses the trusted root CA certificate in the key database file stored locally for client authentication.

- `strong`

  The Server uses the X.500 directory server specified in the SSLX500Host directive for client authentication.

- `passthru`

  When this option is specified, the server performs no validation of the client's certificate. In this case, alternative validation methods, such as a user supplied GWAPI or CGI routine, need to be available.

In our test environment, we changed this option to local. That is to say, IBM HTTP Server used a locally stored key database file to check if a root CA that signed a client certificate is really trusted.

```
SSLClientAuth local
```

The OS/390 HTTP Server provides 13 possible SSL-specific Protection subdirectives and you can control the access of your sensitive resources based on any combination of these subdirectives.

The following six subdirectives are used to identify a client:

```
CommonName, Country, Locality, StateOrProvince, Organization, OrgUnit
```

The following six subdirectives are used to identify an Issuer, that is, a CA:

```
IssuerCommonName, IssuerCountry, IssuerLocality, IssuerStateOrProvince,
IssuerOrganization, IssuerOrgUnit
```

When the last and thirteenth one, `SSL_ClientAuth`, is used, the server permits access only if a client submits its certificate.

To confirm which identity information the client certificate gives to the OS/390 HTTP Server, you should obtain the information using the -vv trace option provided by IBM HTTP Server:

```
Client certificate data:
 Organization = VeriSign, Inc.
 Organizational Unit = Digital ID Class 1 - Netscape
 Issuer Common Name = VeriSign Class 1 CA Individual Subscriber-Persona Not Validated
 Issuer Organization = VeriSign, Inc.
 Issuer Organizational Unit = www.verisign.com/repository/RPA Incorp. By Ref.,LIAB.LTD(c)98
```

Figure 58. Client certificate data obtained by -vv trace

You can define some SSL-specific Protection subdirectives based on the result shown in Figure 58.

Figure 59 shows an example of the protection based on the client certificate information. The use of these subdirectives does not create any association of the certificate owner with a RACF user ID. Therefore, a Userid subdirective or a user ID positional parameter on the Protect directive is needed to specify the surrogate user ID for this request. In our example, user ID WEBADM will be used.

```
Protection IMW_Admin {
      ServerId           IMWEBSRV_Administration
      AuthType           Basic
      Organization       "VeriSign, Inc." 1
      OrgUnit            "Digital ID Class 1 - Netscape"
      IssuerOrganization "VeriSign, Inc."
      IssuerOrgUnit      "www.verisign.com/repository/RPA Incorp. By Ref.,LIAB.LTD(c)98"
      Mask               anybody@(*) 2
}

Protect /admin-bin/* IMW_Admin WEBADM
Protect /reports/*   IMW_Admin WEBADM

Pass    /admin-bin/webexec/*  /usr/lpp/internet/server_root/admin-bin/webexec/*
Exec    /admin-bin/*          /usr/lpp/internet/server_root/admin-bin/*
Pass    /reports/javelin/*    /usr/lpp/internet/server_root/pub/reports/javelin/*
Pass    /reports/java/*       /usr/lpp/internet/server_root/pub/reports/java/*
Pass    /reports/*            /usr/lpp/internet/server_root/pub/reports/*
```

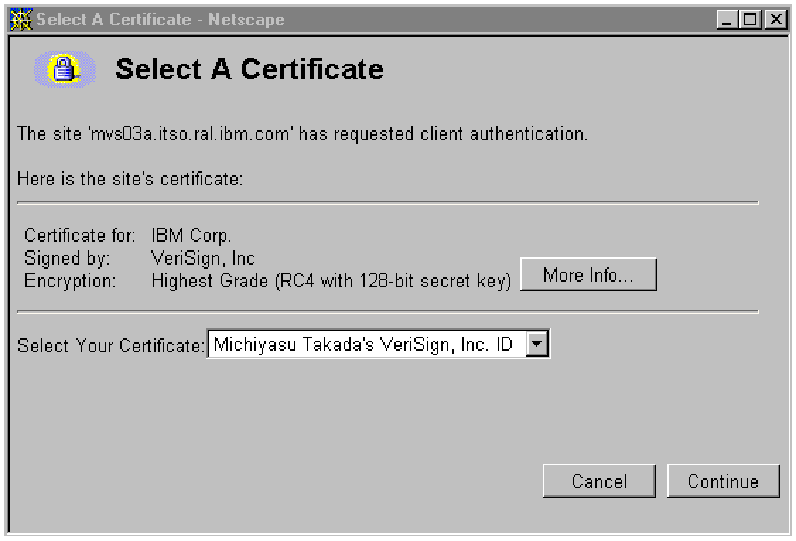
Figure 59. Access control using SSL V3 client authentication

1 Specify one or more certificate identities. We selected Organization, OrgUnit, IssuerOrganization, and IssuerOrgUnit for client authentication. Unless the certificate identities the client submits completely match this information, the authentication fails. If you have specified the -vv trace option in the server's start procedure, you will see the following message:

```
Forbidden... SSL Client Authentication failed for URL
/usr/lpp/internet/server _root/admin-bin/webexec/cfginit.html for
9.24.106.145.
The Client Certificate subject DN is not valid
```

2 Only if the client certificate data matches the entry specified in the SSL-related Protection subdirectives does the server permit the client to access this resource. You can control this more strictly by defining a specific user common name.

When you access an IBM HTTP Server that has client authentication enabled, you receive the window that prompts you to select a certificate (see Figure 60).


Select A Certificate - Netscape

Select A Certificate

The site 'mvs03a.itso.ral.ibm.com' has requested client authentication.

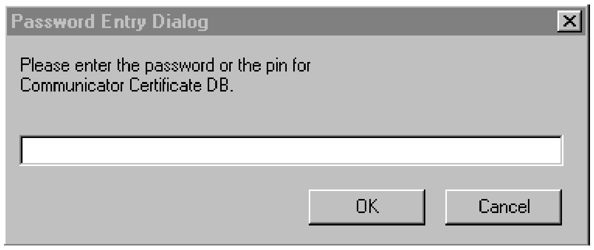
Here is the site's certificate:

Certificate for:	IBM Corp.
Signed by:	VeriSign, Inc
Encryption:	Highest Grade (RC4 with 128-bit secret key)

Select Your Certificate: Michiyasu Takada's VeriSign, Inc. ID

Figure 60. Netscape Navigator's window requiring your certificate

In Figure 60, after you click **Continue**, you will see the pop-up window shown in Figure 61 if you set a password for your certificate. We strongly recommend you set a password when storing your certificate into your browser. If you do not, someone else could easily use your certificate while you are away from your workstation. Here, you must enter a valid password. This password is what you entered when you first stored your new certificate.


Password Entry Dialog

Please enter the password or the pin for
Communicator Certificate DB.

Figure 61. Enter a password for your certificate

After you succeed in client authentication, you can establish an SSL connection and communicate with IBM HTTP Server for OS/390 securely.

IBM HTTP Server for OS/390 V2R10 provides the PKIServ utility which is a Web-based utility that enables an installation to issue client certificates to users with a user ID in RACF. For more information see the documentation for APARs OW45211/OW45212 and the following URL:

```
http://www-1.ibm.com/servers/eserver/zseries/zos/racf/webca.html
```

5.1.7 Associating a client certificate with a RACF user ID

The IBM HTTP Server is able to execute Web requests with a user ID retrieved from RACF on the basis of the client certificate received during an SSL handshake. The following example shows how the parameter %%CERTIF%% is used in a Protection directive for this purpose:

```
Protection very_secure {
        ServerId        CICS_Project
        AuthType        Basic
        SSL_ClientAuth  client
        UserId          %%CERTIF%%
        PasswdFile      %%SAF%%
        Mask            anybody
```

The Mask value of “Anybody” is needed to prevent the Web server from prompting the client for user ID and password. If “All” is used, the user will have to enter a valid user ID and password in addition to the client certificate.

`SSL_ClientAuth client` will cause all requests from clients who cannot present an acceptable certificate to fail with a “403 Forbidden” status code. If this subdirective has not been specified, the Web server will fall back to %%CLIENT%% processing if the client does not provide a valid certificate. The client certificate must be signed by a CA whose root CA certificate is contained in the Web server’s key database or RACF keyring to be acceptable.

5.1.7.1 RACF digital certificate support

The security server component that was first shipped in OS/390 Version 2 Release 4 introduced the ability to verify digital certificates and to assign a user ID based on the certificate. The digital certificate and the user-related information are stored in the RACF class DIGTCERT. The `RACDCERT` command can be used to add, alter, list or delete DIGTCERT profiles.

The digital certification flow is:

- IBM HTTP Server passes the client’s digital certificate to UNIX System Services. It is assumed that the Web server verifies the validity of the client certificate. This means that the server has verified that the certificate is genuine and was issued by a trusted authority, that its validity date is current, and that the client is the owner of the certificate.
- UNIX System Services passes the certificate as a parameter into RACF's initACEE callable service.
- RACF extracts information from the digital certificate, identifies a RACF user ID from it, and builds the ACEE.

For detailed information see *OS/390 V2R10 Security Server (RACF) Security Administrator's Guide*, SC28-1915.

5.1.7.2 Install and maintain digital certificates in RACF

To install a digital certificate in RACF you need to enable the RACF DIGTCERT class and add it to the RACLIST:

```
SETR CLASSACT(DIGTCERT) RACLIST(DIGTCERT)
```

To maintain DIGTCERT, the command `RACDCERT` is used and needs to be defined in the AUTHCMD NAMES list in the IKJTSOxx member of SYS1.PARMLIB. The `RACDCERT` command is a RACF TSO command used to:

- List information about the existing certificates for a specific RACF-defined user ID, or your own user ID.

- Add a certificate definition and associate it with a specified RACF-defined user ID, or your own user ID, and set the TRUST flag.
- Alter the TRUST flag for an existing definition.
- Delete an existing definition.

To facilitate the addition of a certificate definition, the input to `RACDCERT` for the ADD function is the name of a data set that contains the digital certificate.

To issue the `RACDCERT` command, you must have one of the following RACF authorities or permissions:

- The SPECIAL attribute
- Sufficient authority to RACF resource IRR.DIGTCERT.function in the FACILITY class, where function is LIST, ADD, ALTER or DELETE
  - READ access to IRR.DIGTCERT.function to perform the function for yourself
  - UPDATE access to IRR.DIGTCERT.function to perform the function for others

The following setting shown in Figure 62 allows a user to register a certificate with his or her own RACF user ID:

```
RDEFINE FACILITY IRR.DIGITCERT UACC(NONE)
PERMIT IRR.DIGITCERT.LIST CLASS(FACILITY) ID(TAKADA) ACCESS(READ)
PERMIT IRR.DIGITCERT.ADD CLASS(FACILITY) ID(TAKADA) ACCESS(READ)
PERMIT IRR.DIGITCERT.ALTER CLASS(FACILITY) ID(TAKADA) ACCESS(READ)
PERMIT IRR.DIGITCERT.DELETE CLASS(FACILITY) ID(TAKADA) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Figure 62. Protect RACDCERT command

Whenever you issue a `RACDCERT ADD, ALTER` or `DELETE`, you should refresh the DIGTCERT class by issuing the `SETROPTS RACLIST(DIGTCERT) REFRESH` command. If you do not refresh the RACLISTed DIGTCERT profiles, RACF will still use the new digital certificate. However, performance may be affected.

The TRUST and NOTRUST parameters are used to specify whether the mapping of a certificate to a RACF user ID is valid or not.

There are two ways to register a digital certificate with RACF:

- Using the TSO command interface with the RACF command `RACDCERT`.
- Using the RACF self-registration Web application distributed in OS/390 V2.6 and later in SYS1.SAMPLIB (members RACINSTL and IRR31933).

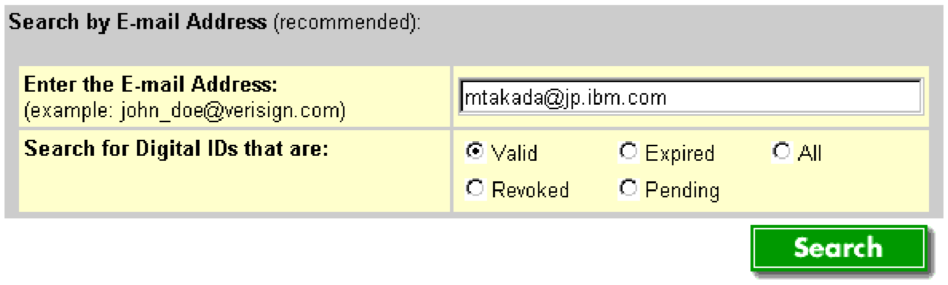
5.1.7.3 Register a certificate using RACDCERT

Unfortunately there is more than one format in which a digital certificate can be stored, and not all versions of all Web servers and browsers support the same selection of formats. A prime example is that OS/390 V2R7 and earlier releases do not support the format known as Public Key CryptoSystem (PKCS) #12, whereas the popular browsers use precisely this format. Therefore, you may need to obtain your client certificate in the PKCS#7 format supported by older releases of OS/390. OS/390 V2R8 and later releases support PKCS#12.

You can obtain a copy of your certificate in any format from the certification authority. Usually this is done using a browser. To download a file in PKCS#7 format you have to go to your certification authority home page, do a search for your valid certificates, and download your certificate again with this option.

We show how to obtain a PKCS#7 format certificate from the VeriSign Web site:

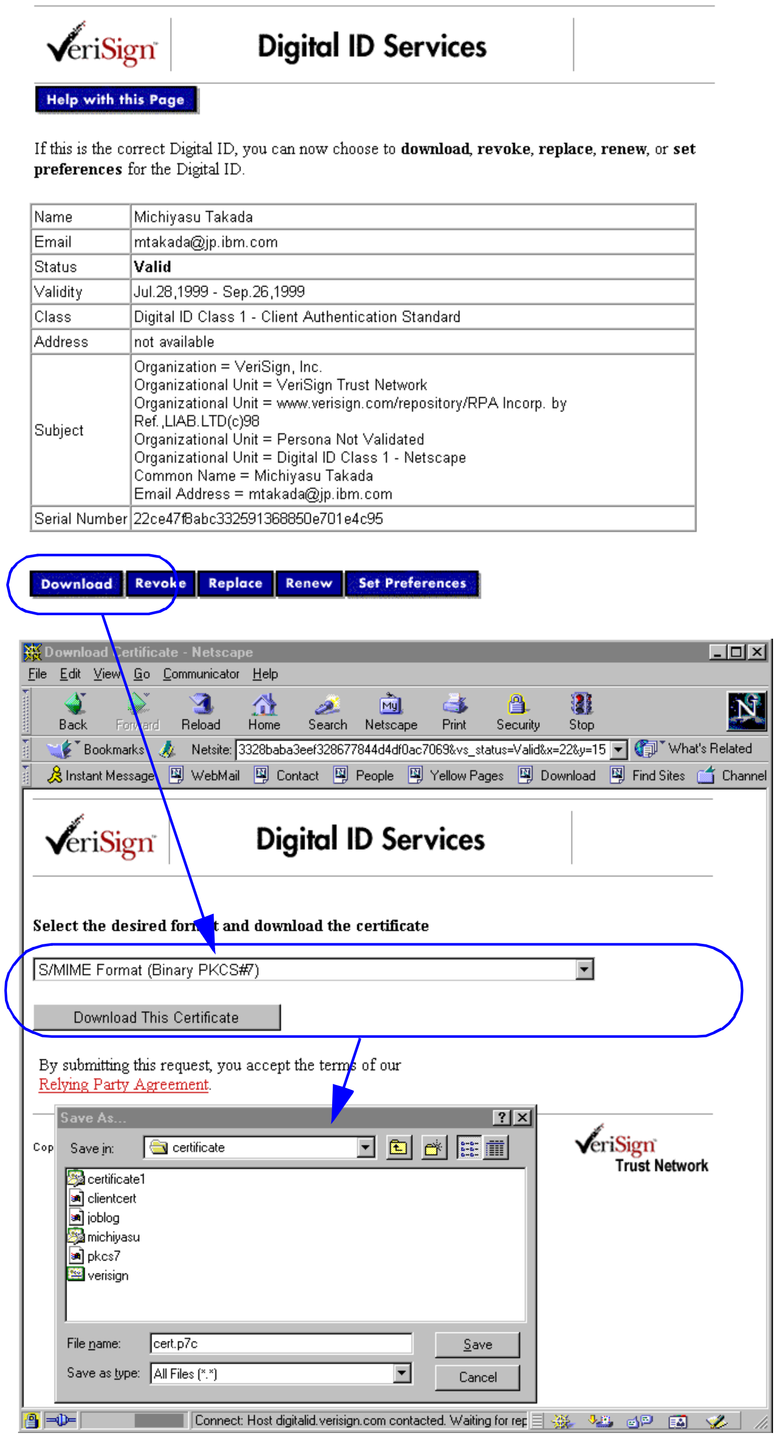
```
https://digitalid.verisign.com/services/client/index.html
```



Search by E-mail Address (recommended):

Enter the E-mail Address: (example: john_doe@verisign.com)	mtakada@jp.ibm.com
Search for Digital IDs that are:	[✓ Valid] Expired All Revoked Pending

Figure 63. Search for digital ID



If this is the correct Digital ID, you can now choose to **download**, **revoke**, **replace**, **renew**, or **set preferences** for the Digital ID.

Name	Michiyasu Takada
Email	mtakada@jp.ibm.com
Status	**Valid**
Validity	Jul.28,1999 - Sep.26,1999
Class	Digital ID Class 1 - Client Authentication Standard
Address	not available
Subject	Organization = VeriSign, Inc. Organizational Unit = VeriSign Trust Network Organizational Unit = www.verisign.com/repository/RPA Incorp. by Ref.,LIAB.LTD(c)98 Organizational Unit = Persona Not Validated Organizational Unit = Digital ID Class 1 - Netscape Common Name = Michiyasu Takada Email Address = mtakada@jp.ibm.com
Serial Number	22ce47f8abc332591368850e701e4c95

Download
Revoke
Replace
Renew
Set Preferences
Download Certificate - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks
Netsite: 3328baba3eef328677844d4df0ac7069&vs_status=Valid&x=22&y=15
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channel
VeriSign
Digital ID Services

Select the desired format and download the certificate

S/MIME Format (Binary PKCS#7)

Download This Certificate

By submitting this request, you accept the terms of our Relying Party Agreement.

File name: cert.p7c

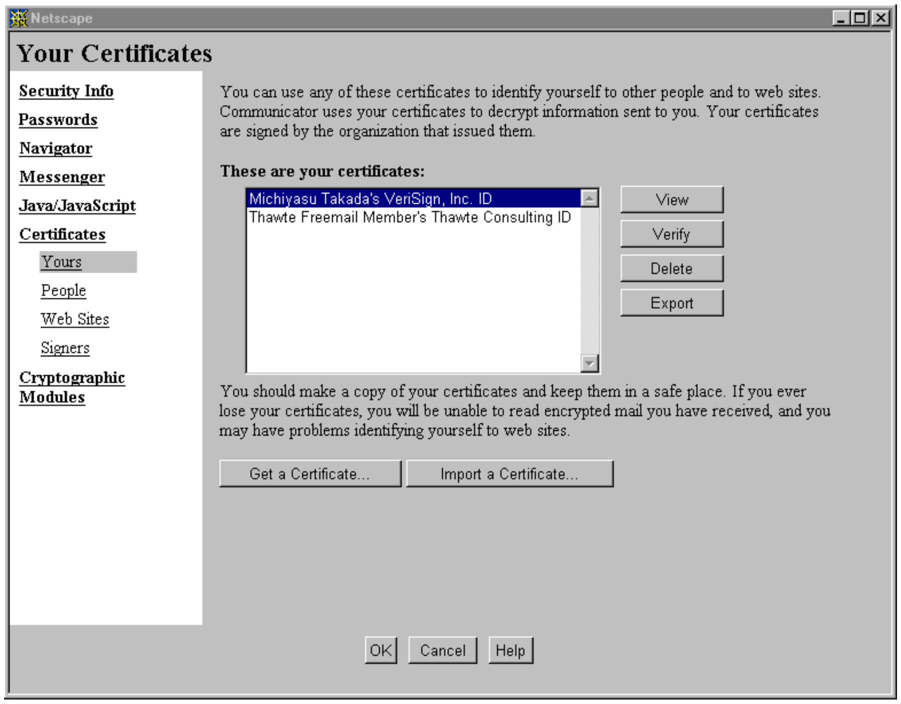
Save as type: All Files (*.*)

Connect: Host digitalid.verisign.com contacted. Waiting for rep

Figure 64. Download a certificate in PKCS#7 format from VeriSign Web site

You can search for your certificate by e-mail address, name or Digital ID Serial Number and Issuer Name at the Web page shown in Figure 63 on page 124. Then the Web page that contains certificate information that met the criteria appears. From this page, you can download various certificate formats including PKCS #7 format (Figure 64).

If your OS/390 system is V2R8 or later, you can use PKCS#12 format to register your certificate into the RACF database. In this case, we show (Figure 65) how you can simply extract your certificate in PKCS#12 format from the Netscape browser.



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.

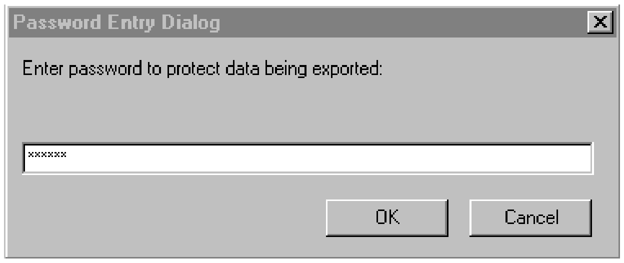
These are your certificates:

Michiyasu Takada's VeriSign, Inc. ID
Thawte Freemail Member's Thawte Consulting 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.

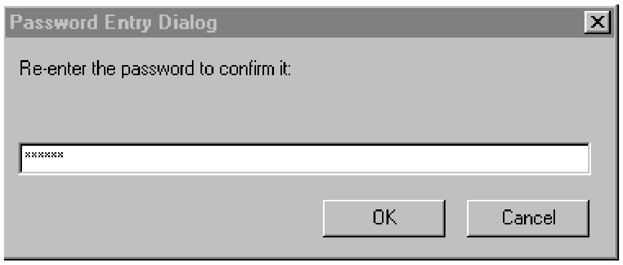
Figure 65. Exporting your client certificate in PKCS#12 format

First click the **Security** button on the **Navigation Toolbar**. After the **Your Certificates** window shown in Figure 65 appears, click **Yours** under **Certificates**. Then, under **These are your certificates**, select the client certificate you desire and click **Export**. Now you are required to enter a password for your browser's certificate database, which you originally specified in Figure 61 on page 121.



Enter password to protect data being exported:

******

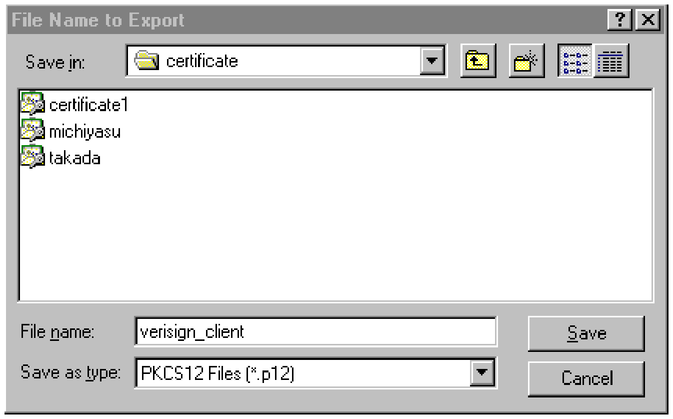


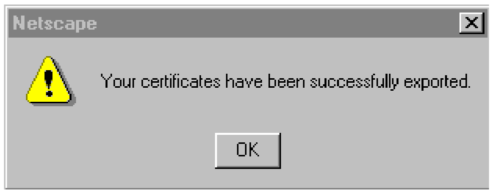
Re-enter the password to confirm it:

******

Figure 66. Setting a password for your certificate

Then you are required to enter a password for your certificate. This password is required later to issue the RACF command `RACDCERT ADD`.





Your certificates have been successfully exported.

Figure 67. Storing your certificate in your workstation

Finally store your exported certificate in a local file. You will need to transfer this file to an MVS data set as a binary file, using or NFS or some other means.

To use the `RACDCERT` command the file must be in an MVS sequential data set. We recommend you allocate a five to ten track sequential data set using the following DCB attributes:

DSORG=PS
RECFM=VB
LRECL=4096
BLKSIZE=27998

Register the certificate using the `RACDCERT` TSO command:

```
racdcert id(takada) add('takada.certif.pkcs12') trust password('password')
```

Figure 68. Register a certificate in the RACF database

This example registers the certificate stored in the sequential data set TAKADA.CERTIF.PKCS12 with the user ID TAKADA. If you registered a certificate in PKCS #12 format and forgot to specify the password option, you will see the following message.

```
IRRD127I The data set contains a PKCS12 encrypted certificate. The PASSWORD
keyword must be specified to process the certificate. The certificate is not
processed.
```

This password is the one specified in Figure 66 on page 126. If you are using PKCS #7 format, you do not have to specify any password for your certificate.

You will see the following messages when the request has been completed successfully.

```
IRRD119I Certificate Authority not defined to RACF. Certificate added with
TRUST status.
```

The message is saying that you are adding a certificate to RACF and RACF does not know about the certificate.

You may now list your certificate issuing the `RACDCERT LIST` command as shown in Figure 69.

```
RACDCERT ID(TAKADA) LIST

Digital certificate information for user TAKADA:

  Label: Michiyasu Takada's VeriSign, Inc
  Status: TRUST
  Start Date: 1999/07/27 20:00:00
  End Date:   1999/09/26 19:59:59
  Serial Number:
       >22CE47F8ABC332591368850E701E4C95<
  Issuer's Name:
       >CN=VeriSign Class 1 CA Individual Subscriber-Persona Not Validated.OU<
       >=www.verisign.com/repository/RPA Incorp. By Ref.,LIAB.LTD(c)98.OU=Ver<
       >iSign Trust Network.O=VeriSign, Inc.<
  Subject's Name:
       >mtakada@jp.ibm.com.CN=Michiyasu Takada.OU=Digital ID Class 1 - Netsca<
       >pe.OU=Persona Not Validated.OU=www.verisign.com/repository/RPA Incorp<
       >. by Ref.,LIAB.LTD(c)98.OU=VeriSign Trust Network.O=VeriSign, Inc.<
  Private Key Type: Non-ICSF
  Private Key Size: 512
  Ring Associations:
  *** No rings associated ***
```

Figure 69. Displaying certificate information associated with RACF user ID

5.1.7.4 Certificate self-registration with RACF

You can register your certificate in the RACF database without issuing the `RACDCERT` command. You may use the application to enable autoregistration of digital certificates, RACF Self-Registration (RAR). For detailed information regarding how to set up RAR, see *Ready for e-business: OS/390 Security Server Enhancements*, SG24-5158. A sample window from the application is shown in Figure 70 on page 129.

Storing client certificates in the RACF database is an excellent way for creating a one-to-one mapping between the client certificates of employees and their user IDs, but it does not scale well in the case where, for instance, large numbers of customer certificates need to be mapped to a small number of surrogate user IDs.

5.1.7.5 Certificate name filtering

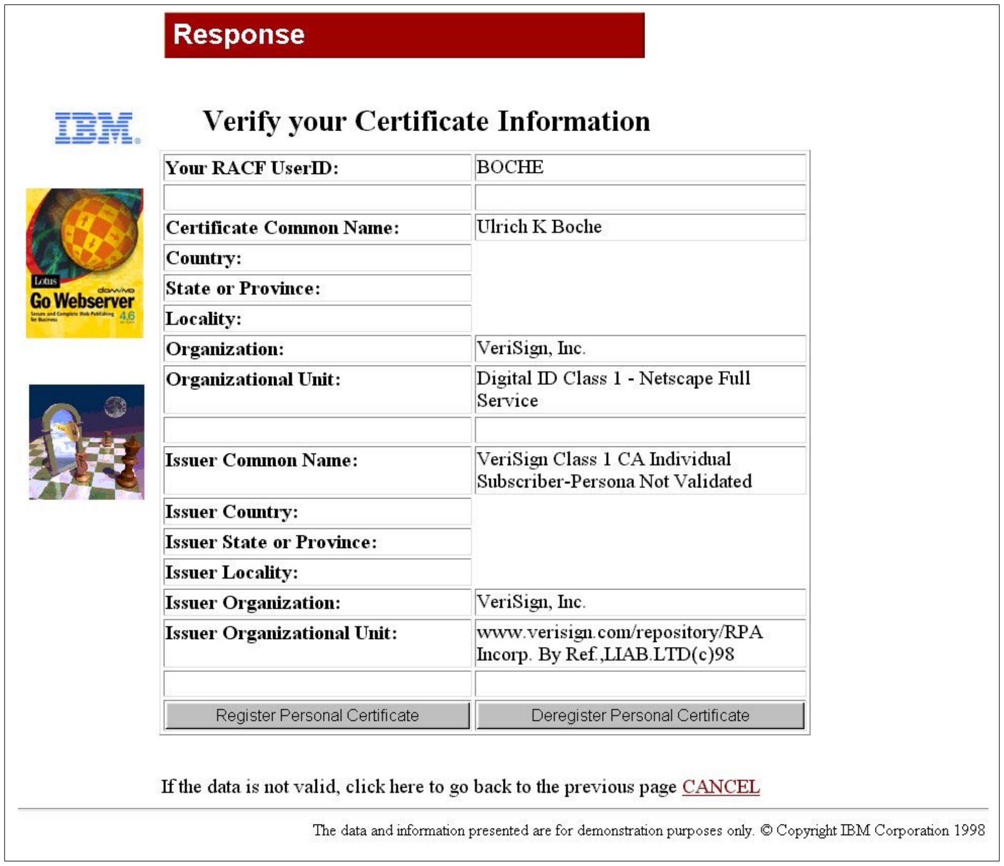
APARs OW40129 and OW40130 for OS/390 provide support for associating RACF user IDs with client certificates, called certificate name filtering.

Using certificate name filtering, client certificates are not stored in the RACF database. The association between one or more certificates and a RACF userid is achieved by defining a filter rule that specifies parts of the distinguished name of the certificate owner and/or issuer (CA) that need to be matched for the userid to be assigned. A sample filter rule might look like the following:

```
RACDCERT ID(DEPT3USR) MAP SDNFILTER(‘OU=DEPT3.OU=NY.OU=Sales.O=XYZ Co’)
```

This sample filter rule would associate userid DEPT3USR with all certificates when the distinguished name of the owner has the organization “XYZ Co” and the organizational units “DEPT3”, “NY”, and “Sales” in it.

For applications like the OS/390 Web server, the TN3270 server, or the FTP server in z/OS V1R2 which can call the RACF interface to get a userid that is associated with a client certificate, it is transparent whether the certificate was stored in the RACF database or a certificate name filter rule was used.



Response

Verify your Certificate Information

Your RACF UserID:	BOCHE
Certificate Common Name:	Ulrich K Boche
Country:	
State or Province:	
Locality:	
Organization:	VeriSign, Inc.
Organizational Unit:	Digital ID Class 1 - Netscape Full Service
Issuer Common Name:	VeriSign Class 1 CA Individual Subscriber-Persona Not Validated
Issuer Country:	
Issuer State or Province:	
Issuer Locality:	
Issuer Organization:	VeriSign, Inc.
Issuer Organizational Unit:	www.verisign.com/repository/RPA Incorp. By Ref.,LIAB.LTD(c)98
Register Personal Certificate	Deregister Personal Certificate

If the data is not valid, click here to go back to the previous page CANCEL

The data and information presented are for demonstration purposes only. © Copyright IBM Corporation 1998

Figure 70. Certificate self-registration application sample window

5.1.8 Retrieving LDAP information

As discussed in 2.6.1.2, “Authentication with the OS/390 LDAP server” on page 44, centralized user management is one of the key roles LDAP was designed to provide. That includes support for the synchronization of user IDs and passwords. In Lotus Domino Go Webserver for OS/390 Release 5.0, LDAP support, that is to say the ability to access the LDAP server for user authentication, was introduced. This means that if authentication information (such as a set of user IDs and passwords or user certificates) has been stored in the LDAP directory, users can use this information to access protected resources in IBM HTTP Server. This authentication information can be used not only by IBM HTTP Server for OS/390 but also by any other LDAP-enabled applications for the purpose of user authentication. In addition, using LDAP support allows multiple IBM HTTP Servers for OS/390 to share configuration information.

IBM HTTP Server LDAP support is extremely scalable. You have a choice of LDAP configurations including:

- A single LDAP server.

- High availability configurations using multiple LDAP servers, for example, primary, secondary, and so on.
- Different LDAP servers to be accessed for different requests. For example, requests may come from two different IP addresses, and the Web server could contact a different LDAP server for each one.

You do not necessarily need to select OS/390 LDAP Server as your repository for IBM HTTP Server LDAP support. Any other LDAP servers can interact with IBM HTTP Server. However, as mentioned in 2.6.1, “The OS/390 LDAP server” on page 43, OS/390 LDAP Server provides various scalable features like exploiting DB2 and/or RACF back-end server, Parallel Sysplex support, connection optimization and so forth. Therefore, in an environment where many different servers exploit LDAP for authentication, it is better for you to implement OS/390 LDAP Server for centralized user management.

In the following example, we will show you how to extract user IDs and passwords stored in the LDAP server.

5.1.8.1 Configuring LDAP on IBM HTTP Server

To provide the server with information about the LDAP servers in which to store information, you have to use the LDAPInfo directive. Storing information on an external LDAP server allows applications to share the same information.

```
LDAPInfo ITSOLDAPSRV 1 {
    Host 9.24.104.113 2
    Transport TCP  3
    ClientAuthType Basic 4
    ServerAuthType Basic    5
    ServerDN "cn=HTTP Server,ou=ITSO,ou=Raleigh,o=IBM_US,c=US" 6
    ServerPasswordStashFile /u/takada/ldap/ldappasswd 7
    UserNameFilter "(&(objectclass=organizationalPerson)(cn=%v1* %v2*))" 8
    UserSearchBase "o=IBM_US, c=US" 9
}
```

Figure 71. Specifying the LDAP server information

1 This is the name you want to associate with this LDAP server setup. The name can then be used by subsequent Protection directives to point to this LDAP setup.

2 Specify the host name of the LDAP server you want to exploit for authentication.

3 Specify the port number on which the LDAP server listens. The default port is 389 for non-SSL-transported connections, and 636 for SSL-transported connections.

4 Specify the authentication type for connections made on behalf of the client. Basic means the client must provide a user ID and password in order to authenticate. Another option is Cert, meaning that the client’s certificate is used for authentication.

5 Specify the authentication type for the server’s connection to the LDAP server. Basic means that if IBM HTTP Server logs in to the LDAP server, it uses its distinguished name from the ServerDN subdirective, and the password provided in the password stash file. If you specify None, the LDAP server will allow anonymous access.

6 This is the distinguished name of the HTTP Server. This name is used when accessing the LDAP server when ServerAuthType is set to Basic. You have to store this distinguished name in the LDAP server.

7 This is the file containing the encrypted password to access the LDAP server. The password is only used if ServerAuthType is Basic. Create the stash file using the `htadm` command or using the Configuration and Administration forms. The syntax of the `htadm` command to create a stash file is:

```
htadm -stash file_name password
```

This password must be registered as an attribute (normally userPassword attribute) in the LDAP server.

8 Specify the filter used to convert the username as input by the user to a search filter for an LDAP entry. The default is "(&(objectclass=person)(cn=%v1* %v2*))" where %v1 and %v2 are the words typed by the user. For example, if the user types "Cameron Diaz" in the window shown in Figure 46 on page 108, the resulting search filter would be "(cn=Cameron* Diaz*)".

9 Specify the starting point for the LDAP server to search for user names.

If you have multiple LDAP servers and want to use those for authentication, you can specify multiple LDAPInfo directives. Of course, each label must be unique (**1**).

When you have specified the LDAPInfo directive shown in Figure 71 on page 130 and the LDAP server is running, IBM HTTP Server tries to connect to the LDAP server at initialization time. If IBM HTTP Server succeeds in connecting to the LDAP server, you will see the following message in the trace obtained by specifying the -vv option in the server's start procedure:

```
Communications established successfully with LDAP servers.
```

If you do not have a -vv trace, you can check if IBM HTTP Server can connect to the LDAP server by using the `onetstat -c` command shown in the following screen:

```
TAKADA @ RA03:/u/takada>onetstat -p TCPIPA -c|grep WEBSRV
WEBSRV   00000C1D 172.16.250.3..1051     9.24.104.113..389      Establish
WEBSRV   00000C20 9.24.104.113..443      0.0.0.0..0             Listen
WEBSRV   00000C1F 9.24.104.113..80       0.0.0.0..0             Listen
```

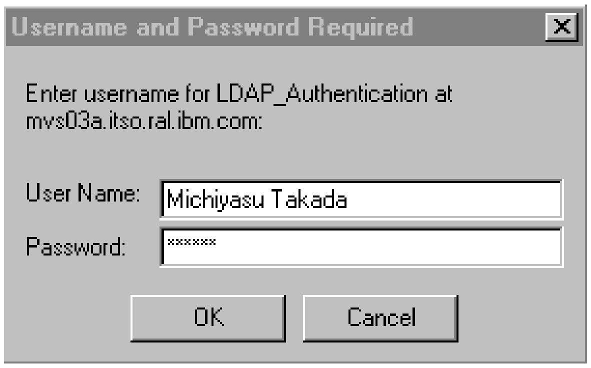
5.1.8.2 How to use authentication information stored in LDAP

Figure 72 on page 132 shows how to control access to resources with the label "CGI-Program" by using LDAP. You will see the new keyword "%%LDAP%% in the PasswdFile and GroupFile subdirectives in this example. This means that IBM HTTP Server uses the LDAP directory for authentication.

```
Protection CGI-Program {
      ServerId        LDAP_Authentication 1
      PasswdFile      "%%LDAP:ITSOLDAPSRV%%" 2
      GroupFile       "%%LDAP:ITSOLDAPSRV%%" 3
      Mask            "cn=OS390 IP Security, ou=Groups, o=IBM_US, c=US" 4
}

Protect /cgi-bin/* CGI-Program INTERNAL
```

Figure 72. Using LDAP for authentication against protected resources

1 Specify the realm name. You can see the value you specified here in the window, which prompts users to enter their user IDs and passwords:


Username and Password Required

Enter username for LDAP_Authentication at mvs03a.itso.ral.ibm.com:

User Name: Michiyasu Takada

Password: ******

Figure 73. Netscape's window for entering a user name stored in LDAP

The user name entered in the window shown in Figure 73 is translated into "cn=Michiyasu Takada" based on the UserNameFilter specified in Figure 71 on page 130 (**8**). A password entered here must be registered in the LDAP directory.

2 This means that user IDs and passwords are stored in the LDAP directory. The syntax is `%%LDAP[:PrimaryLdapServer[, SecondaryLdapServer]]%%`. PrimaryLdapServer and SecondaryLdapServer match LDAP servers defined in LDAPInfo directives. The LDAPInfo directive must precede any %%LDAP%% references in the configuration file.

3 You can also specify the LDAP servers that you want to use for group information. The syntax is the same as that of PasswdFile. Even if the common name and password a user entered are correct, if that common name is not registered in the group entry specified in any mask subdirective (DeleteMask, GetMask, Mask, PostMask, and PutMask) and stored in the LDAP server specified in the GroupFile subdirective, IBM HTTP Server will deny this request.

4 This means that the common name a user entered must be contained in the group entry "cn=OS390 IP Security, ou=Groups, o=IBM_US, c=US" stored in the LDAP directory specified in the GroupFile directive.

Note

While it is possible to use LDAP for user authentication in IBM HTTP Server for OS/390, this option should be considered with great care. For each request for a resource protected by a Protection directive that specifies LDAP, the Web server needs to contact the LDAP server for authentication. For a large number of requests, this can create considerable overhead.

If RACF is used for authentication, OS/390 UNIX will cache the security environments of users that have already been authenticated in main storage. Therefore, re-authentication of a user can be done with minimal overhead.

For all users which are defined in RACF, use %%SAF%% for authentication. LDAP can be considered for users that are not RACF defined but only defined in the LDAP directory. If the authentication frequency is high, methods for saving the authentication state of users should be considered to limit the overhead involved in LDAP authentication.

5.1.8.3 Creating user entries in the OS/390 LDAP server

In this section, we will show how to create the LDAP user entries and group entry. In this example, we used the OS/390 LDAP Server. However, we do not show how to set up this server because that is beyond the scope of this book. If you want to know how to set up the OS/390 LDAP Server, see *OS/390 Security Server LDAP Server Administration and Usage Guide*, SC24-5861.

You can use the ldif2db utility to load entries from a file in LDAP Directory Interchange Format (LDIF) into a directory. The database must already exist and so must the suffixes under which new entries are being added. The ldif2db utility may be used to add entries to an empty directory database or to a database that already contains entries.

The syntax is:

```
ldif2db -i <input file> -f <configuration file>
```

No additional parameters other than the filenames of the LDIF input file and the LDAP server configuration file are necessary since all other information (such as the suffix) is contained in the LDIF file.

The LDIF format is used to convey directory information or a description of a set of changes to directory entries. An LDIF file consists of a series of records separated by line separators. A record consists of a sequence of lines describing a directory entry or a sequence of lines describing a set of changes to a single directory entry. An LDIF file specifies a set of directory entries or a set of changes to be applied to directory entries but not both.

The following is an example of creating an input file in the LDIF format and then executing the ldif2db utility to load entries into the LDAP directory. We added four entries to our existing LDAP directory in order to use it for IBM HTTP Server authentication.

```
dn: cn=OS390 IP Security, ou=Groups, o=IBM_US, c=US                           2
objectclass: groupOfNames
description: OS/390 IP Security Team
cn: OS390 IP Security
member: cn=Otto Zech, ou=ITSO, ou=Raleigh, o=IBM_US, c=US
member: cn=Jorge Rivera, ou=ITSO, ou=Raleigh, o=IBM_US, c=US
member: cn=Michiyasu Takada, ou=ITSO, ou=Raleigh, o=IBM_US, c=US

dn: cn=Otto Zech, ou=ITSO, ou=Raleigh, o=IBM_US, c=US                         1
objectclass: organizationalPerson
cn: Otto Zech
sn: Zech
title: OS/390 UNIX System Services Security
userPassword: secret

dn: cn=Jorge Rivera, ou=ITSO, ou=Raleigh, o=IBM_US, c=US                      1
objectclass: organizationalPerson
cn: Jorge Rivera
sn: Rivera
title: OS/390 IPSec
userPassword: secret

dn: cn=Michiyasu Takada, ou=ITSO, ou=Raleigh, o=IBM_US, c=US                  1
objectclass: organizationalPerson
cn: Michiyasu Takada
sn: Takada
title: OS/390 UNIX Application Security
userPassword: secret
```

Figure 74. Input File in the LDIF format for creating the LDAP entries

1 We created these entries as user accounts to access the resources with a label "CGI-Program" within IBM HTTP Server (see Figure 72 on page 132). When you are prompted to input your user ID and password, you have to input a common name (cn, for example, Otto Zech) and a password specified in the userPassword attribute. Then the common name you entered is applied to the rule specified in th UserNameFilter subdirective, that is, "(&(objectclass=organizationalPerson)(cn=%v1* %v2*))". Finally, IBM HTTP Server looks for this entry in the LDAP directory based on the UserSearchBase subdirective (for example, "o=IBM_US, c=US") and checks if a password you entered corresponds to a value in the userPassword attribute.

2 In this example, we also specified the %%LDAP%% key word in the GroupFile subdirective (see Figure 72 on page 132). Therefore, we created a group entry permitted to access restricted resources. This distinguished name must coincide with one specified in the Mask subdirective. In addition, the user common name you entered from your browser must be included in this group entry.

```
TAKADA @ RA03:/u/takada/ldap>export STEPLIB=DB2V510.SDSNLOAD 1
TAKADA @ RA03:/u/takada/ldap>ldif2db -i ourteam.ldif -f /etc/ldap/slapd.conf 2
GLD0010I Reading configuration file /etc/ldap/slapd3.conf.
GLD0010I Reading configuration file /etc/ldap/slapd.at.system.
GLD0010I Reading configuration file /etc/ldap/slapd.cb.at.conf.
GLD0010I Reading configuration file /etc/ldap/slapd.at.conf.
GLD0010I Reading configuration file /etc/ldap/slapd.oc.system.
GLD0010I Reading configuration file /etc/ldap/slapd.cb.oc.conf.
GLD0010I Reading configuration file /etc/ldap/slapd.oc.conf.
GLD0010I Reading configuration file /etc/ldap/pagent_at.conf.
GLD0010I Reading configuration file /etc/ldap/pagent_oc.conf.
GLD0052I Configuration read securePort 636.
GLD0053I Configuration read security of none.
GLD0054I Value connectionsAllowed is set to 1.
GLD0040I The value for the 'maxthreads' optoin is out of range (10, 75000). The
default (75000) will be used.
GLD0129I The value for the 'maxConnections' option is out of range (0, 2047). Th
e default (2047) will be used.
GLD0130I The value for the 'waitingthreads' option is out of range (10, 75000).
The default (75000) will be used.
GLD2062E The LDAP Server will operate in single-server mode.
GLD2004I ldif2db: 4 entries have been successfully added out of 4 attempted. 3
```

Figure 75. Executing the ldif2db utility

Figure 75 shows how to execute the ldif2db utility. Before issuing the ldif2db utility, you have to define the DB2 load library in the STEPLIB environment variable if you have not specified the library to LNKLST (**1**). Then you can execute the ldif2db utility as shown in **2**. We used a file "/u/takada/ldap/ourteam.ldif" as input. If you see the message shown in **3**, you have succeeded in loading entries specified in an input file in the LDIF format.

Now you can use user entries stored in the LDAP directory for authentication. As mentioned earlier, these user entries can be used for authentication not only by IBM HTTP Server but also other LDAP-enabled applications such as WebSphere, Apache, and so on.

Note

In Lotus Domino Go Webserver 5.0.0 and IBM HTTP Server for OS/390 V5.1, there is a problem where the password entered from the browser is never accepted by the LDAP server, even though the user ID and password are correct. In our test environment (OS/390 V2R8, IBM HTTP Server for OS/390 V5.2), we experienced this problem. IBM HTTP Server (or DGW) encrypts the password and passes this encrypted password to the LDAP server. However, the password cannot be stored encrypted in the LDAP server because the application (most likely) needs a clear password for authentication. That is why the IBM HTTP Server cannot authenticate using the LDAP server. APARs to fix this problem are available:

- DGW 5.0.0 and IBM HTTP Server 5.1 APAR PQ27178
- IBM HTTP Server 5.2 APAR PQ28035

For more detailed information about LDAP support, refer to *HTTP Server Planning, Installing, and Using*, SC31-8690.

5.1.9 Conclusion

These are our recommendations for using IBM HTTP Server for OS/390 in a secure manner:

1. Intranet scenario

   If users in the intranet access your company's common information such as company news, new product news, yellow pages and so forth, it is sufficient to create surrogate user IDs, for example, "INTERNAL". These user IDs should not have authority to access any resources beyond what is needed for the service they are supposed to provide. Consider to define these user IDs as RESTRICTED in RACF.

   If you want to enable only certain users to access certain resources, specify the Protect and Protection directives (or the DefProt directive) in the server configuration file and instruct users to enter their RACF user ID.

   When you let users access confidential information such as employee pay data, evaluation data and so forth, you should encrypt this type of data using SSL. Furthermore, if possible, you should use client authentication and associate the client certificates with their RACF user IDs using the `RACDCERT` TSO command or the self-registration application. This enables users to access sensitive data securely without entering their passwords.

2. Internet scenario

   If you want your Web server to be open to the public, you should create surrogate user IDs, for example, "EXTERNAL". Similar to the intranet case, these user IDs should have as little authority as possible. By creating surrogate user IDs, when users on the Internet want to access public information such as press releases and marketing information, they do not need passwords. On the other hand, if users need to transmit their confidential personal data such as credit card numbers, account numbers and so on, the connection between users and your Web server should be encrypted with SSL. The use of client certificates in an Internet environment requires careful planning, especially if large numbers of clients are involved. You should get your server certificate not by self-signing but from an external CA.

5.2 TN3270 server

This section is divided in accordance with areas of TN3270 configuration relating to security:

- Connection security (SSL)
- Mapping control
- Resource overrun control

The TN3270 server supports both TN3270 and TN3270E (TN3270 Enhancements, RFC 2355) protocols. Unless otherwise noted, the term TN3270 will be considered to encompass both TN3270 and TN3270E. From a security perspective, TN3270 and TN3270E can be considered identical (the differences are in the data stream functions supported).

TN3270E?

Do not confuse TN3270E with Extended Data Stream. The E stands for Enhanced. TN3270E or TN3270 can both negotiate EDS.

At times, this section deals with parameters coded in the TCP/IP profile data set. Since it would be cumbersome to make a specific reference for all instances, the reader is directed to the following two manuals for clarification of any of the material presented:

- *IBM Communications Server for OS/390 IP Configuration Reference*, SC31-8726
- *IBM Communications Server for OS/390 IP Configuration Guide*, SC31-8725

Since this redbook is not an implementation manual, you will not find detailed syntax or examples in this text.

5.2.1 Connection security (SSL)

Beginning with CS for OS/390 V2R6, the TN3270 server has supported Secure Sockets Layer (SSL) Version 3.0. SSL V3.0 provides secure data transmission between TN3270 server and an SSL-enabled TN3270 client. SSL V3.0 also enforces server authentication; client authentication is optional. Client authentication support was added in CS for OS/390 V2R8 IP and it requires SSL V3.0 and higher. See 2.4, “Secure Sockets Layer” on page 36 for complete details.

Starting with OS/390 V2R10, negotiated TN3270 (IETF Internet Draft TLS-based Telnet Security) is also supported. For more information on this protocol, see 2.5, “Transport Layer Security protocol (TLS)” on page 41.

The TN3270 server uses GSKKYMAN or the RACF `RACDCERT` for certificate management (either is acceptable). For information on creation and control of certificates, please see Appendix D, “OS/390 certificate management using RACF” on page 321 for details on certificate management for TN3270.

To access a keyring, the TCP/IP profile statement required is the KEYRING statement. The KEYRING statement can be used to point to an MVS, HFS or SAF controlled key file.

Warning

TN3270 ring file read routines may alter the contents of a ring file. An incompatibility exists with the read routines in TN3270 for OS/390 V2R8 and newer releases. If you are testing a newer release and the TN3270 server reads the ring file, this file will no longer be compatible should you wish to fall back to the OS/390 V2R8 TN3270 server. If migrating from OS/390 V2R8, be sure to keep a backup copy of the ring file as a fail safe.

5.2.1.1 Encryption features

See Table 9 for information on specific encryption capabilities and FMID levels to support TN3270.

Table 9. Encryption feature for TN3270 SSL support with SSL V3.0 clients

Encryption feature	Element for SSL V3 client
Base OS/390 V2R6: JTCP350 OS/390 V2R7: JTCP370 OS/390 V2R8: JTCP380	NULL SHA/MD5/NULL
Level 1 OS/390 V2R6: JTCP35T OS/390 V2R7: JTCP373 OS/390 V2R8: JTCP383	RC2/RC4 MD5 NULL SHA/MD5/NULL
Level 2 OS/390 V2R6: JTCP35L OS/390 V2R7: JTCP372 OS/390 V2R8: JTCP382	DES SHA RC2/RC4 MD5 NULL SHA/MD5/NULL
Level 3 OS/390 V2R6: JTCP35K OS/390 V2R7: JTCP37K OS/390 V2R8: JTCP38K	Triple DES SHA DES SHA RC2/RC4 MD5 RC4 SHA NULL SHA/MD5/NULL
Note: Since OS/390 V2R10, the TN3270 encryption levels available are determined by the System SSL encryption feature levels, not Communication Servers feature level.	

When Level 3 is installed, the following cipher suites can be used in a TN3270 SSL connection:

- SSL_NULL_Null
- SSL_NULL_MD5
- SSL_NULL_SHA
- SSL_RC4_MD5_EX
- SSL_RC4_MD5
- SSL_RC4_SHA
- SSL_RC2_MD5_EX
- SSL_DES_SHA
- SSL_3DES_SHA

Using the ENCRYPTION keyword in the TCP/IP profile, the TN3270 can specify the order of preference for these ciphers. The TN3270 server will negotiate the cipher suites in the order specified on the ENCRYPTION statement. If no cipher suite can be agreed upon, the session request will fail.

5.2.1.2 Single port control

The most notable benefit of negotiated TN3270 is the ability to use a single port for TN3270 access. Prior to OS/390 V2R10, a TN3270 port would either assume that all incoming connections required SSL or no incoming connections would use it. This was accomplished by the following statements in the TCP/IP profile:

```
TELNETPARMS
```

```
   PORT 23
   SECUREPORT 2023
ENDTELNETPARMS
```

In this example, users requiring encryption would be forced to change their TN3270 client to user port 2023. Now, using the following statement would allow TN3270 clients to use either a secure or non-secure session.

```
TELNETPARMS
   SECUREPORT 23
   CONNTYPE ANY
ENDTELNETPARMS
```

Since port 23 is being used for either, no port changes would be required at the TN3270 client. See Figure 76 on page 140 for a graphical representation.

When coding SECUREPORT, an installation may choose one of the following control options for their TN3270 port. These options are configured on the CONNTYPE statement in within a TELNETPARMS block in the TCP/IP profile data set:

SECURE The port assumes that all incoming connections will begin the session with an SSL handshake. If a standard SSL handshake fails, a negotiated handshake is tried. No other connection types are allowed. This is the default value for the CONNTYPE statement.

NEGTSECURE Negotiated telnet using IAC START_TLS option. If the client can't negotiated this option, the connection will fail.

ANY SSL is attempted first. If that fails, negotiated SSL is attempted. If that also fails, the session will be established with no encryption.

BASIC No encryption is used. A port defined with the PORT statement can be used only for basic connections.

NONE No connection allowed at all. This parameter only makes sense in the context of a PARMSGROUP. See 5.2.2, "Mapping control" on page 141.

From a security standpoint, both SECURE and NEGTSECURE provide the equivalent level of encryption (that is, they both would negotiated SSL V3.0). However, NEGTSECURE would take longer for session establishment, since the telnet IAC option negotiation is part of the process. Using option ANY could present an even longer session establishment process, as the client and server run through SSL, negotiated SSL and finally regular TN3270 option negotiation. The length of time awaiting an SSL handshake response can be controlled via the SSLTIMEOUT parameter. The default is 5 seconds.

As we will see later, the ideal situation is to use PARMSGROUP to force certain groups of clients to connect only with an encrypted session, while allowing your internal (intranet) users to connect in BASIC. The intention here is to allow all of your telnet users to function via the same well-known telnet port, 23.

Note that multiple secure ports can be defined by having multiple TELNETPARMS blocks with a single SECUREPORT statement in each. However, only one keyring identifier, either a keyring file name or a RACF common keyring name, can be used for all secure ports.

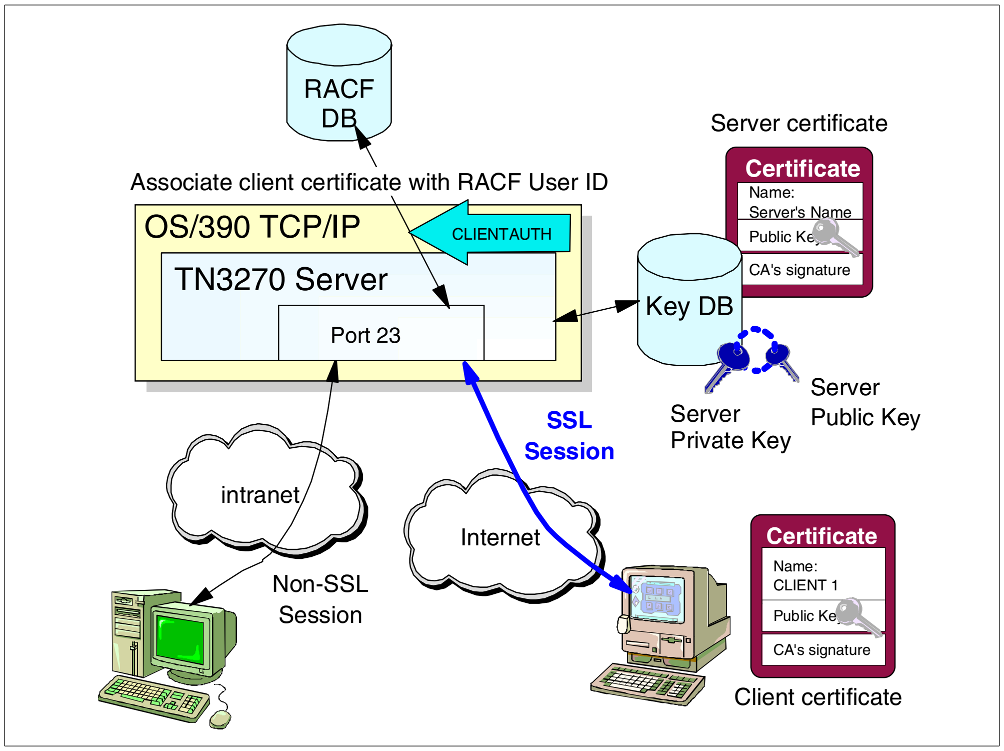


Figure 76. Overview of TN3270 server SSL support

5.2.1.3 TN3270 client authentication

When a secure connection is negotiated, the option exists to require client authentication. Client authentication is also illustrated graphically in Figure 76. Note that if client authentication fails any of the checks, no connection at all will be established -- not even the USSMSG10 screen will be displayed.

There are two different levels of authentication that can be specified on the CLIENTAUTH statement:

SSLCERT Normal SSL client authentication -- the client must send a recognized certificate. See 2.4.2, “Establishing secure communications with SSL” on page 37.

SAFCERT Normal SSL client authentication, plus one required and one optional level of security:

- Using a SAF product such as RACF, a check is made to ensure that the incoming certificate is associated with a SAF user ID. See 2.4.4, “SSL with OS/390 RACF” on page 39 for more information. This allows a granularity of client certificates, ensuring that they are only used by selected user IDs.
- If SERVAUTH class has been activated and a profile for the telnet port (port 23) has been defined, then the user ID must also be permitted to this resource.This can prevent a potential hacker from using a certificate that was perhaps intended for a server other than TN3270. Like the SAF check, this forces more checking with respect to the incoming user ID. The syntax of the resource name is:

```
EZB.TN3270.systemname.tcpname.PORT00023
```

If desired, client authentication can be used in conjunction with certificate revocation lists via an LDAP server. The CRLLDAPSERVER statement in the TCP/IP profile data set points to the LDAP server that contains the revocation information. The LDAP server will be checked on every client authentication. Only certificate revocation list supported is Vault Registry CRLs.

5.2.1.4 Session take over

A brief warning needs to be issued with respect to the TKOSPECLURECON parameter. If a TN3270E client inadvertently loses its connection to the TN3270 server, the TN3270 server may not immediately be aware that the session has been broken. When the client attempts a reconnect, the session is refused because the LU is still allocated. If the TN3270E client knows the LU name, the session can be re-established by a take-over of the specific LU.

Before the session can be re-established, the TN3270 server will send a timemark on the original connection to see if the client responds. The TKOSPECLURECON parameter specifies how long to wait for a response. Do not, under any circumstances, code a value of 0, since this will stop the TN3270 server from checking for the original connection. If a hacker knew the LU name of your session, it could be easily stolen while your session is still active.

This does leave a small but real danger: a potential hacker could disrupt network connectivity long enough to break existing TN3270E sessions. Then, by specifying one of the LU names to a broken session, he could take over an existing connection. This would occur even if TKOSPECLURECON was greater than zero!

If such a threat seems significant on your network, do not use TKOSPECLURECON. Instead, use TKOSPECLU to force the client to enter a user ID and password again. Of course with TKOSPECLU, the LU is freed and the session is lost -- TKOSPECLURECON actually re-sends the last screen displayed in the original session, bypassing a logon altogether.

To make the TN3270 reconnect support more secure, TN3720 SSL client authentication should be used. The TN3270 server will verify that at reconnect the certificate received during the SSL handshake matches the one received on the original connection. If they do not match, the reconnect will not be allowed.

5.2.1.5 SSL client configuration

For information on configuring Host On-Demand for negotiated telnet, see *IBM WebSphere Host On-Demand: Version 5 Enhancements,* SG24-5989. For information on configuration HOD for SSL, see *IBM Secureway Host On-Demand 4.0: Enterprise Communications in the Era of Network Computing,* SG24-2149.

5.2.2 Mapping control

TN3270 mapping control can be used to improve control of TN3270 client access to SNA applications. It enables an installation to identify or classify clients using several different characteristics:

- IP address or sub-network
- Host name or host name group
- Interface

With TN3270E, some further control can be obtained by specifically coding the LU name or LU pool in the TN3270E client configuration file. This allows a specific client to request a specific LU at connection establishment. This LU could then be mapped directly to an IP address and an application name. From a security viewpoint, this does not directly enhance security, since any client could specify the same LU or alter the hard coded LU name in the client configuration file. Of course, it might be difficult for a hacker to determine the right LU name.

Finally, hard coding and maintaining LU names in TN3270E client configuration files represents a significant administrative concern.

Using these aforementioned client characteristics, an installation can use mapping statements to provide highly granular access control. All of these statements are part of the BEGINVTAM section of a TCP/IP profile data set. Let's take a closer look at some of the statements we can use. First let's review the statements used for grouping:

IPGROUP TN3270 clients can be identified by an IP address, a list of IP addresses or a subnetwork and mask combination. In a DHCP environment, use of IP address subnet masking would be preferred.

HNGROUP Like an IPGROUP, only using host names instead of IP addresses. This is a powerful client grouping mechanism. Wildcards can be used in place of any particular qualifier. Further, wildcards can represent multiple qualifiers -- allowing something like:

```
HNGROUP **.IBM.COM ENHNGROUP
```

The double asterisks represents any number of qualifiers.This example would encompass host names such as RALEIGH.IBM.COM and VANCOUVER.CAN.IBM.COM.

LUGROUP TN3270 clients can be associated with (assigned) a group of LU's.

PARMSGROUP This parameter allows a grouping to be defined that encompasses CONNTYPE, CLIENTAUTH, ENCRYPTION and the DEBUG statement. Essentially, allowing an installation to control the security requirements of a specific group of users. We'll discuss this more when we review the PARMSMAP statement.

Now, building on these grouping statements, let's see how we can make some mapping assignments to control TN3270 users:

LUMAP This statement is the fundamental link between a client identifier and a pool of LU's. At connection establishment, the LUMAP is checked. This statement allows a group of LU's to be assigned to either an HNGROUP or an IPGROUP. At the simplest level, it allows an installation to track a group of users (say, your Vancouver location) by mapping their network to a specific pool of LU's. The LUMAP statement can also assign a default application, allowing an installation to ensure that certain clients are not only given a particular LU (or group of LU's), but also limit them to a specific application.

Note

Today, most TN3270 clients will negotiate a TN3270E session. When this is done, LU selection will occur during negotiation after all mapping rules are applied except the optional LU-to-application mapping rule. This mapping rule is not used frequently. If it is used, the SIMCLIENTLU parameter can be coded in TELNETPARMS to defer LU selection until after application selection, allowing the LU-to-application rule to be effective.

ALLOWAPPL There are two significant parameters (from a security perspective) for this statement. One, of course, is the application name to define which applications are allowed to be accessed. The other is the LU name or LU group that specifies which LUs can establish a session with the named application. For example, the LUMAP statement selects and an LU group based upon a client's IP address. When this client selects an application from, for example, an USSMSG10 screen, the ALLOWAPPL statement can permit or deny this LU access to the application.

RESTRICTAPPL Similar in effect ALLOWAPPL but adds the condition that the client must provide a user ID and password to access the application. The user ID must be listed on the RESTRICTAPPL statement and an LU or LU group can optionally be associated with the user ID.

DEFAULTAPPL An installation can map a client group to a specific application directly. Some parameters such as DEFONLY or FIRSTONLY may be used to add further control to the level of security provided by the default statement.

PARMSMAP For an installation that intends upon ensuring appropriate security levels, this statement may be useful. This parameter allows connection parameters defined in a PARMSGROUP block to be applied at the client identifier level. The security settings supported under PARMSGROUP include CLIENTAUTH, ENCRYPTION, and CONNTYPE.

Further information about parameters discussed here can be found in *IBM Communications Server for OS/390 IP Configuration Guide*, SC31-8725 and *IBM Communications Server for OS/390 IP Configuration Reference*, SC31-8726.

5.2.3 Resource overrun control

The TN3270 server also support data overrun protection using the following parameters in TELNETPARMS:

MAXRECEIVE This parameter can be used to protect the server against a client stuck in a send-data loop. You can set the maximum number of bytes received from a client without an End Of Record (EOR) being received. When the server receives enough data to exceed the limit, it will terminate the connection. For a large file transfer, because the sender typically divides the file into smaller records to be sent and the receiver rebuilds the file when all records are received, this setting should not terminate the data transfer.

MAXVTAMSENDQ With this parameter, you can protects against using up large amounts of storage to hold data destined for a VTAM application that is not receiving data. It specifies the maximum number of data segments (RPLs) queued to be sent to VTAM, and when the queue size exceeds the limit, the server will drop the connection.

MAXREQSESS This parameter specifies the maximum number of session requests received by the Telnet server in a 10 second period. A BIND received by Telnet defines a session request. If the number of BINDs received in a 10 second period exceeds the limit, an error is reported. This parameter protects against session logon loops that are possibly created by an automatic CLSDST-PASS to an inactive session.

The MAXREQSESS parameter cannot protect against logon loops caused by an inactive default application and a client using auto-reconnect. The best protection against the auto-reconnect loop is to code the MSG07 statement which keeps the client from being disconnected.

Refer to *IBM Communications Server for OS/390 IP Configuration Guide*, SC31-8725 and *IBM Communications Server for OS/390 IP Configuration Reference*, SC31-8726 for more information.

5.3 UNIX System Services Telnet server

The UNIX System Services Telnet server is used to enable remote Telnet clients to log in to your UNIX shell environment in either raw mode (also called character-at-a-time mode) or line mode.

The UNIX Telnet server is started by InetD for each incoming Telnet connection. When the Telnet session is complete, the Telnet server will exit. Each active Telnet session will have a separate instance of the Telnet server which will communicate with the Telnet client.

Figure 77 on page 145 shows an overview of how the Telnet server is implemented in UNIX System Services.

InetD listens for Telnet connection requests on port 23. Immediately after such a request arrives, InetD clones itself by means of the fork() service and starts the Telnet server program as specified in /etc/inetd.conf.

The Telnet server program enters the initial session setup window with the Telnet client and requests a RACF user ID with a valid OMVS segment and password from the client end user. In this phase, user ID and password are checked. After these are accepted by the current phase of the Telnet server, the Telnet server restarts itself via a new spawn call with a new set of parameters and, in addition, a shell process is started.

Both the Telnet server process and the shell process execute under the end user ID, and not the original user ID with UID=0 assigned to the Telnet server.

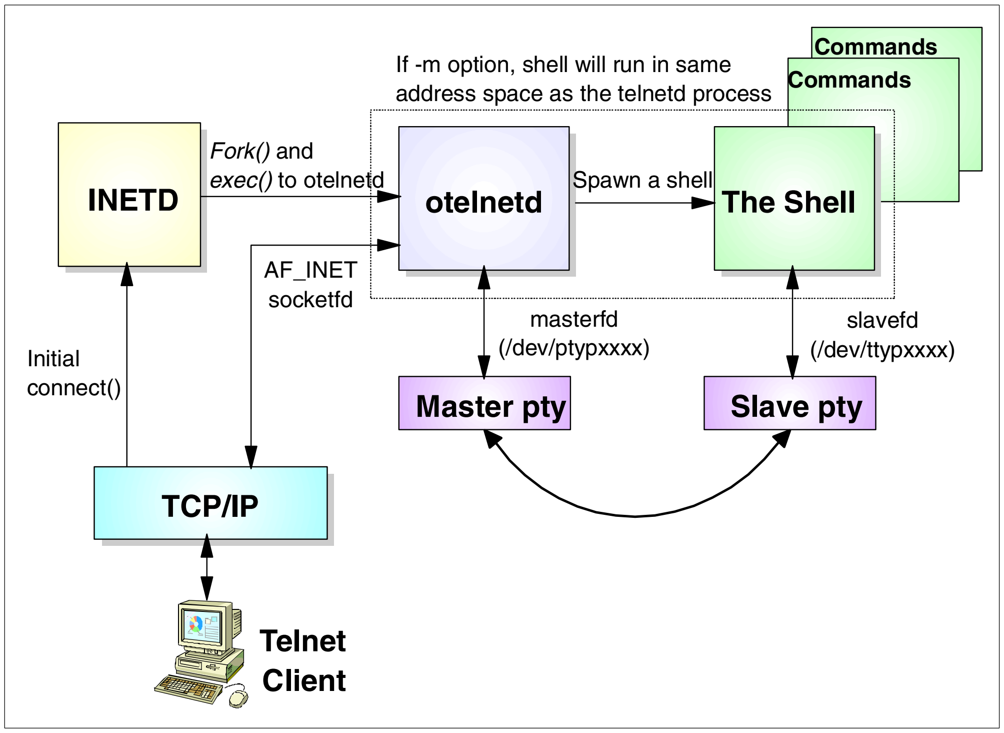


Figure 77. OS/390 UNIX Telnet server overview

Fortunately UNIX System Services does not provide an /etc/passwd file and all authentications for the UNIX Telnet Server are conducted by RACF. So if malicious hackers try to get at the /etc/passwd file on UNIX in order to find user IDs with strong authority such as a superuser, their attempts will end in failure.

OS/390 UNIX Telnet server, however, does not support one-time password systems or SSH. RACF user ID and passwords from a Telnet client are transmitted in clear text. This means that hackers have the possibility of eavesdropping and detecting the OS/390 user ID and password from a trace. Therefore, it is better to use the IP packet filtering function provided by a firewall and permit only trusted users to access the Telnet server. If you permit access from the Internet, you should configure the IPSec function. Since the Telnet server does not support SSL, IPSec is the only encrypted way (and therefore, the only sure way) of protecting Telnet (other than TN3270E) traffic. For more information regarding IP packet filtering and IPSec, see Chapter 2, “TCP/IP security overview” on page 11.

5.3.1 Kerberized UNIX Telnet server support

Kerberos, a network authentication protocol, is designed to provide strong authentication for client/server applications using symmetric key cryptography.

UNIX Telnet become Kerberized in z/OS V1R2 to provide stronger security and allow secure data traffic in a network. This support is implemented using the Kerberos Version 5 APIs.

The user data transferred between a Telnet server and client can be protected by being encrypted with a symmetric key obtained from the Kerberos Version 5 server.

The implementation is based on the following RFCs:

- *RFC 2941 - Telnet Authentication Option*
- *RFC 2942 - Telnet Authentication: Kerberos Version 5*
- *RFC 2946 - Telnet Data Encryption Option*
- *RFC 2952 - Telnet Encryption: DES 64 bit Cipher Feedback*
- *RFC 2953 - Telnet Encryption: DES 64 bit Output Feedback*

For more information about the Kerberos Version 5 protocol, refer to 2.7, “Kerberos-based security system” on page 46.

5.4 OS/390 FTP server

File Transfer Protocol (FTP) is *primarily* used to transfer files between TCP/IP hosts. We say primarily because with CS for OS/390 IP, FTP can also be used to interact with JES and DB2. See 5.4.8, “FTP and JES” on page 158 for complete details. FTP is one of the most powerful IP based applications; it is also one of the most common. As such, it is also one of the areas where security is critical. In order to discuss security in this environment, we’ll begin with a general overview of how FTP functions.

It is appropriate to mention an often overlooked fact about FTP on the OS/390 (and z/OS) platform. Each and every mountpoint in the HFS is nothing more than an MVS data set. As such, recovery and backup of HFS file systems can be managed from the MVS side of the fence. Further, all of the TCP/IP executable files and many of the UNIX System Services executables exist only in the MVS files system. These unique aspects and others (we haven’t really given RACF its dues) of OS/390 help make it an environment that can be considerably less vulnerable to hacking than a standard AIX/UNIX implementation.

For more information of the FTP implementation on CS for OS/390 V2R10 IP and earlier, see *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228.

5.4.1 The ABCs of FTP

An FTP client is considered to be the TCP/IP application that initiates (via CONNECT socket call) the FTP control session. Often, this is done from a command line (either a TSO prompt or a UNIX/DOS prompt). There are, however, many graphical applications that can perform as FTP clients. Most regular Internet users are aware that Web browsers can be used for FTP protocol as readily as for HTTP.

The FTP server is the target TCP/IP application to which the client connects. To enable these incoming connections, the server normally issues a LISTEN() socket call on well-known port 21.

The OS/390 SecureWay Communications Server FTP server and client exploit OS/390 UNIX System Services, providing access to both traditional MVS data sets and OS/390 UNIX System Services Hierarchical File System (HFS) files.

The OS/390 FTP server and client allow users to transfer both MVS data sets and HFS files into and out of an OS/390 system. In addition, functions such as JES and SQL are supported by the OS/390 FTP server (using both MVS data sets and HFS files as the job/SQL source).

The FTP protocol requires FTP to use two different ports. Hence two different connections can be, and often are, active at the same time. One connection is the control connection, over which all control information is exchanged. This connection is active throughout the entire FTP session. All FTP commands are sent via this connection, as are most of the replies. However, the LS and DIR FTP subcommands will receive their replies via a data connection. For the control connection, the well-known port 21 is used by the FTP server (by default).

The second connection used by the FTP server is the data connection. This is the connection over which all data is transferred (and LS and DIR subcommand output, as previously noted). The data connection only exists for the duration of a single file transfer. The well-known port 20 is usually used by the FTP server as the default for the FTP data connection. See 5.4.7, "FTP firewall-friendly" on page 157 for exceptions and implications.

The FTP server does not use InetD as a listener process but has its own listener program, the FTP daemon address space. See Figure 78 for an overview of the FTP server implementation in OS/390 UNIX System Services.

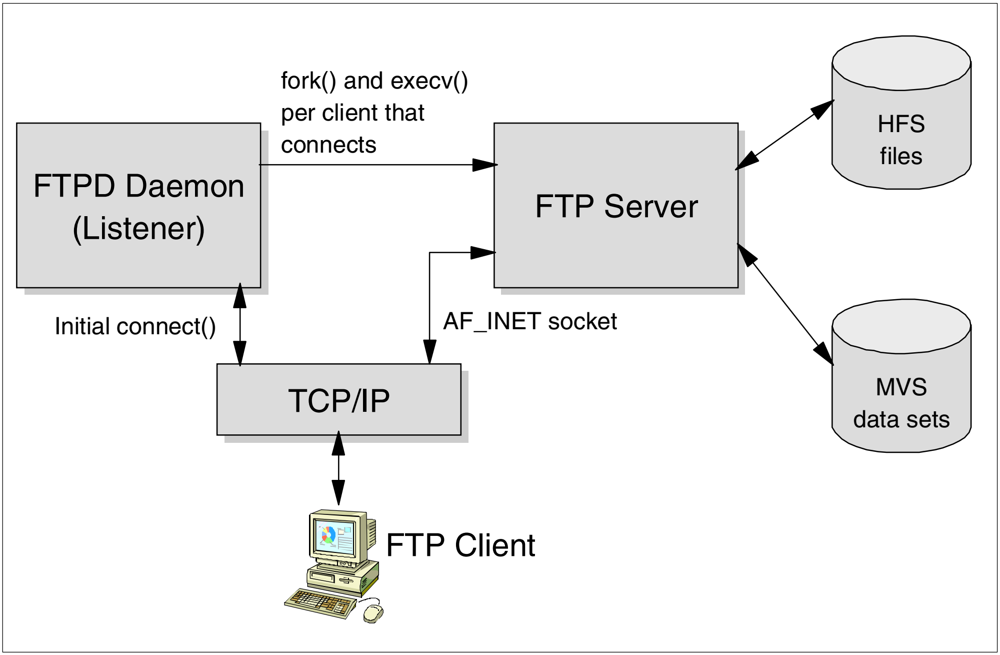


Figure 78. FTP server implementation overview

5.4.2 FTP and job names

Figure 79 provides an overview of the process flow of the FTP daemon and FTP servers. Depending on an installation's requirements, there are two different ways of controlling the job name of the process forked by the FTP daemon on behalf of a logged in FTP client.

Although it is not strictly a security issue, notice that in Figure 79, after starting a cataloged procedure FTPD, the FTP daemon issues a fork() to create a new address space called FTPD1. When FTPD1 accepts an incoming connection, another fork() is executed to create an address space for handling this new user's session. This is the part that may be of interest to a system administrator: the newly forked process will take the next available (repeats are allowed) job name using the FTPDn pattern, where n is 1-9. Once a user has entered a user ID and password and assuming the RACROUTE verification is successful, the job name is changed to that of the user ID. Note that this has not always been the case -- the switch to the incoming user's ID was implemented in APARs PQ14425 and PQ27473.

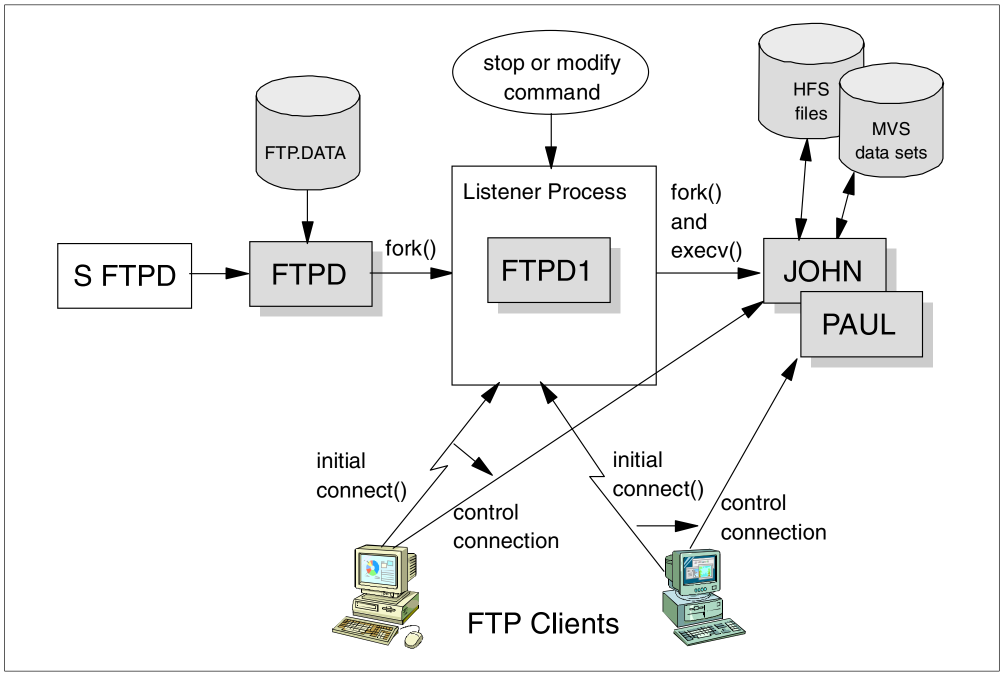


Figure 79. Process flow of OS/390 FTP server

There are some WLM implications to this change in the job name. These implications are not part of the scope of this text. Regardless, some installations may prefer to use the ENVAR parameter _BPX_JOBNAME in the FTPD cataloged procedure in order to simplify WLM configuration:

```
//FTPD PROC MODULE='FTPD',PARMS=''
//FTPD EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//        PARM=('POSIX(ON)ALL31(ON)',
//        'ENVAR("_BPX_JOBNAME=FTPD"',
//        '"TZ=EST")/&PARMS')
```

Using _BPX_JOBNAME=FTPD as shown above significantly changes what a system administrator will see for the job name of a logged in user. In Figure 79, we saw that after entering a user ID and password, the job name became that of the user ID. Using the procedure above, instead of seeing JOHN and PAUL after an FTP user has logged on, the job name associate with all FTP user address spaces would be FTPD. Again, FTPD has not always supported the _BPX_JOBNAME environment variable. This capability was added via APAR PQ32536.

From a security perspective, this means that an FTP user cannot be identified by the job name.

5.4.3 FTP and RACF

5.4.3.1 The server

The FTP server requires a user ID with UID(0). For example, the following RACF command could be used to define the userid FTPD:

```
ADDUSER FTPD OMVS(UID(0) HOME('/') PROGRAM('/bin/sh'))
```

In order to ensure the started task remains superuser, the following started task table entry would be required:

```
DC  cl8'ftpd'  procedure name
DC  cl8'ftpd'  procedure name
DC  cl8'ftpd'  userid
DC  cl8'    '
DC  xl1'40'    trusted user
DC  xl7'00'    reserved
```

A word needs to be said about BPX.DAEMON access. Ideally, the fewer user IDs with DAEMON authority, the more secure your system. In reality, however, it would be exceptionally difficult for a hacker to make user of DAEMON authority and FTPD (primarily because the FTPD executable code all resides in MVS libraries). Testing has determined that it is not required to permit the FTPD task to BPX.DAEMON. However, if ANONYMOUS support is used without a password (see 5.4.6, “FTP’s anonymous levels” on page 151) then the anonymous user ID needs to be permitted to the SURROGAT class.

5.4.3.2 Users

During a logon to the FTP server, the user is prompted for a RACF user ID and password, which are verified via the SAF interface in OS/390. A user must have a user ID on the OS/390 system to be able to access the FTP server. However, this user ID may not even be known to an end user -- see 5.4.6, “FTP’s anonymous levels” on page 151 for more details.

When the FTP server processes a logon request from a client user, it will use OS/390 system services to verify the password for the user ID and, optionally, set a new password. In addition, the FTP server job name will be saved as the application name and the IP address of the client will be saved as a terminal ID..

The terminal ID is saved as an 8-byte character string. Each octet is converted to hexadecimal. For example, IP address 9.24.104.79 in hexadecimal is X’09.18.68.4F’. Removing the octet notation and converting to a character string, the resulting terminal ID is C’0918684F’. If RACF SETROPTS has been used to activate TERMINAL(READ), all terminals (FTP client IP addresses) will be able to access FTP. The same is true if the TERMIAL class has not been activated at all.

If TERMINAL(NONE) has been used, all terminals (client FTP addresses or networks) must be explicitly permitted. To permit all users in the 9.24.104 subnet (from our example above), the following command would be appropriate:

```
RDEFINE TERMINAL 091868* UACC(READ)
```

All requests for access to data sets are checked via the SAF interface based on the user ID entered at logon to the FTP server. When FTP opens a data set on behalf of the network user, MVS OPEN uses the credentials of the network user.

Note

Please note that all users of the OS/390 FTP server must have a valid OMVS segment in their RACF user profiles. This is true even when the user only wants to access MVS data sets via the OS/390 FTP server. If desired, a default OMVS segment can be defined which will be applied to all user IDs that do not have one explicitly defined. See 3.3.8, “Default user” on page 72 for details.

If required, a user is able to change a RACF password during logon to the FTP server.

5.4.4 APPL class and FTP

When an FTP user logs onto FTP, a RACROUTE VERIFY with the APPL field set to the FTP job name is issued. This allows the RACF APPL class to be used to control which user IDs will be allowed to use the FTPD application. If you have certain user IDs (for example, user IDs coming from the Internet only) that you wish to prevent from ever using FTP, this would be the method to use. Note that APARs OW41895 and PQ32537 are both required for OS/390 V2R8 and earlier.

5.4.5 FTP security user exits

In OS/390 SecureWay Communications Server IP you have the option to enable four FTP server security user exits:

- FTCHKIP

  You may use this exit to control which TCP/IP hosts are allowed to connect to the FTP server. The exit is given control when a new connection is being opened. The exit receives the IP address and the port number of the client host and may return a return code indicating whether the connection should be allowed or denied. At ITSO Raleigh we used this exit to prevent TCP/IP hosts outside our test subnets from connecting to the FTP server.

  If the connection is denied, the following message will be sent to the user:

  ```
  421 User Exit rejects open for connection.
  ```

  Of course, FTCHKIP has some overlap with both SAF TERMINAL access (above) and network access control (4.1, “Network access control” on page 91).

- FTCHKPWD

  You may use this exit to control which user IDs may log on to your FTP server. The exit is given control just after the user has entered a user ID and password. The exit receives both the user ID and the password of the user and may return a return code indicating whether the logon should be allowed or rejected.

  If logon is denied, the following message will be sent to the user:

  ```
  530 Logon attempt by 'userid' rejected by user exit.
  ```

- FTCHKCMD

In this exit you may control which FTP subcommands a user is allowed to use. For example, you may want to use this exit to disable the `DELE` (delete) command for all users, or to disable the `PUT` command for certain users. The exit receives the user ID, the command, and the argument string entered with the command.

The exit may accept or reject the command by returning a return code. At the International Technical Support Organization we used this exit to disable all delete commands and to prevent any anonymous user from changing the password for our default anonymous user ID (FTPANOM).

If the `DELE` (delete) command is denied, the following message will be sent to the user:

```
500 User Exit denies Userid 'userid' from using Command 'DELE'.
```

- FTCHKJES

  Using this exit you may control which users are allowed to submit MVS batch jobs via the FTP server. You may also implement a check to see if a user is trying to submit a job with a user ID that is not the same as the user ID that was used for logging on to the MVS FTP server. The exit receives the user ID and the buffer containing the JCL statements that are about to be submitted via an internal reader to JES. The exit is allowed to modify the passed buffer. The exit may reject the job submission or accept it.

  If the remote job submission is denied, either of the following messages will be sent to the user depending on the command issued by the user:

```
125 User Exit refuses this Job to be submitted by userid
550 JesPutGet aborted
```

  or

```
550 User Exit refuses this Job to be submitted by userid
```

5.4.6 FTP’s anonymous levels

Conventional use of an FTPD server brings up a special security aspect. It is quite common to provide FTP services for unidentified users (essentially, the general public). The de facto convention for unidentified FTP users is to use the user ID *anonymous*. The OS/390 FTP server supports three distinct levels of anonymous users. These three levels are controlled via the ANONYMOUSLEVEL keyword in the FTP.DATA configuration file.

Providing FTP anonymous capabilities is akin to leaving the front door of your home open to the general public. It behoves you to make sure you close all interior doors and passageways to the maximum degree possible. some installations might consider using NETACCESS in conjunction with the FTP anonymous user ID to limit which networks can use anonymous.

By far, the preferred method of implementing anonymous FTP is with ANONYMOUSLEVEL 3. With anonymous FTP, the more security control in place is always better. This should be the goal when making decisions about what capabilities a user should have. Anonymous level 3, when configured optimally, can provide a very high level of security.

5.4.6.1 Level 1 (the old way)

Anonymous level 1 support can be implemented in three distinct flavors. The definitions are made in the FTP server's FTP.DATA configuration data set or file. The syntax for defining anonymous access is as follows:

```
ANONYMOUS user_id/password
```

The three flavors of anonymous level 1 are defined as follows:

- No options

  When an FTP client enters a user ID of anonymous, the FTPD server uses an MVS user ID of ANONYMO. The user will be prompted for a valid password.

  If this user ID has not been defined in RACF or if the user ID has been defined but no OMVS segment created, the logon request fails.

Default anonymous user and passwords

If no user ID is supplied on an ANONYMOUS statement and the FTP daemon is not permitted to BPX.DAEMON, then the user ID ANONYMO must be defined to the SURROGAT class, and the user ID associated with the FTPD started procedure has to have the READ permission to it.

The following commands can be used to define the SURROGAT resource:

```
SETROPTS CLASSACT(SURROGAT)
RDEFINE SURROGAT BPX.SRV.ANONYMO UACC(NONE)
PERMIT BPX.SRV.ANONYMO CLA(SURROGAT) ID(ftpd_uid) ACC(READ)
SETROPTS RACLIST(SURROGAT) REFRESH
```

- User_id

  When the FTP client user enters a user ID of anonymous, the FTPD server will act as if the user had entered the user ID specified here. The user will be requested to enter the correct password for this userid. All access to resources will be made based on the specified user ID. If this user ID has not been defined in RACF or if the user ID has been defined but no OMVS segment created, the logon request fails.

- User_id/password

  When the FTP client user enters a user ID of anonymous, the FTPD server will act as if the user had entered the specified user ID, but the user will not be requested to enter a password. The FTPD server will use the specified password in its request to RACF to create a user security environment.

From the previous description it follows immediately that traditional support of anonymous users is only possible in the last flavour. Most implementations would find this third option to be the most acceptable method of supporting anonymous users.

Here is an example of a RACF user ID called GUEST for use with anonymous support:

```
ADDUSER GUEST OMVS(UID(999) HOME('/u/guest') PROGRAM('/bin/sh'))
```

To limit access to resources in the hierarchical file system, we connected this user to the same RACF group we used for public access to our MVS Web server, the EXTERNAL group. It is recommended that a unique GID be associated with the group for high-risk users such as Web clients and anonymous FTP. It is possible for system administrators to implement GID file access control throughout the entire HFS.

In this example, user GUEST has no TSO segment defined. Even if someone was able to read FTP.DATA where the password has to be stored in clear text, no TSO access is possible. If the password was compromised, another possible entry into MVS could be through rlogin or Telnet. To manage that situation, we defined a profile in the /u/guest directory that only contained an exit command. See Figure 80.

```
echo Hello $LOGNAME
echo Sorry, you do not have access authority to this system.
exit
```

Figure 80. .profile in GUEST user directory

Since `.profile` is executed when an rlogin or telnet user connects, this profile will immediately disconnect a potential hacker. There is no reason why the FTP anonymous user ID should be used for anything other than FTP access.

As further precaution, ownership of .profile was assigned to OMVSKERN. File attributes were modified to 045, which restricts GUEST to reading and executing this file. To write a shell history log, we gave .sh_history attributes of 060 so GUEST can write to it. The example is shown below:

```
TAKADA @ RA03:/u/guest>ls -nal
total 48
drwx------   2 9999     3          8192 Aug 19 14:14 .
drwxr-xr-x  20 0        2          8192 Aug 19 14:05 ..
----r--r-x   1 0        3            86 Aug 19 14:13 .profile
----rw----   1 9999     3            22 Aug 19 14:16 .sh_history
```

Similarly, you can prevent someone from connecting to OS/390 UNIX System Services by using an anonymous user ID in another way. To do so, you should specify "/bin/false" instead of "/bin/sh" as the PROGRAM environment variable in the OMVS RACF user profile segment. In this case, if someone tries to log on using this user ID, the attempt will fail because the shell program will never be executed.

5.4.6.2 Level 2 (transitional anonymous)

According to the *IBM Communications Server for OS/390 IP Configuration Guide*, SC31-8725, ANONYMOUSLEVEL 2 is considered to be for migration purposes only and it is not recommended. The reason for this is that ANONYMOUSLEVEL 2 was originally released as a PTF for OS/390 V2R8. A short time later, level 3 was provided with OS/390 V2R10 as a more complete solution to the weakness of level 1. Level 2 contains the same capabilities as level 1, with the significant addition of a `chroot` command. When STARTDIRECTORY is set to HFS, the `chroot` command will change the root of the file system o be the user’s home directory. This effectively limits the anonymous user to interacting with files within his or her home directory (a powerful security feature).

Because level 2 is not a recommended feature and level 3 provides everything level 2 does, we will not go into any further detail on level 2.

5.4.6.3 Level 3 (anonymous deluxe)

It seems unlikely that, given the capabilities of ANONYMOUSLEVEL 3, any other level of control would be considered. Level 3 gives all that level 2 provides plus a series of additional control features.

More needs to be said about the `chroot` command. The `chroot` command stands for *change root* and that is exactly what it does. When an anonymous user logs on, the root directory (normally "/") is changed to the HOME directory as assigned in the user's OMVS segment. The rest of the file system ceases to exist -- no executable files, no configuration files. All UNIX System Services commands are no longer accessible to the user. The effect of the `chroot` command is indicated by the dotted line in Figure 81.

An installation shell script to populate the anonymous user's root (home) directory is provided in /usr/lpp/tcpip/samples/ftpandir.scp. For an example of how the anonymous directory structure is built by ftpandir.scp, see also Figure 81.

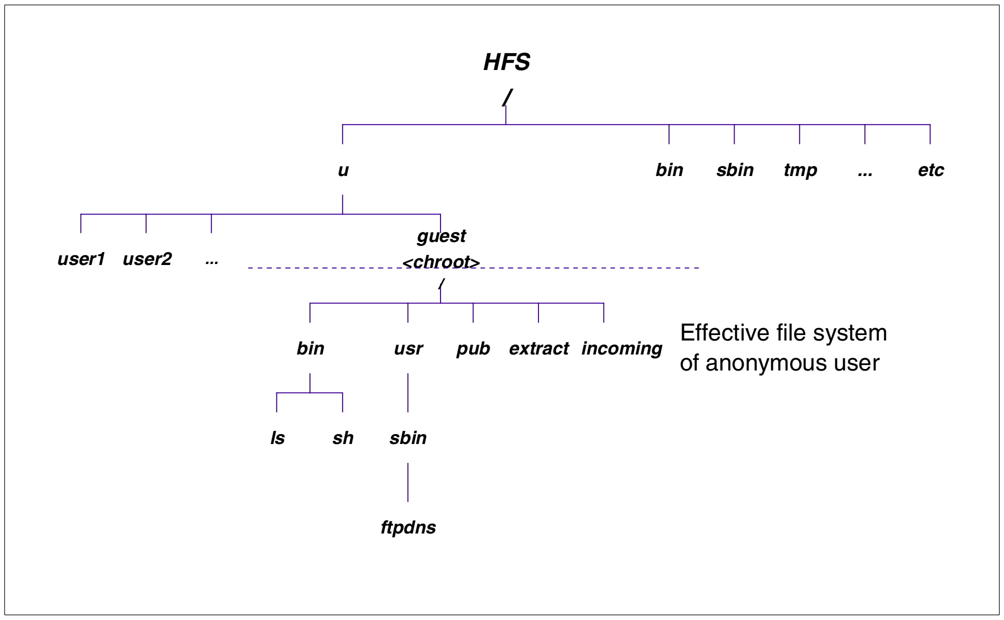


Figure 81. Anonymous directory structure

Like all your user directories, this anonymous file system should be a unique mount point. That is, it should be an HFS data set all by itself. After the HFS file system is created, mount it to the anonymous user's HOME directory and create the directory structure shown in Figure 75 using the ftpandir.scp script. In the event of a hacker trashing your anonymous file system, you can readily replace the entire mountpoint.

Anonymous directory structure

The anonymous directory structure is required for both level 2 and 3 of anonymous support. Note that this structure must be located under the HOME directory defined in the RACF user profile for the anonymous user.

When ANONYMOUSLEVEL 3 is specified, there are a several statements that become significant from a security perspective. Again, as a rule -- if an anonymous user doesn't strictly need the function, then disable it. All of the statements that begin with the prefix ANONYMOUS in the following list are statements that only apply when ANONYMOUSLEVEL 3 is specified:

STARTDIRECTORY This parameter controls the initial file system (MVS or HFS) that all FTP users are placed into at login. It is expected that you code this as HFS if you are enabling anonymous logons. There is no real need for anonymous access to the MVS file system. Such users should have their own user IDs. This parameter must not contradict ANONYMOUSEFILEACCESS.

ANONYMOUSFILEACCESS Again, anonymous users shouldn't need the MVS file system, so set this to HFS to prevent users from being able to change into the MVS file system at all.

FILETYPE This parameter should be set to SEQ in order not to contradict ANONYMOUSFILETYPESEQ.

ANONYMOUSFILETYPESEQ This parameter must be set to TRUE to allow the user to transfer files. This parameter is counter-intuitive. Once would not expect a userid confined to the HFS to need FILETYPE=SEQ. However, the only other options for FILETYPE are SQL and JES; FILETYPE=SEQ encompasses HFS files.

ANONYMOUSFILETYPEJES Code this to FALSE. Anonymous users should not be having anything to do with JES.

ANONYMOUSFILETYPESQL Again, set this to FALSE. It is unlikely that an installation would want anonymous users using SQL.

ANONYMOUSHFSFILEMODE/ANONYMOUSHFSDIRMODE The default mode settings control the access bits on files and directories created by anonymous users. Control of these setting is entirely dependent upon the expected use of information received from anonymous users. A value of 000 provides the highest security, since it renders files essentially inaccessible.

EMAILADDRCHECK This is a weak form of security, but there is no harm in forcing users to at least enter a valid format of an e-mail address. Coding FAIL for this statement will force users to enter a valid format. Even if it is an invalid e-mail, only the format is checked (no name server look-up occurs).

```
STARTDIRECTORY HFS
FILETYPE SEQ
ANONYMOUSLEVEL 3
ANONYMOUS GUEST/RINGO
ANONYMOUSFILEACCESS HFS
ANONYMOUSFILETYPESEQ TRUE
ANONYMOUSFILETYPEJES FALSE
ANONYMOUSFILETYPESQL FALSE
EMAILADDRCHECK FAIL
```

Figure 82. Example of recommended anonymous statements

Overall, using level 3 allows a system administrator to run a tight ship for anonymous FTP. Notice that we've specified a password for a user ID in plain text. If you specify a user ID only here, FTP server will ask an FTP client to enter the password for him when it connects, which is inexpedient in most anonymous FTP implementations. The only way to avoid to specifying an unencrypted password into a flat file is not configuring any user ID on the ANONYMOUS keyword, that is using the default anonymous user ID ANONYMO. Note that, in this case, FTP server has to be allowed to *read* BPX.DAEMON or act as a surrogate of ANONYMO.

Returning to our front door of the house analogy, once must remember that if any interior doors are left unlocked, they will eventually be opened.

The reader (that's you) may have noticed one disconcerting, though minor, weakness: the password for the anonymous userid must be placed into the FTP.DATA configuration file on the ANONYMOUS statement. Using RACF data set protection on this data set is always a good idea. When this file contains a userid and a password, it becomes a requirement for security reasons. Of course, since the FTP anonymous user can't access the MVS files system, this is a good argument for *not* having the FTP.DATA file in the HFS.

Table 10 provides a cross reference for the effective interactions between the ANONYMOUS and ANONYMOUSLEVEL statements. The left hand column indicates the values coded on the ANONYMOUS statement, while the uppermost column indicates the three possible ANONYMOUSLEVEL settings.

Table 10. Anonymous statement interactions

		ANONYMOUSLEVEL statement		
		1	**2**	**3**
ANONYMOUS statement	**none** (see note)	- No password check - No access control	- No password check - chroot() - Control HFS only	- e-mail address for password - chroot() - Filetype JES, SEQ, and SQL
	uid	- RACF password for the user ID - No access control	- RACF password for the user ID - chroot() - Control HFS only	- RACF password for the user ID - chroot() - Filetype JES, SEQ, and SQL
	uid/password	- No password check - No access control	- No password check - chroot() - Control HFS only	- e-mail address for password - chroot() - Filetype JES, SEQ, and SQL
Note: The FTP server has to have the READ access to BPX.DAEMON or class SURROGAT, although defining SURROGAT class would be recommended.				

In z/OS V1R2, the FTP server supports RACF surrogate for an installation-defined anonymous user name with the anonymous level set to 3, so that no clear password has to be defined in FTP.DATA.

5.4.7 FTP firewall-friendly

This parameter applies to the FTP client only. The FTP server will support any FTP clients that want to operate in firewall-friendly mode.

As mentioned before (5.4.1, "The ABCs of FTP" on page 146) an FTP session is comprised of two connections:

Control connection	active at all times, carries FTP commands and most replies.
Data connection	active during transfer of data and output from DIR/LS FTP subcommands.

As you can see in Figure 83, the data connection is normally (by default) established in the reverse direction from which the control connection is established. That is, the connection request flows from the server to the client. There are simple reasons for this: when the FTP client user wants to transfer data, the client will choose an ephemeral port, establish a listen on this port and then send a `PORT` command to the server saying `please connect to this ephemeral port`. The server receives this information and sends a connect() to the client at the specified port.

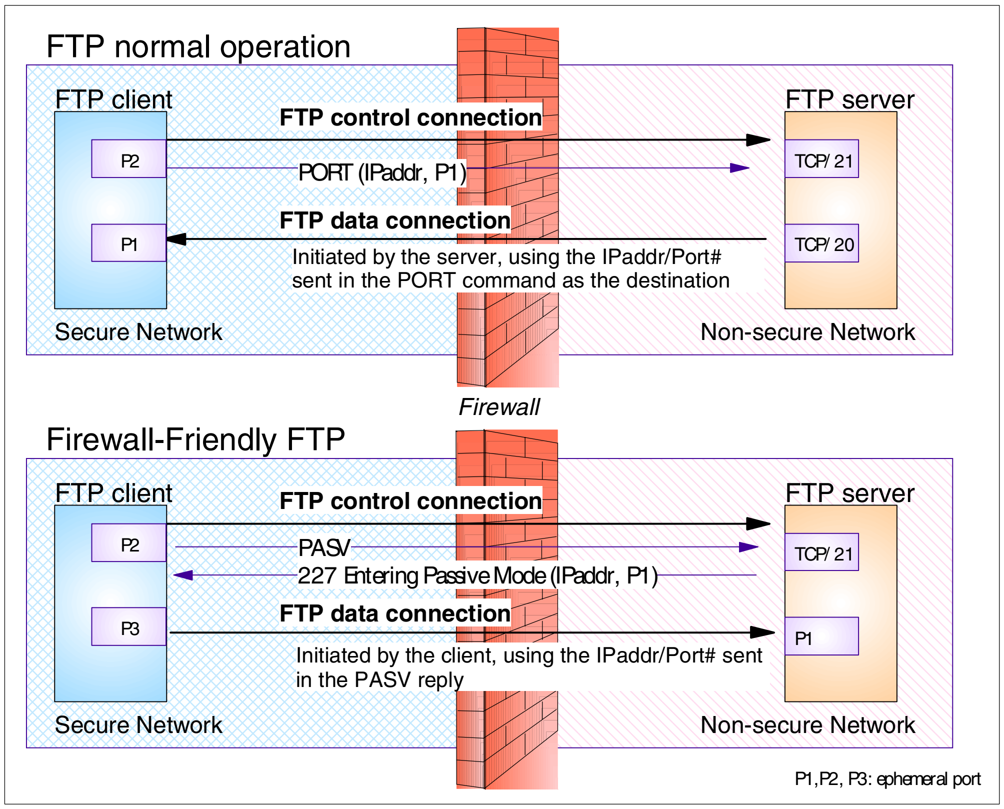


Figure 83. Firewall friendly FTP

This can present a problem: many firewall implementations will filter outbound connection requests that use a target ephemeral port (see 2.2.2.1, "Packet filtering" on page 26). Don't get confused -- we are discussing a scenario in which the FTP client is inside the secure network and the target FTP server is on the outside. In addition, many firewall implementations are reluctant to change this rule simply to allow FTP data transfers. As such, RFC 1579 was written to provide a standardized method of changing the way a data connection is established. RFC 1579 describes an FTP enhancement called firewall-friendly.

With OS/390 V2R10 and later releases (available in earlier releases via APAR PQ34332), the FTP client supports a new `locsite` parameter called `fwfriendly`. Setting fwfriendly at the local site will cause an FTP data transfer request (get, put, or dir/ls) to issue a `PASV` command instead of the `PORT` command. The `PASV` command will ask the FTP server to establish a new listen on an ephemeral port. The server will tell the FTP client what port it is listening on and the connection request will flow from client to server. Again, see Figure 83 for details.

5.4.8 FTP and JES

By setting FILETYPE=JES, an FTP user can interface with JES spool. Jobs can be submitted (`put` subcommand), retrieved (`get` subcommand) and spool listings can be reviewed (`dir` and `ls` subcommands).

WIth OS/390 V2R10, the scope of spool interaction an FTP user can perform has expanded somewhat. A new FTP.DATA parameter called JESINTERFACELEVEL can be specified.

5.4.8.1 JESINTERFACELEVEL 1

This is the more secure of the two options available (and hence, this is the one that is recommended from a security perspective). This is also the only level available in releases of OS/390 prior to OS/390 V2R10. With level 1, FTP JES users are limited to interacting with job names matching their userid plus one character.

5.4.8.2 JESINTERFACELEVEL 2

Since this level has a much wider scope of interaction possible, we'll discuss some more general spool related controls available. Although they aren't as critical for JESINTERFACELEVEL 1, all of these options could still be used in a level 1 implementation.

There are three different ways to control JES usage in an FTP environment. Unfortunately, there is no simple switch to shut down JES usage altogether. Here are the three controls possible:

ANONYMOUSFILETYPEJES	As discussed in the previous section, this parameter only applies to ANONYMOUSLEVEL 3 usage. Hence, it only controls anonymous logon users.
FTCHKJES	Discussed in 5.4.5, "FTP security user exits" on page 150. This exit can only be used to control job submission. This is no small concern -- a potential hacker that is allowed to submit jobs is a serious security problem. Luckily, this isn't the only way to control JES job submission.
SDSF-type job control	This is the good news. The FTP JES interface assumes and requires that you have configured your SDSF and JES environment according to your implementation's security requirements.

From a security perspective, the following is true: if it isn't possible via SDSF, then it won't be possible via FTP. This is because the FTP JES interface uses RACF resources the same way SDSF does. Consequently, in order to delete a job, ALTER access to JESSPOOL resource <nodeid>.<userid>.<jobname>.<Dsid>.<dsname> is required. Only READ access to the resource is required to retrieve spooled output.

For listing of jobs, the SDSF resource ISFCMD.DSP.*.jesx is used.

In summary, secure configuration of the SDSF interface is the correct method of ensuring secure access to the FTP JES interface. For more information, see your local JES expert; alternatively, see the *OS/390 Security Server for OS/390 RACF Security Administrator's Guide*, SC28-1915.

5.4.9 FTP transport affinity

The FTP server issues a bind to INADDR_ANY. As such, it is a generic server and the security implications of 4.4, "Transport affinity" on page 97 apply. Please refer to this section for more information.

5.4.10 Auditing FTP usage

Auditing is an important component of securing any TCP/IP based application. Because FTP (both client and server) on CS for OS/390 IP is able to use SMF records, auditing is reliable and thorough. An implementation needs to add only one statement to the FTP.DATA configuration data set: SMF STD. This will enable SMF records to be cut for the following subcommands:

- APPEND
- DELETE
- RENAME
- RETRIEVE
- STORE
- STORE UNIQUE
- Login failure

The SMF records contain the expected information, such as user ID, remote IP address and command name. In addition, a considerable amount of session information is provided. It should be noted, however, that the intended function of SMF records is for accounting information.

A limitation of using SMF recording for an audit trail is that it is not very effective from a real-time perspective. In order to view SMF records, the SMF dump program must be run to direct the desired SMF records to a file for viewing.

There are also specific SMF records for JES and SQL: SMFJES and SMFSQL respectively.

Another possible audit method for determining FTP daemon activity is the FTP trace option. Be warned, the FTP server trace creates a tremendous amount of output. You must have syslogd running and correctly configured. See 5.12.1, "syslogd isolation" on page 234 for security issues with syslogd. If syslogd is running, FTP daemon trace messages will be sent to the system console (MVS system log). However, it will provide an accurate record of all session activity. FTP tracing can be enabled on the fly using the MVS console `MODIFY` command:

```
MODIFY ftpprocname,TRACE
```

Tracing can also be narrowed down to a specific user ID. It can be turned off by substituting `notrace` for the `trace` keyword. There are two things to be aware of:

1. The FTP daemon trace is really an internal diagnostic tool -- some of the records require a knowledge of FTP internals in order to understand them.
2. When using the `MODIFY` command to start tracing, it will not affect existing connections. It will apply to newly established connections only.

5.4.11 FTP Kerberos support

In z/OS V1R2, both an z/OS FTP server and client support Kerberos Version 5 authentication technologies. This support is implemented using the GSS-API mechanism. Both the server and client are able to authorize each other using the third party authentication server. In addition, session data is encrypted.

With Kerberos support, data transferred over an FTP control connection can be encrypted with a symmetric cryptographic key. The key is provided by the Kerberos server and shared between the server and client, preventing passwords from flowing in clear text. Encryption can be used for the FTP data connection also.

Several new keywords are required for the FTP.DATA configuration file in order to enable Kerberos support. In addition, new FTP subcommands can be used to specify the characteristics of the authentication and encryption methods.

If you are planning to support FTP clients running on various platforms rather than z/OS, the Kerberized FTP clients are required. They are available on various platforms including Linux and IBM AIX.

This implementation in z/OS V1R2 is based on *RFC 2228 - FTP Security Extensions*, where further information can be found.

5.4.12 FTP and TLS/SSL

Of course, the OS/390 FTP server is normally configured to require user IDs and passwords from all clients to authenticate a user. It is readily apparent that this is one of the most critical forms of security. Any time a password is transmitted over the network, it is important that we protect it somehow.

Unfortunately, as shown in Figure 84 on page 162, RACF user IDs and passwords passed to the OS/390 FTP daemon are normally transmitted in clear text through an IP network. In this example, you can see that the password of user ID “TAKADA” is “mitmit”. In addition, all of the data over this session will be sent as clear text. This is a significant security exposure to customers wishing to move data across nonsecure networks.

There is some good news.

In z/OS V1R2, the FTP server and client are capable of supporting TLS/SSL. However, there may be some delay before socket level encryption can be widely used, since few other clients at this time are capable of supporting this standard.

One example of an SSL capable client is the WS_FTP Pro 6.6. For details, see the following URL:

```
http://www.ipswitch.com
```

Of course, this is not a product endorsement -- since FTP SSL clients are not particularly common at the time of writing, we've included this information.

Alternatively, if you plan to use the FTP server through nonsecure networks, you can use IPSec (see 2.3.1, "IPSec" on page 30 for further information).

```
Packet Number 12
TOK: ====( 75 bytes transmitted on interface tr0 )==== 16:20:35.406828701
TOK: 802.5 packet
TOK: 802.5 MAC header:
TOK: access control field = 0, frame control field = 40
TOK: [ src = 00:04:ac:30:24:b4, dst = 40:00:52:00:50:42]
TOK: 802.2 LLC header:
TOK: dsap aa, ssap aa, ctrl 3, proto 0:0:0, type 800 (IP)
IP:  < SRC =    9.24.104.241 >  (rs600020.itso.ral.ibm.com)
IP:  < DST =     9.24.104.33 >  (mvs03c.itso.ral.ibm.com)
IP:  ip_v=4, ip_hl=20, ip_tos=16, ip_len=53, ip_id=1815, ip_off=0
IP:  ip_ttl=60, ip_sum=945a, ip_p = 6 (TCP)
TCP: <source port=32792, destination port=21(ftp) >
TCP: th_seq=2a069071, th_ack=6bd3fb8b
TCP: th_off=5, flags<PUSH | ACK>
TCP: th_win=15840, th_sum=3a2, th_urp=0
TCP: 00000000     55534552 2074616b 6164610d 0a          |USER takada..   |.

Packet Number 13
TOK: ====( 91 bytes received on interface tr0 )==== 16:20:35.415201062
TOK: 802.5 packet
TOK: 802.5 MAC header:
TOK: access control field = 10, frame control field = 40
TOK: [ src = c0:00:52:00:50:42, dst = 00:04:ac:30:24:b4]
TOK: routing control field = 02c0,  0 routing segments
TOK: 802.2 LLC header:
TOK: dsap aa, ssap aa, ctrl 3, proto 0:0:0, type 800 (IP)
IP:  < SRC =     9.24.104.33 >  (mvs03c.itso.ral.ibm.com)
IP:  < DST =    9.24.104.241 >  (rs600020.itso.ral.ibm.com)
IP:  ip_v=4, ip_hl=20, ip_tos=16, ip_len=67, ip_id=27862, ip_off=0
IP:  ip_ttl=60, ip_sum=2e8d, ip_p = 6 (TCP)
TCP: <source port=21(ftp), destination port=32792 >
TCP: th_seq=6bd3fb8b, th_ack=2a06907e
TCP: th_off=5, flags<PUSH | ACK>
TCP: th_win=65535, th_sum=ba12, th_urp=0
TCP: 00000000     33333120 53656e64 20706173 73776f72     |331 Send passwor|
TCP: 00000010     6420706c 65617365 2e0d0a               |d please...     |

Packet Number 15
TOK: ====( 75 bytes transmitted on interface tr0 )==== 16:20:37.127343785
TOK: 802.5 packet
TOK: 802.5 MAC header:
TOK: access control field = 0, frame control field = 40
TOK: [ src = 00:04:ac:30:24:b4, dst = 40:00:52:00:50:42]
TOK: 802.2 LLC header:
TOK: dsap aa, ssap aa, ctrl 3, proto 0:0:0, type 800 (IP)
IP:  < SRC =    9.24.104.241 >  (rs600020.itso.ral.ibm.com)
IP:  < DST =     9.24.104.33 >  (mvs03c.itso.ral.ibm.com)
IP:  ip_v=4, ip_hl=20, ip_tos=16, ip_len=53, ip_id=1824, ip_off=0
IP:  ip_ttl=60, ip_sum=9451, ip_p = 6 (TCP)
TCP: <source port=32792, destination port=21(ftp) >
TCP: th_seq=2a06907e, th_ack=6bd3fba6
TCP: th_off=5, flags<PUSH | ACK>
TCP: th_win=15840, th_sum=d383, th_urp=0
TCP: 00000000     50415353 206d6974 6d69740d 0a          |PASS mitmit..   |
```

Figure 84. IP packet trace for logon to the OS/390 FTP server

5.5 OS/390 TFTP server

The TFTP protocol is a standard protocol with STD number 33. Its status is elective and it is described in RFC 1350. Updates to TFTP can be found in the following RFCs: 1785, 2347, 2348, and 2349.

TFTP is an extremely simple protocol to transfer files. It is implemented on top of the User Datagram Protocol (UDP). The TFTP client initially sends a read/write request via port 69, then the server and the client determine the port that they will use for the rest of the connection. TFTP lacks most of the features of FTP; the only thing it can do is read or write a file from or to a server.

TFTP is installed in the /usr/lpp/tcpip/sbin/ directory.

To start the TFTP server from the command line, type the `tftpd` command:

```
tftpd [-l] [-p port] [-t timeout] [-r maxretries] [-c concurrency_limit] [-s
maxsegsize] [-f file] [-a archive directory [-a ...]] [directory ...]
```

Note

The TFTP server uses well-known port 69. The TFTP server has *no user authentication*. Any client that can connect to port 69 on the server has access to TFTP. If the TFTP server is started without a home directory, it allows *access to the entire HFS*. To restrict access to the HFS, start the TFTP server with a list of directories.

If you want to use the TFTP server on OS/390, you should specify an absolute path name for a directory. You may specify no more than 20 directories on the tftpd command line.

If a list of directories is specified, only those specified directories are active. That list is used as a search path for incoming requests that specify a relative path name for a file. Activating a directory activates all of its subdirectories. TFTP will not allow access to any other parts of the file system.

On the other hand, if the TFTP server is started without a list of directories, all mounted directories are considered active. In this case, all HFS files are open to every TFTP client because TFTP does not provide user authentication.

In addition, the TFTP server pre-forks a child process to handle incoming requests when the concurrency limit is exceeded. Consequently, immediately after starting the TFTP server, two TFTP processes exist.

In case of a flood of concurrent TFTP requests, the TFTP server may fork additional processes. When the number of concurrent requests being processed drops below the concurrency limit, the number of TFTP processes is decreased back to two.

TFTP may sound like a very dangerous alternative to FTP, but it is extensively used to download code and initial configurations to routers and simple workstations, because of its simplicity. Because of its lack of security, you ought to take these simple steps if you require to run a TFTP server on your OS/390:

- Ensure that the TFTP home directory list is short and harmless.
- Use IPSec whenever a nonsecure network is involved.
- Restrict access to port 69 on packet filters in firewalls.

5.6 OS/390 NFS server

The Network File System (NFS) was developed by Sun Microsystems in 1985 and has been implemented on various platforms such as Sun Solaris, HP/UX, AIX, Linux, Windows, OS/390 and so forth. NFS clients can use the resources of NFS servers transparently as if they were their own local file systems.

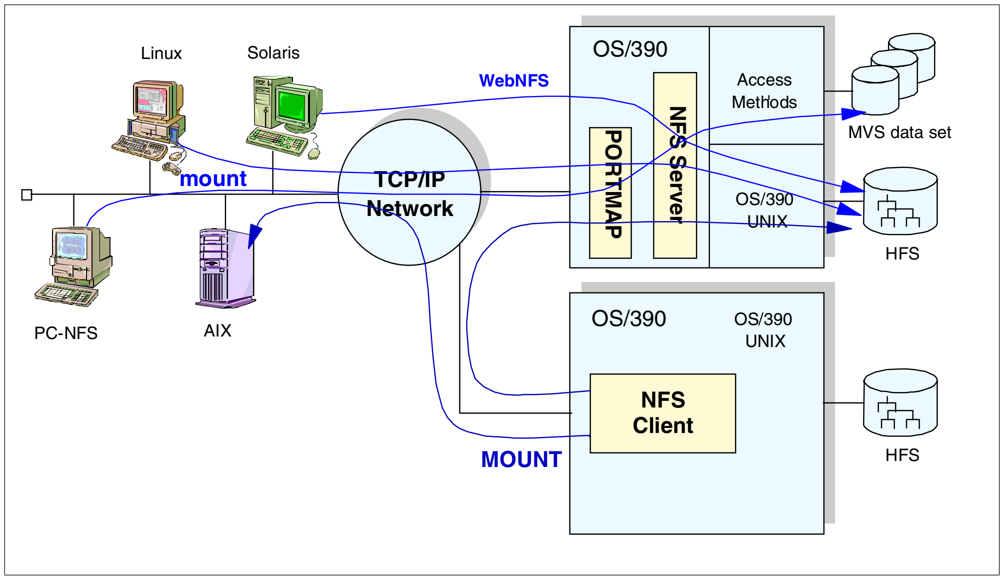


Figure 85. OS/390 NFS overview

Client systems in a TCP/IP network that support the NFS client protocol can use traditional MVS data sets and UNIX System Services HFS files as part of their file system. Currently, OS/390 UNIX System Services HFS files cannot be physically shared in read/write mode among multiple OS/390 systems. Therefore, the combination of the OS/390 NFS server with the OS/390 NFS client is one of the alternative solutions that you can use to share OS/390 UNIX System Services HFS files through a TCP/IP network.

NFS can use TCP or UDP to transport its data, and has been enhanced in recent releases of OS/390. Since the purpose of this section is to discuss OS/390 NFS security, if you want to know more about the OS/390 NFS server, see *OS/390 Network File System Customization and Operation*, SC26-7253.

5.6.1 OS/390 NFS security levels

The OS/390 NFS server provides the following four security levels. These security levels are controlled by the security parameter in the OS/390 NFS attributes data set:

- Unrestricted data access

  When you specify `security(none)` in the attributes data set, no security checking is performed. We never recommend the use of this security level.

- Exports list checking

  When you specify `security(exports)` in the attributes data set, the EXPORTS data set is used to check security. The EXPORTS data set is an MVS data set that is specified in the EXPORTS DD statement of the NFS server started procedure. The EXPORTS data set corresponds to the UNIX "/etc/exports" file.

  The EXPORTS data set can control which client users can mount which MVS data sets, based on the client machine's specified IP address. It can also control which client users can mount all or part of the HFS. The entries in the exports data set specify which MVS high-level qualifiers or HFS directories can be mounted. The system administrator can use this data set to limit mounts to accredited clients only.

  Note, however, that it is relatively easy to spoof an IP address, so exports list checking basically does not provide any real security.

- SAF checking

  When you specify `security(saf)` in the attributes data set, the System Authorization Facility (SAF) checking is performed. RACF, or an equivalent security product, provides the security information.

  The server uses SAF to validate the OS/390 user ID and password supplied by the client user. It also uses SAF to validate that the client user is allowed to access the data set on MVS. A RACF user ID must be defined for each client user that requires access to the server.

  For HFS data, OS/390 checks the UNIX permission bits before granting file access to a client user. The permission checking is based on the OS/390 UNIX System Services UID obtained by the mvslogin process, not the UID passed in by the client on the RPC request. For users accessing HFS, their RACF user ID must have an OMVS segment defined in the RACF profile.

- Both SAF and exports list checking

  When you specify `security(safexp)` in the attributes data set, NFS checks for both RACF authorization and exports list authorization before granting a client user access to OS/390 data. For HFS data, OS/390 checks the UNIX permission bits before granting file access to a client user. The UNIX permission checking is based on the OS/390 UNIX System Services UID obtained by the mvslogin process, not the UID passed in by the client on the RPC request. This is the most restrictive means of limiting DASD access. It requires client users to use the mvslogin command.

Note

If you specified `security(saf)` or `security(safexp)` in your OS/390 NFS attributes data set, you must download the `mvslogin` and `mvslogout` client commands to the NFS client system. For more information, see *OS/390 Network File System Customization and Operation*, SC26-7253.

5.6.2 Security information exchange between NFS client and server

This section describes how an IP host name and a RACF user ID are provided to MVS NFS.

1. How is the client IP host name resolved by the NFS server?

   If an NFS client connects to the NFS server, only the IP address is transmitted over the network. On the NFS server side, the client IP address is resolved into a client host name by using either local hosts tables or a query to a local or remote name server. It should be noted that in most implementations it is fairly easy to use a fake IP address on the client side. As mentioned previously, the EXPORTS data set simply lists the host name permitted to mount some directories of the host on which the NFS server is running. If a malicious user can get at IP address information listed in the EXPORTS data set by using the `showmount` command (see Figure 86) and tries to mount using this IP address illegally, unfortunately the NFS server permits this operation. Therefore, in the OS/390 environment, it is better to use both SAF and EXPORTS list checking.

```
takada@rs600020[/home/takada]showmount -e mvs03a      1
export list for mvs03a:
/HFS/u/takada         rs600020                        2
/HFS/etc              (everyone)                      3
/TCPIP.TCPPARMS.R2505 mvs03c                          4
```

Figure 86. Display exported directory Information on OS/390

   1 This command shows a list of the directories on OS/390 host mvs03a that some NFS clients can mount.

   2 This means that host rs600020 can mount the HFS directory /u/takada.

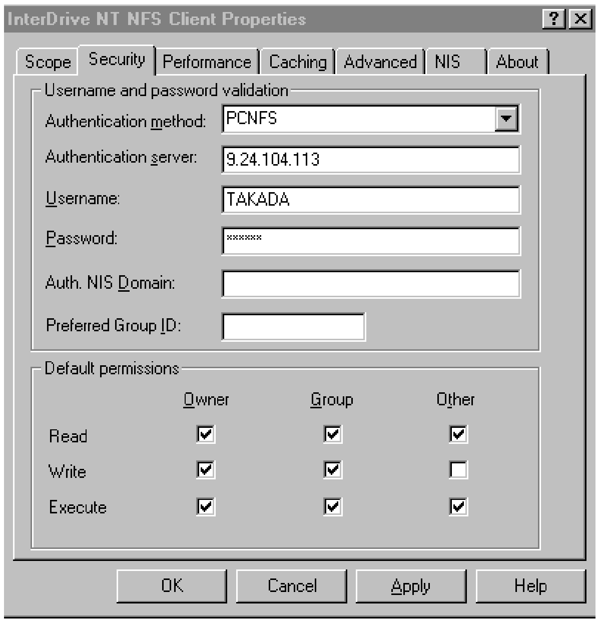
   3 This means that every host can mount the HFS directory /etc.

   4 TCPIP.TCPPARMS.R2505 is a partitioned MVS data set. The host mvs03c can mount this data set as if this was a directory. PDS members are regarded as files.

2. How is a RACF user ID provided to NFS?

- For single user PC systems, this is usually a simple task. Most PC systems support an extension to the NFS protocol called PC-NFS client. The OS/390 NFS server has, like many NFS implementations, a PC-NFS server. PC-NFS support is enabled by specifying the pcnfsd verb in the NFS attributes data set. If PC-NFS support is enabled on the NFS server, after a `mount` command is received from the NFS client, the server prompts the client asking for a user ID and password. Both are checked by RACF.

  In our test environment, we used the InterDrive NT NFS Client provided by FTP Software. Figure 87 on page 167 shows how to specify the security information. In this case, you have to specify OS/390's host name or IP address in the Authentication server field and then fill in the RACF user ID and password in the Username and Password fields.



InterDrive NT NFS Client Properties

Scope | Security | Performance | Caching | Advanced | NIS | About

Username and password validation

Authentication method: PCNFS

Authentication server: 9.24.104.113

Username: TAKADA

Password: ******

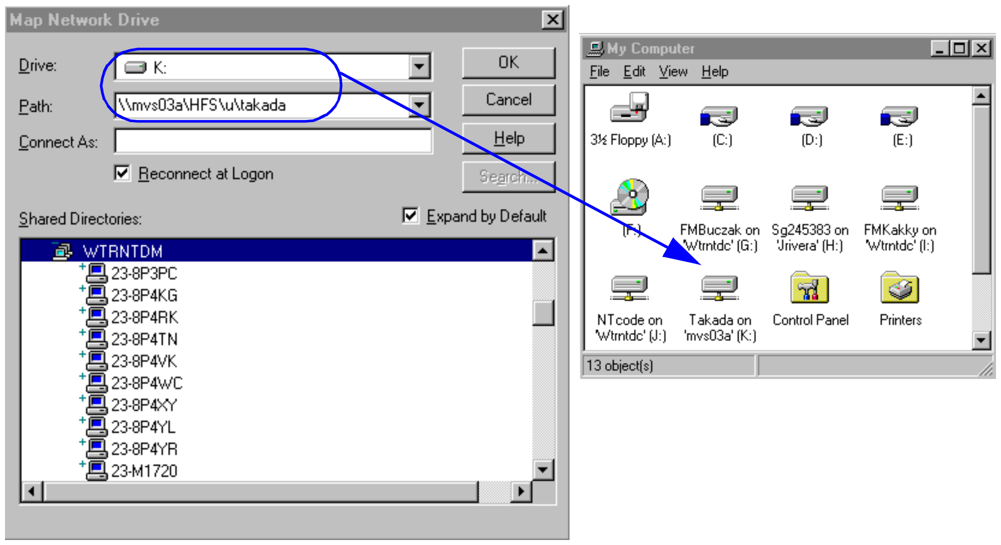
Auth. NIS Domain:

Preferred Group ID:

Default permissions

	Owner	Group	Other
Read	✓	✓	✓
Write	✓	✓	
Execute	✓	✓	✓

Figure 87. Defining RACF user ID and password in PC-NFS client configuration



Map Network Drive

Drive: K:

Path: \\mvs03a\HFS\u\takada

Connect As:

✓ Reconnect at Logon

Shared Directories: ✓ Expand by Default

WTRNTDM
23-8P3PC
23-8P4KG
23-8P4RK
23-8P4TN
23-8P4VK
23-8P4WC
23-8P4XY
23-8P4YL
23-8P4YR
23-M1720

Figure 88. Mount processing on Windows NT client

When you want to mount OS/390 resources (MVS data sets, HFS or both) in your Windows NT environment, you have to map the OS/390 resources to logical drives. First, right-click the **Network Neighborhood** icon on the desktop and then select **Properties**. After the Map Network Drive window appears (Figure 88), fill in the OS/390 resources in the Path field and click **OK**. Meanwhile, the RACF user ID and password are sent to the OS/390 security server specified in Figure 87 and then the OS/390 security server checks if they are valid. OS/390 resources you specify in the Path field must be expressed in the following manner:

```
\\hostname\resource_name
```

In this case, `hostname` is mvs03a and `resource_name` is \HFS\u\takada. Now you can use OS/390 resources as if they were local disk drives.

- The problem arises with UNIX systems, which usually do not provide a PC-NFS client, because they are by design multiuser systems.

  In most cases on a UNIX system, users must have superuser authority (usually called root authority) to issue a `mount` command. Because standard mount processing does not provide user ID and password checking, NFS running on OS/390 has implemented a protocol extension to mount called mvsmount. The mvsmount protocol extension is implemented by the commands `mvslogin` and `MImvslogout`, which allow the entry of a RACF user ID and password on several UNIX platforms.

  To enable client users to access the MVS system, you must install the `mvslogin` and `mvslogout` commands on the client workstations. The source code to install these commands is provided in the hlq.NFSTARB data set. For some client machines, you might need to modify the code to port these commands so they run on your client machine.

  In our environment, we ported these clients to the AIX V4.3.2 operating system. Figure 88 shows how to use the `mvslogin` and `mvslogout` commands on the AIX operating system. In this case, we specified `security(safexp)` in the NFS attribute data set.

```
takada@rs600020[/home/takada]showmount -e mvs03a                                    1
export list for mvs03a:
/HFS/u/takada          rs600020
/HFS/etc               (everyone)
/TCPIP.TCPPARMS.R2505 mvs03c
takada@rs600020[/home/takada]su                                                     2
root's Password:
takada@rs600020[/home/takada]mount mvs03a:/HFS/u/takada hfsdir                      3
takada@rs600020[/home/takada]mount mvs03a:/HFS/etc etcdir
takada@rs600020[/home/takada]mount mvs03a:/TCPIP.TCPPARMS.R2505 mvsdir
mount: access denied for mvs03a:/TCPIP.TCPPARMS.R2505
mount: giving up on:
        mvs03a:/TCPIP.TCPPARMS.R2505
Permission denied
takada@rs600020[/home/takada]mount                                                  4
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
         /dev/hd4         /                jfs    Aug 09 13:57 rw,log=/dev/hd8
         /dev/hd2         /usr             jfs    Aug 09 13:57 rw,log=/dev/hd8
         /dev/hd9var      /var             jfs    Aug 09 13:57 rw,log=/dev/hd8
         /dev/hd3         /tmp             jfs    Aug 09 13:57 rw,log=/dev/hd8
         /dev/hd1         /home            jfs    Aug 09 13:58 rw,log=/dev/hd8
         /dev/lvkakky     /home/kakky      jfs    Aug 09 13:58 rw,log=/dev/hd8
         /dev/lv00        /usr/inst.images jfs    Aug 09 16:40 rw,log=/dev/hd8
mvs03a   /HFS/u/takada    /home/takada/hfsdir nfs3   Aug 19 11:21

mvs03a   /HFS/etc         /home/takada/etcdir nfs3   Aug 19 11:21
takada@rs600020[/home/takada]exit
takada@rs600020[/home/takada]mvslogin mvs03a takada                                 5
Password required
GFSA973A Enter MVS password:******                                                  6
GFSA978I TAKADA logged in ok.                                                       7
           :
           :
takada@rs600020[/home/takada]mvslogout mvs03a                                       8
GFSA958I uid 5002 logged out ok.
```

Figure 89. NFS mount process using mvslogin and mvslogout commands

1 You can check which directory you can mount by means of the `showmount -e` command.

2 To issue the `mount` command, you have to gain root authority.

3 This shows three mount processes. The reason why the last one fails is that hosts other than mvs03a cannot mount TCPIP.TCPPARMS.R2505 (1).

4 This command enables you to check the status of all mounted directories in your UNIX system.

5 This shows how to issue the `mvslogin` command; mvs03a is the host name of the NFS server and TAKADA is the RACF user ID.

6 You have to enter your password as registered in the OS/390 RACF database.

7 The GFSA978I message means the authentication has succeeded. If the client UID and GID on the UNIX system do not match the UNIX UID and GID, you will see the following informational message that follows the GFSA978I message:

```
Mismatch in uid/gid: OpenEdition uid is 50011, gid is 3,
          client uid is 5002, gid is 1.
```

This informational message contains the OS/390 UNIX UID and GID for the OS/390 user identification.

8 This shows how to issue the `mvslogout` command.

Table 11 shows the OS/390 NFS processing at the time of mount requests.

Table 11. OS/390 NFS server processing of a mount request

Security Option	Export File	HFS file	MVS dataset
none	Not required	No checking, exported	No checking, exported
saf	Not required	No checking, exported	No checking, exported
exports	Required	Checking export file	Checking export file
safexp	Required	Checking export file	Checking export file

5.6.3 Access to the HFS

HFS security is based on permission bits associated with an HFS file, UID and GID values associated with the file, and the requesting RACF user ID.

A UID associated with a user is a number specified in the OMVS segment of a RACF user ID. A GID associated with a user is a number specified in the OMVS segment of the default RACF group to which the user belongs.

Permission bits specify whether read, write, or execute permission is granted to the file owner, the group to which the file owner belongs, or to everyone. When a file is created, it is automatically associated with the UID of the user that creates the file (the file owner) and the GID of the directory it is in (the parent directory).

If a UNIX System Services user tries to access an HFS file, the UID and GID are compared with the UID and GID associated with the file. Depending on whether the values are equal, UNIX System Services grants the access rights of the file owner, the owner's group, or the rights that are granted to everyone.

If NFS is used to access HFS files, we must take into account which UIDs and GIDs are in effect on the NFS client system, and which security scheme is used with the OS/390 NFS server.

Because the UID and GID are associated with an HFS file (and not related to a user name or group name), the `ls -l` command on an NFS client system will return different file-owning user names and file-owning group names to what is on the NFS server system, if the assignment of UIDs to user names and GIDs to group names is not consistent within the NFS network.

Also we have to consider which UID and GID is assigned to the NFS client users when they access an HFS file. This is dependent on the NFS security scheme that is in use:

- If EXPORTS security (`security(exports)`) is used, there is no identification to RACF. For this reason, the UID and GID values associated with the user ID acting on the NFS client system are used for UNIX System Services security checking. An exception to this rule is UID=0 (superuser). It is mapped to 65534. This prevents malicious users from accessing mounted OS/390 resources with superuser authority.

- If SAF security (`security(saf`) or `security(safexp)`) is used, the NFS client user has to identify itself to RACF by either a PC-NFS mount or the `mvslogin` command. RACF associates UID and GID values with the NFS client user. These values are used in further processing on the OS/30 NFS side. The problem is that additional security checking is done on the NFS client side. This complicates matters when the UID/GID values are not consistent in your network.

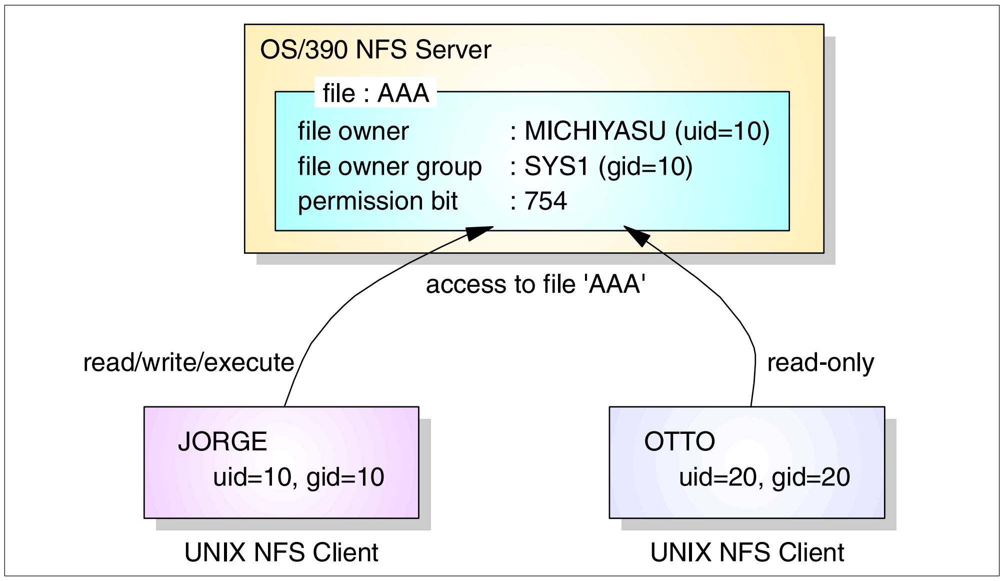


Figure 90. NFS connection between a UNIX NFS client and an MVS NFS server

Figure 90 shows an example of an NFS connection between a UNIX NFS client and an OS/390 NFS server using EXPORTS security.

MICHIYASU is a user with UID=10, GID=10 defined under UNIX System Services and this user is an owner of the HFS file AAA. HFS file AAA has permission bits 754 which mean:

- The file owner can access this file with read/write/execute authority.
- Users who belong to the file owner group have read/execute authority.
- Other users can only read this file.

Now there are two other NFS client user IDs accessing the HFS file. Let us call them JORGE with UID=10, GID=10, and OTTO with UID=20, GID=20.

Note

Independent of the security scheme used, a user with superuser authority (UID=0) can issue NFS `mount` commands on UNIX systems.

JORGE will get access to MICHIYASU’s files, that is, file AAA, as if it were MICHIYASU because they have the same UID and GID values (10). On the other hand, OTTO, which has a UID=20 and GID=20, gets world access, that is, read-only access.

If `security(safexp)` is specified in the OS/390 NFS attribute data set, OTTO issues the `mvslogin` command and passes MICHIYASU's RACF user ID and password to the OS/390 security server as shown in Figure 89 on page 169. OTTO can access the file AAA with READ/WRITE/EXECUTE authority.

Table 12 lists how the OS/390 NFS server handles a file request from an NFS client in each security option:

Table 12. OS/390 NFS server processing of a file request

Security Option	MVSLOGIN	HFS file	MVS dataset
none	Not required	Check file permission bits	No checking
saf	Required	SAF check	SAF check
exports	Not required	Check file permission bits	No checking
safexp	Required	SAF check	SAF check

Note

MVS conventional data sets do not support UNIX permission bits in the file attribute structure. By disabling the SAF security, you lose authorization checking for file operation to MVS conventional data sets. Therefore, if you need to give NFS clients access to MVS data sets, you should enable SAF security and protect your MVS data sets by making a RACF MVS data set profile.

5.6.4 Conclusion

The following are our recommendations for using NFS with UNIX System Services:

1. Regardless of the security scheme you use, assign consistent UID and GID values in your NFS network. Each user should have the same UID and GID on every system he or she works on.
2. Enable the PC-NFS support in OS/390 NFS and use PC-NFS where possible.
3. Use EXPORTS security when you can trust your UNIX system administration.
4. Use SAF security when your environment has additional security requirements.

   With SAF security:

   - Follow this sequence of commands when:

   a. Mounting to OS/390 NFS:

      1. Log in to the user ID root on the NFS client (if using a UNIX workstation).
      2. Issue the `mvslogin` command. This is not required to mount, but it is useful to check whether everything is in order.

3. Issue the `mount` command.
4. Check access to the HFS directory by using the `df` command on UNIX or the `qmount` command on DOS and OS/2.

b. Using HFS:

1. Log in to the user ID you want to work with in your UNIX environment.
2. Issue the `mvslogin` command.
3. You should now be able to access the mounted file system as permitted.
4. If you logged out from UNIX, issue the `mvslogin` command after the next UNIX login.

c. Unmounting from OS/390 NFS:

1. Log in to the user ID root on the NFS client (if using a UNIX workstation).
2. Be sure that no other user needs the mounted directory.
3. Issue the `umount` command.
4. Issue the `mvslogout` command.

- Tell the UNIX users which directories are mounted from OS/390 NFS, and that they may have different access rights for HFS files than for local files if the UID and GID values do not match.
- Because the EXPORTS file is not used, the NFS client's `showmount` command (`showexp` under OS/2) will receive the reply:

```
no exported file systems
```

- Be aware of the fact that the user ID used for mvslogin could also be used for other services like rlogin, ftp, and Telnet.

5. SAFEXP security combines EXPORT and SAF security, making it the most secure NFS security level. However, because of its complexity (checking UID and GID values, along with user ID and passwords), this level of security may also cause additional confusion.

   - Use SAFEXP security only if you want to hide parts of the HFS from the outside world.
   - Be aware that faking the IP host address is not a difficult task, especially on PC systems in an office environment.
   - Keep the EXPORTS file as simple as possible. You have more flexibility if you assign access rights to HFS files by using RACF user IDs with different UID and GID values for security checking.
   - With SAFEXP security, the `showmount` command will give a response, but this just reflects the EXPORTS file.
   - All other security exposures mentioned for SAF security apply also to SAFEXP security.

5.7 UNIX System Services rlogind/rshd/rexecd

OS/390 UNIX System Services provides safeguards to eliminate many of the typical exposures associated with TCP/IP applications. IBM has eliminated the use of one of the most dangerous features of traditional UNIX internetworking, the use of trusted host facilities.

In many UNIX systems, the system administrator may create a file, /etc/hosts.equiv, of trusted systems. Alternatively, each user may create a file, .rhosts, in the user's home directory, which lists hosts that the particular user trusts. In either case, the trusted host designation means that user IDs on other hosts may login using rlogin or issue remote shell commands without specifying a password. While this feature may be convenient in some cases, often mistakes are made, which leave systems and servers open to penetration.

For example, a system administrator on sun.tsc.ibm.com creates a /etc/hosts.equiv file shown in the following:

```
aix.tsc.ibm.com  mike
```

This means that a user mike on the host aix.tsc.ibm.com can access sun.tsc.ibm.com without a password. If both hosts trust each other, this file is very useful. However, if a malicious user modifies this file to the following, this host is open to the public and every user can access it without a password:

```
+
```

If users with a high level of authority access this system, they can modify, delete and steal critical information. This is very dangerous regardless of the location of the host.

IBM has eliminated this potential problem for UNIX System Services. A password is *always* required to use rlogin/rsh/rexec on UNIX System Services. The only exception to this is when a client tries to use `rsh` without a password. In this case a user exit (ruserok) is invoked, which you must code to ensure a suitable level of security (for example, by demanding a password).

Below are examples of accessing rlogind and rexecd on UNIX from a client on AIX. In both cases, you have to enter your RACF user ID:

```
takada@rs600020[/home/takada]rlogin mvs03c -l takada
**********************************************************************
*                                                                    *
*  Welcome to SecureWay Communication Servers for OS/390 V2R8        *
*      on ITSO Raleigh RA03                                          *
*                                                                    *
*  This is the /etc/banner page for telnet server use.               *
*                                                                    *
**********************************************************************

takada's Password:
IBM
Licensed Material - Property of IBM
5647-A01 (C) Copyright IBM Corp. 1993, 1999
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

 - - - - - - - - - - - - - - - - - - - - - - - - - - -
 - Improve performance by preventing the propagation -
 - of TSO/E or ISPF STEPLIBs                         -
 - - - - - - - - - - - - - - - - - - - - - - - - - - -
TAKADA @ RA03:/u/takada>
```

Figure 91. Example of using UNIX rlogin

```
takada@rs600020[/home/takada]rexec mvs03a onetstat -p TCPIPA -h
Name (mvs03a.itso.ral.ibm.com:takada): takada
Password (mvs03a.itso.ral.ibm.com:takada):
MVS TCP/IP onetstat CS V2R8       TCPIP Name: TCPIPA           14:00:11
Home address list:
Address          Link             Flg
-------          ----             ---
172.16.250.3     VIPA3A           P
9.24.104.113     TR1
172.16.100.3     M032216B
172.16.103.3     M032216C
172.16.220.50    LOOPBACK
172.16.220.51    LOOPBACK
172.16.220.52    LOOPBACK
172.16.250.254   IUTSAMEH
172.16.240.3     VIPLAC10F003
127.0.0.1        LOOPBACK
```

Figure 92. Example of using UNIX rexec

When the -r option is enabled, if there is no password specified on the `rsh` command from the client, rshd will drive the installation exit. When the installation exit is driven, rshd looks for a program in /usr/sbin named ruserok.

This is the only name that it will look for. If /usr/sbin/ruserok is not found, the request will fail.

When the rshd server invokes /user/sbin/ruserok, it will pass parameters in the following order:

1. Host name
2. Local user's UID
3. Remote user ID
4. Local user ID

If rshd receives a return code of zero from the installation exit, rshd continues. Any nonzero return code from the installation exit will cause rshd to issue the message EZYRS25E to the client and terminate all connections. The following code fragment can be used as an example to begin building a working ruserok installation exit:

```
int main(argc, argv)
int argc;
char *argv[];
char *rhost1; /* "hostname" or "hostname.domain" of client
obtained by caller:
gethostbyaddr(getpeername()) */
int cliuid; /* uid of the user name on client©s system */
char *cliuname; /* user name on client©s system */
char *servuname; /* user name on this (server©s) system */
int rc = 4;
rhost1 = argv[1];
cliuid = atoi(argv[2]);
cliuname = argv[3];
servuname = argv[4];
.
<authenticate user and set rc=0 if valid>
.
return(rc);
```

As discussed in this section, the OS/390 rlogin/rsh/rexec daemon requires users to enter their RACF user ID and password. Because of this, the daemon can be quite secure compared to other UNIX systems, on which users can log in to these daemons without passwords. However, similar to OS/390 Telnet or FTPD, user IDs and passwords that users enter are transmitted in clear text. Therefore, if you plan to use these daemons through nonsecure networks such as the Internet, we recommend that you encrypt your data using IPSec.

5.7.1 OS/390 UNIX rshd Kerberos support

In z/OS V1R2, the OS/390 UNIX rshd server can support the Kerberos Version 5 authentication technologies.

Either Kerberos Version 5 authentication mechanism or that of GSS-API may be used to authenticate clients. If the username in the Kerberos or GSS-API credentials matches the local user ID (local_userID) of the client supplied by the RSH client, then no password is required.

For more information about the Kerberos Version 5, see 2.7, "Kerberos-based security system" on page 46.

5.8 OS/390 SNMP

Simple Network Management Protocol (SNMP) defines an architecture that consists of:

- Network management applications
- Network management agents and subagents
- Network elements or objects

The SNMP network management application can ask agents for specific information about network elements. Conversely, agents can notify a network management application when something happens to one or more network elements. The protocol used between the network management application and agents is SNMP.

The transport protocol for SNMP requests is the User Datagram Protocol (UDP).

SNMP defines both the network management data and the ways in which the data is retrieved or changed by the network management application. Examples of network management data include device definitions, counts of packets received at the IP layer and TCP connection data. The information about the network elements is stored in a Management Information Base (MIB), which is supported by the SNMP agent and its subagents. A MIB variable (or MIB object) is a specific instance of data in a collection of objects related to a common management area.

Figure 93 illustrates the relationships between manager, agents and subagents.

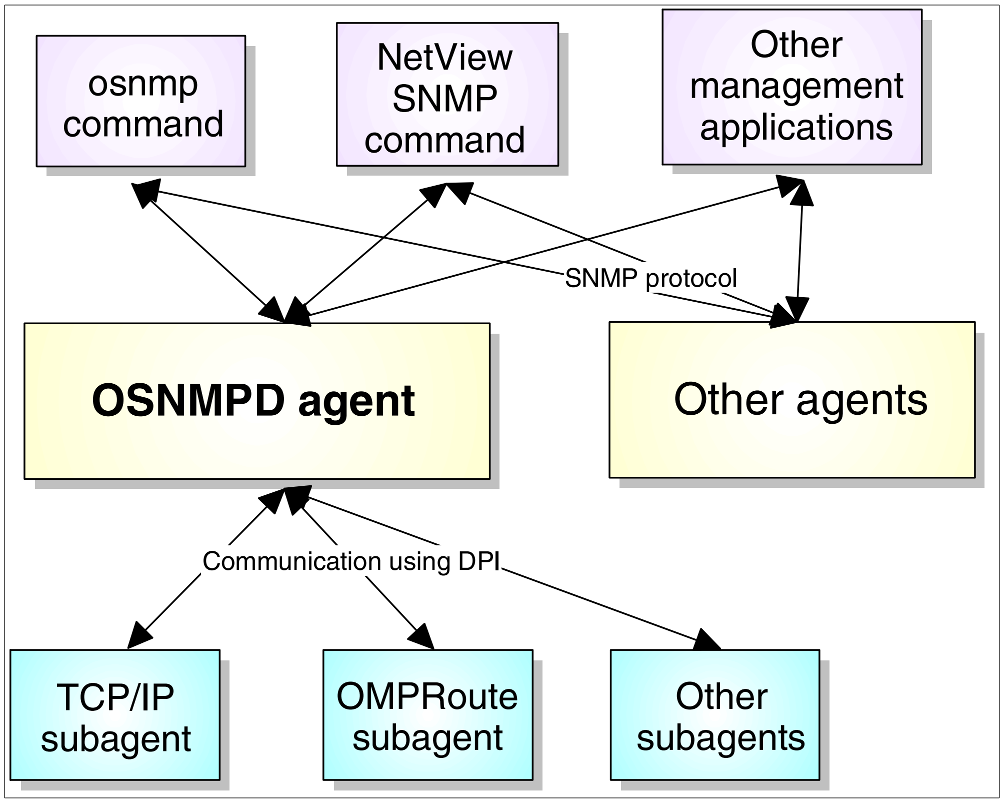


Figure 93. Functional components of SNMP

The following list illustrates the sequence of events from the time you issue an SNMP command until you receive the response:

1. The user issues a NetView SNMP (`SNMP`) or OS/390 SNMP (`osnmp`) command.
2. The command processor validates and encodes the request in a protocol data unit (PDU), and sends it to the SNMP agent.
3. The SNMP agent validates the request and, if necessary, sends it to an SNMP subagent (for example, the OMPRoute subagent, see Figure 93). Requests for agent-oriented objects are handled by the agent and all others are handled by a subagent. To determine which objects are handled by the agent and which by a subagent, refer to *OS/390 SecureWay Communications Server IP User's Guide,* SC31-8514.
4. The agent sends the response to the originator of the request. The command processor displays the response.

5.8.1 SNMP security

Perhaps you are wondering, why security at all? All a hacker can do is read a bunch of variables and maybe change a few minor ones. Ah, but if your hacker reads through some of the MIB variables, he or she might find some very interesting read/write objects. The MIB object ipForwarding, for example, is a read/write variable. If ipForwarding were to be changed from 1 (routing) to 2 (not routing), what would be the consequences to your network? In addition, the potential hacker could change or generate trap or inform values, causing your SNMP manager to make an incorrect decision. Like the rest of the TCP/IP world, SNMP applications want to be able to know who the data came from and whether the data may have been modified (if only we could have such assurance in our personal lives).

The SNMP agent supports what amounts to two types of security:

- SNMPv1 and SNMPv2c
- SNMPv3 security

SNMPv1 and SNMPv2c are community-based security, where a community name (or password) is passed with a request. If the community name is recognized as one that can be used by the IP address from which the request originates, the SNMP agent processes the request. SNMPv1 and SNMPv2c have apparent limitations: no encryption is used, no user IDs are required. Without user IDs, no limit can be placed on the scope of MIB objects that can be queried or set.

Note

The experimental SNMPv2u security is no longer supported. It has been replaced by the standards-based SNMPv3 network management framework. SNMPv3 was introduced in OS/390 V2R7.

SNMPv3 provides a more powerful and flexible framework for message security and access control. Message security involves providing:

- Data integrity checking, to ensure that the data was not altered in transit
- Data origin verification, to ensure that the request or response originates from the source from which it claims to have come

- Message timeliness checking and, optionally, data confidentiality, to protect against eavesdropping. Access control is the ability to control exactly what data an individual user can read or write.

The SNMPv3 architecture introduces the User-Based Security Model (USM) for message security and the View-Based Access Control Model (VACM) for access control. The architecture supports the concurrent use of different security, access control, and message processing models. For example, community-based security can be used concurrently with USM (they do not interact with each other -- they can simply coexist without interference).

5.8.1.1 USM

USM uses the concept of a user for which security parameters (levels of security, authentication and privacy protocols, and keys) are configured at both the agent and the manager. Messages sent using USM are better protected than messages sent with community-based security, where passwords are sent in the clear and can be displayed in traces. With USM, messages exchanged between the manager and the agent have data integrity checking and data origin authentication. Message delays and message replays (beyond what happens normally due to a connectionless transport protocol) are protected against with the use of time indicators and request IDs.

USM supports two authentication algorithms HMAC-SHA and HMAC-MD5. The generation of the authentication keys involves some manual intervention on behalf of the system administrator. Things are simplified somewhat by the `pwtokey` utility. This utility allows a password to be used to generate a key.

For data confidentiality, symmetric encryption algorithms are used. No asymmetric encryption is available. CBC 56-bit DES is the only encryption available. Again, generation of the symmetric key can be done using the `pwtokey` facility with a password as input.

Note that the password used in generation of these keys has no relationship with the community name (password) used for pre-SNMPv3 protocols. The password is simply a method of simplifying the generation of authentication and privacy keys.

For SNMP agents and managers, these keys must be stored into configuration files (snmpd.conf and osnmp.conf on the OS/390 platform, respectively). Protection of these configuration files is paramount to maintain the security of your SNMP environment.

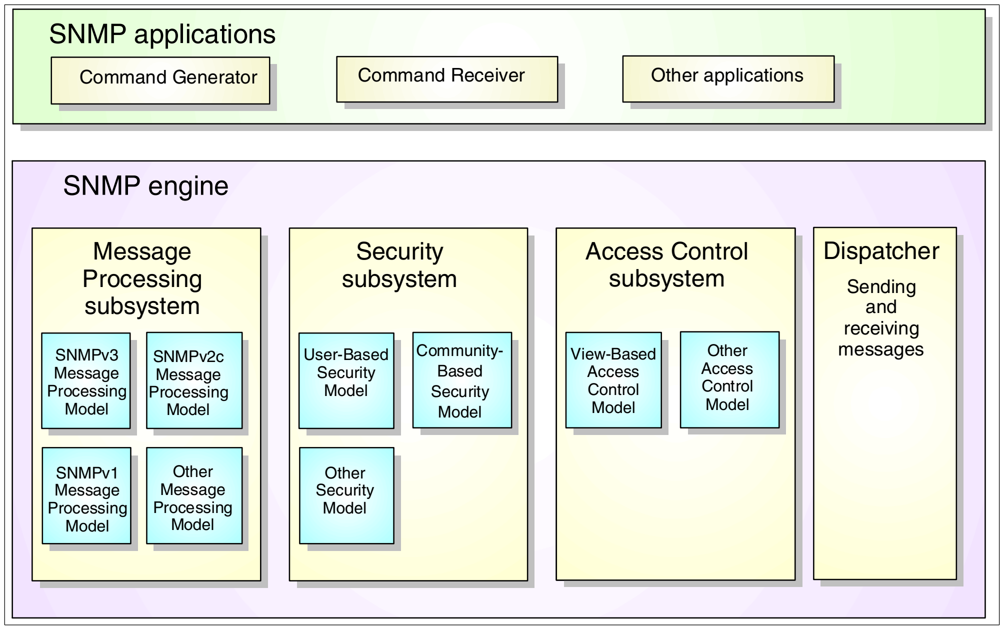
Both the authentication and encryption keys must be manually configured at each SNMP agent or manager. This isn’t as great of a difficulty as it appears, since there should be very few locations on your network that require SNMP interactions.

5.8.1.2 VACM

The use of VACM involves defining collections of data (called views), groups of users of the data, and access statements that define which views a particular group of users can use for reading, writing, or receipt in a trap. This is a powerful and important tool.

VACM views can be used to limit the range of MIB objects that a SNMP management application can access. From there, the view can be restricted to read or write privileges. The view can also be restricted on whether traps or informs can be sent to a given user.

See Figure 94 for more information on USM and VCAM.



SNMP applications
Command Generator
Command Receiver
Other applications
SNMP engine
Message Processing subsystem
SNMPv3 Message Processing Model
SNMPv2c Message Processing Model
SNMPv1 Message Processing Model
Other Message Processing Model
Security subsystem
User-Based Security Model
Community-Based Security Model
Other Security Model
Access Control subsystem
View-Based Access Control Model
Other Access Control Model
Dispatcher
Sending and receiving messages

Figure 94. SNMP subsystem processes

5.8.1.3 Dynamic changes

SNMPv3 also introduces the ability to configure dynamically the SNMP agent using `SNMP SET` commands against the MIB objects that represent the agent's configuration. This dynamic configuration support enables addition, deletion, and modification of configuration entries either locally or remotely.

Remote modification of user keys can be especially useful, simplifying the manual configuration process. The `pwchange` utility can be used to generate new keyChange value. Use of the keyChange value enables keys to be changed over the wire without actually sending any keys.

5.8.1.4 In summary

From the above discussion it is apparent that SNMPv3 provides a reasonable level of security. However, SNMPv3 is not supported by as wide a range of platforms as the older versions. If your network includes SNMP hosts capable of only V1 or V2, then we recommend two solutions:

- Use IPSec to protect SNMP traffic if you are concerned about its integrity and authenticity.
- Use VACM for community-based requests.

USM can be used only when both the SNMP agent and the manager requesting the data support USM, as do the CS for OS/390 SNMP agent and the `osnmp` command. VACM can be used even for community-based requests, but doing so requires migration of existing community name and trap destination definitions.

Table 13 is a list of the advantages and disadvantages of using each type of security:

Table 13. SNMP security overview

SNMPv1/SNMPv2c Advantages	**SNMPv3 Disadvantages**
Traditional standards-based administrative model	Emerging standards-based administrative model
Easy to configure	More involved configuration options
SNMPv1/SNMPv2c Disadvantages	**SNMPv3 Advantages**
SNMPv1 and SNMPv2c allow particular IP addresses to access all data or no data	SNMPv3 allows a particular user to access particular data
Not very robust (password sent in PDU)	Robust (data integrity and data origin authentication)
Any user that can read data can also change data (for objects defined as read-write)	The ability to change data can be limited to specific users
No data confidentiality	Encryption available (separate product)
Configuration changes require restarting of SNMP agent	Configuration changes for USM and VACM can be made dynamically, either locally or remotely

For detailed information regarding implementation of OS/390 SNMP, see *Managing OS/390 TCP/IP with SNMP*, SG24-5866.

5.9 OS/390 LDAP server

As described in 2.6, “Lightweight Directory Access Protocol (LDAP)” on page 42, an LDAP server is extremely beneficial in providing a centralized directory for (among other data) security information relating to users of your network. Because of this, the LDAP server itself requires a very high level of security protection.

On the OS/390 platform, IBM HTTP Server exploits LDAP for authenticating requests from its clients and CS for OS/390 uses the LDAP directory to store some policies for Quality of Service (QoS). Furthermore, you can access the RACF user and group information through the LDAP interface.

Directories contain sensitive information that needs to be protected from unauthorized access and modification. When sending data over networks internally or externally, sensitive information may also need to be protected against eavesdropping and modification during transportation. There is a need to know who is requesting the information and who is sending it.

5.9.1 Security of the directory

The directory may contain many types of information ranging from publicly accessible data, such as e-mail addresses, to very sensitive data such as user passwords. Hence the appropriate level of security for the data in the directory must be defined. In particular, OS/390 LDAP Server provides:

1. Authentication	The requester must prove his or her or its identity to the directory. This is supported using certificates using SASL/SSL.
2. Access Control	The directory server only returns data that the requester is entitled to access. In other words, the requester must have adequate authorization. This is implemented through the use of access control lists (ACLs).
3. Integrity	Data needs to be reliably stored and transmitted so that alterations can be detected. SSL network transmissions are protected against alterations.
4. Confidentiality	Sensitive data transmitted to or from the directory or stored in the directory cannot be easily accessed without proper permission (authorization). User passwords can be stored encrypted in the directory. Network transmissions can be protected using SSL.

The LDAP server provides the following security mechanisms for authentication and privacy.

- Anonymous authentication

  This is useful for read-only access of directory data where that data is not sensitive, such as people's e-mail addresses or office numbers. Essentially, that data is accessible to anyone. To request anonymous authentication, simple authentication is performed with a distinguished name (DN) that is empty.

- Basic authentication (distinguished name and password)

  This provides authentication facilities with the DN and password transmitted over the network in clear text. The use of clear text passwords is not recommended over open networks when there is no authentication or encryption being performed by a lower layer, such as IPSec. Access (read or write) to directory data is granted based on DNs contained in the access control list of the object and/or attributes in the access request.

  OS/390 LDAP Server allows a user to be authenticated to the directory name space using the RACF user ID and password. The RACF identity becomes associated with the user's RACF-style distinguished name (see 5.9.3, "Controlling RACF users and groups through the LDAP interface" on page 188) on the bind operation.

- Simple Authentication and Security Layer (SASL)

  The Simple Authentication and Security Layer (SASL) is a framework for multiple authentication and encryption mechanisms for connection-oriented protocols, as described in RFC 2222. It has been added to LDAP Version 3 to overcome the authentication shortcomings of LDAP Version 2.

  In SASL, connection protocols, such as LDAP, IMAP, and so on are represented by profiles; each profile is considered a protocol extension that allows the protocol and SASL to work together. Among these are IMAP4, SMTP, POP3, and LDAP. Each protocol that intends to use SASL needs to be extended with a command to identify an authentication mechanism and to carry out an authentication exchange. LDAP Version 3 includes such a command: `ldap_sasl_bind()`. Optionally, a security layer can be negotiated to encrypt the data after authentication and thus ensure confidentiality. The IBM

SecureWay Directory running on Windows NT and AIX supports SASL authentication using the Challenge Response Authentication Mechanism with Message Digest 5 (CRAM-MD5) mechanism, which transmits message digests rather than the passwords themselves over the network.

In OS/390 V2R8, the SASL bind mechanism of EXTERNAL is supported by OS/390 LDAP Server. This means that the authentication on the bind is performed by the SSL client authentication that was performed on the initial connection from the client. At the time of writing, OS/390 LDAP Server does not support SASL/CRAM-MD5 authentication.

- Secure Sockets Layer (SSL) certificate authentication and encryption

  As discussed in 2.4, “Secure Sockets Layer” on page 36, the Secure Sockets Layer (SSL) is, strictly speaking, not part of LDAP. It is a lower-level authentication and encryption service that higher-level applications, such as LDAP or HTTP, can use and rely on. It provides strong authentication using certificates and strong encryption using DES, Triple DES, RC2 or RC4 for securing network traffic. SSL is very widely used in the industry.

5.9.2 Using SSL communication

Much sensitive data, such as user passwords, X.509 certificates, private key, security policy and so forth, is stored in an LDAP directory. If a malicious user steals this information by using LAN analyzers or packet trace programs, not only the directory itself but also other systems or applications such as Web servers will be exposed to great danger. The key to protecting your data is to encrypt data and maintain data integrity. SSL is the de facto standard protocol, especially in HTTP communications, to achieve data encryption and data integrity.

OS/390 LDAP Server contains the ability to protect LDAP access with SSL. OS/390 LDAP Server supports server authentication. In addition, client authentication has been introduced in OS/390 V2R8.

With server authentication, the LDAP server must have a digital certificate (based on the X.509 standard). This digital certificate is used to authenticate the LDAP server to the LDAP client application. The LDAP server supplies the client with the LDAP server's X.509 certificate during the initial SSL handshake. If the client validates the server's certificate, then a secure, encrypted communication channel is established between the LDAP server and the LDAP client application.

In addition, if the LDAP server is configured for server and client authentication, and the client sends a digital certificate on the initial SSL handshake, it must be validated by the LDAP server before the secure encrypted communication channel is established between them.

Client authentication by the LDAP server facilitates the use of the SASL bind mechanism of EXTERNAL by an LDAP client. The bind identity becomes the distinguished name in the client digital certificate.

5.9.2.1 Establishing server authentication

To establish SSL server authentication between the LDAP server and an LDAP client, you have to do the following:

- Create a server certificate and keyring using the GSKKYMAN utility.
- Create a key database for an LDAP client.

- Specify the SSL-related options in the LDAP configuration file.

In the following discussion, we show you how to establish the SSL environment. For further information about the certificate management on OS/390 or z/OS, refer to Appendix D, “OS/390 certificate management using RACF” on page 321.

1. Create a server certificate

   We created a self-signed certificate for the OS/390 LDAP server.

Key database file	: /etc/ldap/ssl/ldapsrv.kdb
Key database password file	: /etc/ldap/ssl/ldapsrv.sth
Certificate file	: /etc/ldap/ssl/ldapsrv.crt

   Though a self-signed certificate is useful for test purposes, if you want to open your directory to business partners or users on the Internet, you had better request a trusted CA to issue your server certificate.

2. Create a key database for an LDAP client

   During the SSL handshake protocol, an LDAP client must ensure that a server certificate sent from the LDAP server was really issued by a trusted Certificate Authority.

   If you select a certificate by a trusted CA such as IBM Vault Registry, Thawte, RSA or VeriSign, you only create a key database because these CAs’ certificates are automatically included in any key database created by the GSKKYMAN utility.

   On the other hand, if you create a self-signed certificate for the LDAP server, you need to register this certificate into a client key database as a trusted CA certificate.

   Figure 95 and Figure 96 on page 185 show how to create a key database for an LDAP client and store a self-signed server certificate into a client’s key database.

```
TAKADA @ :/u/takada/ldap>gskkyman


           IBM Key Management Utility

Choose one of the following options to proceed.

   1  - Create new key database
   2  - Open key database
   3  - Change database password

   0  - Exit program

Enter your option number: 1
Enter key database name or press ENTER for "key.kdb": ldapclient.kdb  1
Enter password for the key database.......>******                     2
Enter password again for verification.....>******
Should the password expire? (1 = yes, 0 = no) [1]: 1
Enter password expiration time (number of days) or press ENTER for 60 days:

The database has been successfully created, do you want to continue to work wit
 the database now? (1 = yes, 0 = no) [1]: 1
```

Figure 95. Creating a key database for an LDAP client

When you try to bind to OS/390 LDAP Server using SSL, you must specify a key database name (1) and a password (2) to open this database.

```
            Key database menu

Current key database is /u/takada/ldap/ldapclient.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 6
Enter certificate file name or press ENTER for "cert.arm": ldapsrv.crt
Enter a label for this key................> LDAP Server Certificate

Please wait while certificate is stored...

Your request has completed successfully, exit gskkyman? (1 = yes, 0 = no) [0]:1
```

Figure 96. Storing a self-signed server certificate into a client's key database

Now this self-signed certificate has been registered as a trusted CA certificate in a key database for an LDAP client.

3. Specify the SSL-related options in the LDAP configuration file

   Finally you have to set several SSL-related options in the LDAP server configuration file (normally /etc/ldap/slapd.conf).

```
securePort 636                                                  1
security ssl                                                    2
sslAuth serverAuth                                              3
sslKeyRingFile /etc/ldap/ssl/ldapsrv.kdb                        4
sslKeyRingPWStashFile /etc/ldap/ssl/ldapsrv.sth                 5
sslCipherSpecs 12288                                            6
```

1 Specify a port used for SSL communications. The default is 636.

2 Set `ssl` in the `security` option. If you use only SSL communications, set `security sslonly`.

3 Specify `serverAuth` in the `sslAuth` option for server authentication.

4 Specify a name of the server key database file you created before.

5 This is a password file to open a key database specified in 4. The contents of this password file are encrypted. Instead of this option, you can set a password that you specified when you created a key database with the `sslKeyRingFilePW` option. However, you should not specify your password in a

clear text because if malicious users find this configuration file, they can easily open the server key database specified here using this password.

6 This option is used to specify the SSL cipher specifications that will be accepted from clients. You can use the following ciphers for SSL communication between the LDAP server and LDAP clients only if the clients support these ciphers:

U.S. and Canada version:

- SLAPD_SSL_RC4_MD5_US (0800)
- SLAPD_SSL_TRIPLE_DES_SHA_US (0100)
- SLAPD_SSL_DES_SHA_US (0200)
- SLAPD_SSL_RC2_MD5_EXPORT (1000)
- SLAPD_SSL_RC4_MD5_EXPORT (2000)

Export version:

- SLAPD_SSL_RC2_MD5_EXPORT (1000)
- SLAPD_SSL_RC4_MD5_EXPORT (2000)

You need to specify an integer value with the `sslCipherSpecs` option. This integer is the decimal representation of the OR-ed bit mask defined by the hexadecimal values above.

In our example, we specified 12288 (hex 3000). In other words, we selected RC2_MD5 and RC4_MD5 as the SSL cipher specifications.

To make the LDAP server recognize these SSL-related options in the configuration file, you must recycle the server.

```
TAKADA @ RA03:/u/takada/ldap>onetstat -p TCPIPA -c |grep LDAPSRV
LDAPSRV  0000129F 0.0.0.0..389         0.0.0.0..0           Listen
LDAPSRV  000012A0 0.0.0.0..636         0.0.0.0..0           Listen
```

Now you can establish SSL communication between OS/390 LDAP Server and an LDAP client. Figure 97 shows an example of searching an LDAP entry through SSL using the `ldapsearch` command-line utility:

```
TAKADA @ RA03:/u/takada/ldap>cat ldapsearch.ssl.sh
ldapsearch -h 9.24.104.113 -Z -K ldapclient.kdb \ 1
  -P takada "cn=Michiyasu Takada"
TAKADA @ RA03:/u/takada/ldap>ldapsearch.ssl.sh
cn=Michiyasu Takada, ou=ITSO, ou=Raleigh, o=IBM_US, c=US
objectclass=organizationalPerson
cn=Michiyasu Takada
sn=Takada
telephonenumber=1-812-855-7743
title=OS/390 USS, LDAP, Firewall
postalcode=4609
```

Figure 97. Searching an LDAP entry through an SSL connection

The `-Z` option is used for a secure SSL connection to communicate with the LDAP Server. The `-Z` option is not supported by non-SSL versions of this tool.

The `-K` option is used to specify the name of the SSL key database file for an LDAP client. If the key database file is not in the current directory, specify the fully qualified key database file name. If a key database file name is not specified, this utility will look for the presence of the SSL_KEYRING environment variable with an associated file name. Otherwise, no key database file will be used for server authentication and the default trusted Certificate Authority roots will be used. This parameter is ignored if `-Z` is not specified.

The `-P` option is used to specify the key database file password. This password is required to access the encrypted information in the key database file (including the private key). If the key database file does not contain a private key, which is possible on the LDAP client, then the key database file may have been created without a password. In this case, there is no need to specify a password here. This parameter is ignored if `-Z` is not specified.

Similarly, you can use the other command-line utilities such as `ldapadd`, `ldapmodify`, `ldapmodrdn`, `ldapdelete` and `ldapcp` through the SSL communication channel.

For more detailed information about the LDAP command-line utility, see *OS/390 Security Server LDAP Server Administration and Usage Guide*, SC24-5861.

As discussed in 5.1.8, “Retrieving LDAP information” on page 129, IBM HTTP Server for OS/390 can retrieve authentication information, such as a user password or X.509 certificate, or configuration information from LDAP servers. It is better to use SSL communication between the HTTP Server (in this case, an LDAP client) and the LDAP server because this information is sensitive and should be protected from eavesdropping or altering. For more information on how to establish SSL connection between IBM HTTP Server for OS/390 and an LDAP server, see *IBM HTTP Server for OS/390 Planning, Installing, and Using*, SC31-8690.

Furthermore, if you want to make your own LDAP client application SSL-enabled, you need to add two extra function calls, ldapssl_client_init() and ldapssl_init(), to the source code of your application. For information on how to create your own SSL-enabled LDAP application, see *OS/390 Security Server LDAP Client Application Development Guide and Reference*, SC24-5878.

5.9.2.2 Establish client authentication

As discussed in 5.9.1, “Security of the directory” on page 181, the SASL bind mechanism of EXTERNAL is supported in OS/390 V2R8 or later. If you select the SASL bind mechanism, the authentication on the bind is performed by the SSL client authentication.

To use SASL bind, the following steps must occur:

- The LDAP server must be configured and started with `sslAuth` set to `serverClientAuth` in the server configuration file so that the server can authenticate the client:

```
sslAuth serverClientAuth
```

- The client chooses SSL communication with the LDAP server through the use of `ldap_ssl_client_init` and `ldap_ssl_init`, and sends its certificate on `ldap_ssl_client_init` by having a client certificate with a private key and marking it as the default in its key database.

- The LDAP server authenticates the client's certificate (the LDAP server's key database contains and trusts the CA certificate of the signer of the client's certificate, or if the client is using a self-signed certificate the server contains and trusts the client's certificate).

  In our test environment, we created a self-signed certificate. Therefore we had to store an LDAP client's certificate in the server key database (/etc/ldap/ssl/ldapsrv.kdb) in the same way as shown in Figure 96 on page 185.
- The client performs a SASL bind with the LDAP server through the use of ldap_sasl_bind with the mechanism of EXTERNAL.

Again we examined the SSL client authentication using the same shell script shown in Figure 97 on page 186 and succeeded in searching an LDAP entry. At that time, we obtained the LDAP server trace to confirm if an LDAP client's certificate was really sent or not. The following is a part of the trace you can obtain by setting the -d flag in the LDAP server's started procedure:

```
Trying to extract client DN from CMS
SSLSupport_GetClientDN called
rc returned from CMS is 0
Client DN extracted from CMS is CN = LDAP Client, OU = ITSO, O = IBM_US,
L = Raleigh, stateOrProvinceName = North Carolina, C = US .
```

At this point, the LDAP server will consider the bind DN of the client for authorization purposes to be the client's DN as transmitted in the client's certificate on the ldap_ssl_client_init.

5.9.3 Controlling RACF users and groups through the LDAP interface

Since OS/390 V2R7, the OS/390 LDAP server has provided the ability to access the user and group information stored in the RACF database through the LDAP interface. This enables you to add, list, modify and delete the RACF users or groups from the LDAP-enabled clients on any platforms such as Windows NT, UNIX, OS/400 and OS/390. Only if the clients support RACF-related object classes and attributes do you have the option of consolidating all user information, for example, for some Web servers running on various platforms, to the RACF database.

You can manipulate the RACF user and group information through the LDAP interface, in a similar manner to the following RACF commands:

- `adduser`
- `addgrp`
- `altuser`
- `altgrp`
- `deluser`
- `delgrp`
- `listuser`

Even if you access the RACF database and execute the above commands through the LDAP interface, you must have the RACF authority to add, modify, list and delete the RACF user and group information.

5.9.3.1 Configuring the server to run with the RACF database

To enable the LDAP server to run with the RACF database, you have to modify the LDAP configuration file, the slapd.conf file. Figure 98 explains the RACF-related options in the LDAP configuration file:

```
include /etc/ldap/slapd.at.system
include /etc/ldap/slapd.at.conf
include /etc/ldap/slapd.at.racf                                   1
include /etc/ldap/slapd.oc.system
include /etc/ldap/slapd.oc.conf
include /etc/ldap/slapd.oc.racf                                   2

port  389
security none
maxthreads 0
maxconnections 0
waitingthreads 0
timelimit 3600
sizelimit 500
adminDN "racfid=kakky,profiletype=user,sysplex=ralplex1"          3
adminPW password                                                  4

database sdbm GLDBSDBM                                            5
suffix   "sysplex=ralplex1"                                       6
readOnly off
```

Figure 98. LDAP configuration file to run with RACF database

1 Add the RACF-related attribute definition file. This includes many LDAP-style attributes to be mapped to RACF attributes.

2 Add the RACF-related object class definition file.

3 Specify RACF administrator's distinguished name (DN). This DN is established as a RACF-style DN based upon a RACF user ID. If an administrator's RACF user ID is "admin" and you specify "sysplex=sysplexa, o=IBM_US, c=US" as the `suffix` option explained in **6**, you must obey the following naming rule:

```
adminDN "racfid=admin, profiletype=user, sysplex=sysplexa, o=IBM_US, c=US"
```

This option is required when you want to create, list, modify or delete the RACF user or group information by using the LDAP interface.

4 According to *OS/390 Security Server LDAP Server Administration and Usage Guide*, SC24-5861, if you use the RACF database as an LDAP back-end server, you do not need to specify the RACF administrator's password in this option. In this case, when a client specifies the RACF administrator's password in the LDAP command-line utilities or APIs, the LDAP server checks the RACF database.

However, in our test environment, when we did not specify this option, we could not access the RACF user and group information. Therefore, in this option we had to specify our administrator's RACF password.

5 OS/390 LDAP Server supports RDBM (DB2) and SDBM (RACF) as a back-end server. You can use DB2 and RACF as a back-end server at the same time. If you need to access the RACF information through the LDAP interface, you have to specify at lease `sdbm` in the database option. In addition to that, specify `GLDBSDBM`. This is the name of the LDAP dynamic link library for RACF access.

6 When accessing RACF information, the keyword `sysplex` is required to be present in the suffix for SDBM. We specified our sysplex name “ralplex1” in this option. This might be confusing. You do not need to establish a sysplex environment in order to use this function. Even if you do not have a sysplex name, you can specify any name. The only thing you have to do is specify the keyword `sysplex`.

5.9.3.2 RACF namespace entries

When the SDBM database is used to make RACF information accessible over the LDAP protocol, the top three entries in the hierarchy are reserved, read-only, and generated by the server. The purpose of these reserved entries is to enable a hierarchical representation of RACF names and groups in a sysplex. For example, the top three entries in Figure 99 are:

- sysplex=ralplex1 (suffixDN)
- profileType=user, sysplex=ralplex1
- profileType=group, sysplex=ralplex1

The value of sysplex in the top DN is generated from the suffix line in the slapd.conf file for the SDBM database entry (see 5.9.3.1, “Configuring the server to run with the RACF database” on page 189). As discussed in 5.9.3.1, “Configuring the server to run with the RACF database” on page 189, the keyword sysplex is required to be in the suffix. Figure 99 is a high-level diagram of the RACF back end.

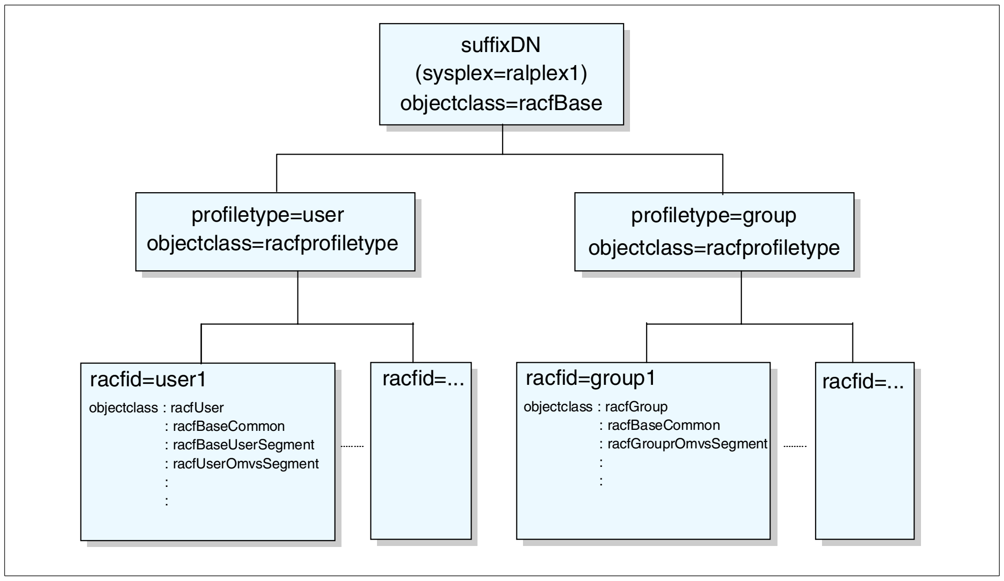


Figure 99. Directory hierarchy using SDBM

5.9.3.3 Accessing the RACF database using the LDAP utilities

You can access RACF information using the LDAP-enabled client. In this section, we show you how to access the RACF user and group information using the LDAP command-line utilities such as `ldapadd`, `ldapmodify`, `ldapdelete` and `ldapsearch`:

1. Listing the existing RACF user information

You can see the existing RACF user information by means of the `ldapsearch` command:

```
ldapsearch -h 9.24.104.113 -p 389 \
  -D "racfid=kakky,profiletype=user,sysplex=ralplex1" -w password \    1
  -b "racfid=takada,profiletype=user,sysplex=ralplex1" \              2
  "objectclass=*"                                                     3
```

Figure 100. Syntax of the ldapsearch command

We created shell scripts for issuing the `ldapsearch` command because this command tends to be very long.

1 Specify the RACF administrator's DN with the -D flag, which is specified with a keyword "adminDN" in the LDAP configuration file (slapd.conf), and the password with the -w flag, which is stored in the RACF database. This user must have RACF authority to list RACF user information.

2 Use the search base as the starting point for the search instead of the default. If -b is not specified, this utility will examine the LDAP_BASEDN environment variable specified in the export command, /etc/profile or $HOME/.profile for a search base definition. As described in 5.9.3.2, "RACF namespace entries" on page 190, the RACF user's directory entry name is automatically fixed.

3 For this search base, the only search filter that is supported is "`objectclass=*`".

For more information about what search filters are supported for each search base, see *OS/390 Security Server LDAP Server Administration and Usage Guide*, SC24-5861.

Figure 101 shows the result of the `ldapsearch` command. You can see various RACF user information belonging to User base, TSO segment, OMVS segment, and so on.

In addition you can modify or delete the existing RACF user by using the `ldapadd`, `ldapmodify` and `ldapdelete` commands, only if you specify the administrator's DN with the -D flag.

```
TAKADA @ RA03:/u/takada/ldap/racf>ldapsearch.sh
racfid=TAKADA,profiletype=USER,sysplex=RALPLEX1
objectclass=racfUser
objectclass=racfBaseCommon
objectclass=racfBaseUserSegment
objectclass=racfUserOmvsSegment
objectclass=SAFTsoSegment
racfid=TAKADA
racfprogrammername=MICHIASU TAKADA
racfowner=racfid=HAIMO,profiletype=USER,sysplex=RALPLEX1
racfauthorizationdate=99.207
racfdefaultgroup=racfid=WTCRES,profiletype=GROUP,sysplex=RALPLEX1
racfpasswordchangedate=99.218
racfpasswordinterval=180
racfattributes=NONE
racfrevokedate=DECEMBER 31, 2000
racfresumedate=NONE
racflastaccess=99.235/18:21:52
racfclassname=NONE
racfinstallationdata=RESIDENT
racfdatasetmodel=NO-MODEL-NAME
racflogondays=ANYDAY
racflogontime=ANYTIME
racfsecuritylevel=NONE SPECIFIED
racfsecuritycategorylist=NONE SPECIFIED
racfsecuritylabel=NONE SPECIFIED
racfomvsuid=0000050011
racfomvshome=/u/takada
racfomvsinitialprogram=/bin/sh
safaccountnumber=222222
safdefaultloginproc=$RISC
saflogonsize=00004096
safmaximumregionsize=00000000
safuserdata=0000
safdefaultcommand=
```

Figure 101. Listing the RACF user information using the ldapsearch command

2. Creating a new RACF group

   To create a new RACF group, you can use the `ldapadd` or `ldapmodify` command. These commands are identical to the RACF `addgroup` command. Figure 102 is an example of the shell script to issue the `ldapadd` command.

   In the same way as the `ldapsearch` command, you have to specify the administrator's distinguished name with the -D flag and his or her password with the -w flag. Then you specify the file (with the -f flag) that contains the entry you want to add using LDIF style input as shown in Figure 102:

```
ldapadd -h 9.24.104.113 -p 389 \
  -D "racfid=kakky,profiletype=user,sysplex=ralplex1" -w password \
  -f /u/takada/ldap/racf/addgrp
```

Figure 102. Syntax of the ldapadd command

   When using LDIF mode style input, attribute types and values are delimited by colons (or double colons (::)). Furthermore, individual changes to attribute values are delimited with a `changetype:` input line. The general form of input lines for LDIF mode is:

```
change_record
```

```
<blank line>
change_record
<blank line>
   :
   :
```

An input file in LDIF mode consists of one or more change_record sets of lines which are separated by a single blank line. Each change_record has the following form:

```
dn:distinguished_name
[changetype:{modify|add|modrdn|delete}]
[change_clause
    :
    :
```

Thus, a change_record consists of a line indicating the distinguished name of the directory entry to be modified, an optional line indicating the type of modification to be performed against the directory entry, along with one or more change_clause sets of lines.

```
dn: racfid=ldapgrp, profiletype=group, sysplex=ralplex1               1
changetype: add                                                       2
racfid: ldapgrp                                                       3
objectclass: racfGroup                                                4
objectclass: racfGroupOmvsSegment                                     5
```

Figure 103. Input for adding the RACF group through the ldapadd command

1 Specify the distinguished name for the RACF group entry you want to register. The naming rule is followed by the hierarchy shown in Figure 99 on page 190.

2 Specify "add" as the change type.

3 Specify the entry name. In this case, we create the RACF group entry named "ldapgrp".

4 Specify the objectclass "racfGroup" because racfid belongs to this object class.

5 Because we will add the OMVS segment to this group later, we specify the object class "racfGroupOmvsSegment" corresponding to the RACF group's OMVS segment.

Now you can add the RACF group ldapgrp into the RACF database through the `ldapadd` command:

```
TAKADA @ RA03:/u/takada/ldap/racf>ldapadd.sh
ldap_init(9.24.104.113, 389)
add racfid:
        ldapgrp
add objectclass:
        racfGroup
        racfGroupOmvsSegment
adding new entry racfid=ldapgrp, profiletype=group, sysplex=ralplex1
add complete
```

Then we add the OMVS segment to the RACF group profile ldapgrp using the `ldapmodify` command:

```
ldapmodify -h 9.24.104.113 -p 389 \
  -D "racfid=kakky,profiletype=user,sysplex=ralplex1" -w password \
  -f /u/takada/ldap/racf/altgrp
```

The following is the LDIF style input we used to add the OMVS segment.

```
dn: racfid=ldapgrp, profiletype=group, sysplex=ralplex1
changetype: modify                                                  1
add: racfOmvsGroupId                                                2
racfOmvsGroupId: 777                                                3
-                                                                   4
```

1 In this case, you specify modify as the change type.

2 Add the racfOmvsGroupId attribute that corresponds to the GID OMVS segment attribute.

3 We set the racfOmvsGroupId attribute to 777. This is the same as setting the GID for the group ldapgrp to 777.

4 If an `add:attribute_name`, `replace:attribute_name`, or `delete:attribute_name` line (a change indicator) is specified, a line containing a hyphen (–) is expected as a closing delimiter for the changes.

```
TAKADA @ RA03:/u/takada/ldap/racf>ldapmodify.sh
modifying entry racfid=ldapgrp, profiletype=group, sysplex=ralplex1
```

Now you can add the attribute "racfOmvsGroupId" to the RACF group entry ldapgrp. We executed the shell script shown in Figure 104, and Figure 105 shows the result. As you see, we succeeded in adding the new RACF group entry ldapgrp and the OMVS segment.

```
ldapsearch -h 9.24.104.113 -p 389 \
  -D "racfid=kakky,profiletype=user,sysplex=ralplex1" -w password \
  -b "racfid=ldapgrp,profiletype=group,sysplex=ralplex1" \
  "objectclass=*"
```

Figure 104. Listing the information in RACF for ldapgrp using the ldapsearch command

```
TAKADA @ RA03:/u/takada/ldap/racf>ldapsearch.sh
racfid=LDAPGRP,profiletype=GROUP,sysplex=RALPLEX1
objectclass=racfBaseCommon
objectclass=racfGroup
objectclass=racfGroupOmvsSegment
racfid=LDAPGRP
racfsuperiorgroup=racfid=SYSPROG,profiletype=GROUP,sysplex=RALPLEX1
racfowner=racfid=KAKKY,profiletype=GROUP,sysplex=RALPLEX1
racfinstallationdata=NO INSTALLATION DATA
racfdatasetmodel=NO MODEL DATA SET
racfgroupnotermuac=TERMUACC
racfsubgroupname=NO SUBGROUPS
racfomvsgroupid=0000000777
```

Figure 105. Listing information about the group ldapgrp

For more detailed information about LDIF mode style input, see *OS/390 Security Server LDAP Server Administration and Usage Guide*, SC24-5861.

3. Creating a new RACF user

   Similar to creating a new RACF group, you can add a new RACF user by using the `ldapadd` or `ldapmodify` command. The following is an example of the shell script used to issue the `ldapadd` command:

```
ldapadd -h 9.24.104.113 -p 389 \
  -D "racfid=kakky,profiletype=user,sysplex=ralplex1" -w athx6p \
  -f /u/takada/ldap/racf/adduser
```

   In this case, we created a new RACF user ldapusr.

```
dn: racfid=ldapusr,profiletype=user,sysplex=ralplex1
changetype:add
objectclass: racfUser                                                  1
objectclass: racfUserOmvsSegment                                       2
racfid: ldapusr                                                        3
racfDefaultGroup: ldapgrp                                              4
racfPassword: pswd                                                     5
racfOmvsUid: 50020                                                     6
racfOmvsHome: /u/ldapusr
racfOmvsInitialProgram: /bin/sh
```

Figure 106. LDIF mode style input to add a new RACF user

   1 To add a new RACF user, the objectclass racfUser is mandatory.

   2 When you want to add the OMVS segment to the RACF user profile, this objectclass is required to be specified.

   3 Specify a new RACF user's name in the racfid attribute.

   4 We specified ldapgrp, created earlier as a default RACF group for this user.

   5 You can specify an initial RACF password for this user in the racfPassword attribute.

   6 You can specify the OMVS segment for this user by setting the racfOmvsUid (UID), racfOmvsHome (HOME) and racfOmvsInitialProgram (PROGRAM) attributes.

   Now you can add a new RACF user ID ldapusr by executing the shell script ldapadd.sh:

```
TAKADA @ RA03:/u/takada/ldap/racf>ldapadd.sh
adding new entry racfid=ldapusr,profiletype=user,sysplex=ralplex1
```

   To confirm whether the user ID ldapusr is added successfully or not, you can use the `ldapsearch` command (Figure 107). You ought to be able to find the attributes we set in LDIF mode style input shown in Figure 106.

```
ldapsearch -h 9.24.104.113 -p 389 \
  -D "racfid=kakky,profiletype=user,sysplex=ralplex1" -w password \
  -b "racfid=ldapusr,profiletype=user,sysplex=ralplex1" \
  "objectclass=*"
```

Figure 107. The ldapsearch command to confirm the user information

```
TAKADA @ RA03:/u/takada/ldap/racf>ldapsearch.sh
racfid=LDAPUSR,profiletype=USER,sysplex=RALPLEX1
objectclass=racfUser
objectclass=racfBaseCommon
objectclass=racfBaseUserSegment
objectclass=racfUserOmvsSegment
racfid=LDAPUSR
racfprogrammername=UNKNOWN
racfowner=racfid=KAKKY,profiletype=USER,sysplex=RALPLEX1
racfauthorizationdate=99.235
racfdefaultgroup=racfid=LDAPGRP,profiletype=GROUP,sysplex=RALPLEX1
racfpasswordchangedate=00.000
racfpasswordinterval=180
racfattributes=NONE
racfrevokedate=NONE
racfresumedate=NONE
racflastaccess=UNKNOWN
racfclassname=NONE
racfinstallationdata=NO-INSTALLATION-DATA
racfdatasetmodel=NO-MODEL-NAME
racflogondays=ANYDAY
racflogontime=ANYTIME
racfsecuritylevel=NONE SPECIFIED
racfsecuritycategorylist=NONE SPECIFIED
racfsecuritylabel=NONE SPECIFIED
racfomvsuid=0000050020
racfomvshome=/u/ldapusr1
racfomvsinitialprogram=/bin/sh
```

Figure 108. Listing the information in RACF for LDAPUSR

Now you can access the UNIX shell through rlogin, Telnet or FTP using this user ID because this user ID already has the OMVS segment. In addition to the User base segment, Group base segment and OMVS segment shown in this section, you can set the following user RACF segments:

- TSO segment
- LANGUAGE segment
- CICS segment
- OPERPARM segment
- WORKATTR segment
- NetView segment
- DCE segment
- User OVM segment
- DFP segment

4. Deleting a RACF user ID

   Finally we show how to delete an existing RACF user ID through the LDAP interface. You can remove a RACF user profile for a distinguished name from RACF by using the `ldapdelete` command-line utility. This is the equivalent of the RACF `deluser` command:

```
ldapdelete -h 9.24.104.113 -p 389 \
  -D "racfid=kakky,profiletype=user,sysplex=ralplex1" -w password \
  "racfid=ldapusr,profiletype=user,sysplex=ralplex1"
```

Figure 109. Removing a RACF user profile using the ldapdelete command

The directory management tools available with the OS/390 are command line driven such as the `ldapadd` command. There are, however, also tools available that provide a Graphical User Interface (GUI) to manage an LDAP directory. In this section we will show examples of two of the available tools, for demonstration purposes only. These directory management tools are:

- IBM's SecureWay Directory Management Tool

  This tool is part of the SDK for the SecureWay Directory Server and can be downloaded from:

  `http://www.ibm.com/software/network/directory/downloads`

- The LDAP Browser/Editor by Jarek Gawor

  This tool is available for download from:

  `http://www-unix.mcs.anl.gov/~gawor/ldap`

  It is also commercially available through Argonne National Laboratory.

For further information about those GUI tools, refer to *OS/390 Security Server 1999 Updates: Installation Guide*, SG24-5629.

5.9.4 Using access control

Access control of information in the LDAP server is specified by setting up access control lists (ACLs). ACLs provide a means to protect information stored in an LDAP directory. Administrators use ACLs to restrict access to different portions of the directory, or specific directory entries. LDAP directory entries are related to each other by a hierarchical tree structure. Each directory entry (or object) contains the entry's distinguished name, a set of attributes and their corresponding values.

ACLs and groups may be created and managed through the LDAP Directory Server Access Control List and Group Administration utility (ldapcp) or the ldif2db program.

ACLs are represented by a set of attributes that appear to be a part of the entry. The attributes associated with access control, such as owner, ownerPropagate, acl and aclPropagate are unusual in that they are logically associated with each entry, but they can have values that depend upon other entries higher in the hierarchy. Depending upon how they are established, these attribute values can be explicit to an entry, or inherited from an ancestor entry.

Use of LDAP's SDBM database described in 5.9.3, "Controlling RACF users and groups through the LDAP interface" on page 188 allows a user to be authenticated to the directory namespace using the RACF ID and password. The RACF identity becomes associated with the user's RACF-style distinguished name on the bind operation. It is then possible to set up ACLs for namespace entries managed by the RDBM back end using RACF-style user and group DNs.

This controls access to RDBM database directory entities using the RACF user ID or group.

5.9.4.1 Access control attributes

Access to LDAP directory entries and attributes is defined by ACLs. Each entry in the directory contains a special set of attribute/value pairs which describe who is allowed to access information within that entry.

The various aspects of the ACL model are represented in six attributes that are part of every directory entry. These are:

- entry Owner

  This represents the owner of this particular directory entry. The entry Owner receives complete access to all attributes of the entry.

- owner Propagate

  You can set a TRUE or FALSE in this attribute. When you set TRUE, the owner should be propagated down the entire directory hierarchy.

- aclEntry

  See below for a description of this one.

- aclPropagate

  A TRUE or FALSE flag is set in this attribute. If TRUE is selected, the ACL should be propagated down the entire directory hierarchy.

- aclSource

  This is an attribute, that cannot be modified by the user, that identifies the directory entry with which the ACL information is associated.

- ownerSource

  Another attribute that cannot be modified by the user. It identifies the directory entry with which the owner information is associated.

5.9.4.2 aclEntry attribute

This is a multivalued attribute that describes access to attributes of the associated LDAP entry, as well as permissions on the entry itself. An aclEntry lists the following types of information:

- **Scope of the protection**

  This attribute defines who has rights to the entry (the ACL subjects).

- **Attribute access classes**

  This defines to what classes of attributes the subject has access.

- **Access permissions**

  This defines what rights the subject has.

1. Scope of the protection

   The scope of the protection is based on the following two types of privilege attributes:

access-id:	The distinguished name of an entity or entry being granted access
group:	The distinguished name of the group entity or entry being granted access

Privilege attributes take the form of type: name where type refers to either access-id or group and name is the distinguished name. The following is an example of ACL subjects:

```
access-id:cn=LDAP Administrator, ou=ITSO, ou=Raleigh, o=IBM_US, c=US    1
group:cn=OS390 IP Security, ou=Groups, o=IBM_US, c=US                   2
```

1 The DN type is `access-id` and the DN itself is `cn=LDAP Administrator, ou=ITSO, ou=Raleigh, o=IBM_US, c=US`.

2 The DN type is `group` and the DN itself is `cn=OS390 IP Security, ou=Groups, o=IBM_US, c=US`.

2. Attribute access classes

   Attributes requiring similar permission for access are grouped together in classes. Attributes are assigned to an attribute access class within the attribute definition files (lapd.at.* files). The three user-modifiable attribute access classes are:

   - Normal
   - Sensitive
   - Critical

   Each of these classes is discrete, which means that access to one class does not imply access to another class. Permissions are set with regard to the attribute class as a whole. The permissions set on a particular attribute class apply to all attributes within that class.

   For example, a person's name would typically be defined in the normal class. Perhaps a social security number would be considered sensitive, and any password information for the user would be considered critical.

3. Access permission

   There are six basic operations that may be performed on a directory object. From these operations, the base set of ACL permissions is taken. These six operations are:

   - Add an object
   - Delete an object
   - Read an attribute value
   - Write an attribute value
   - Search for an attribute
   - Compare an attribute value

   These permissions are discrete, which means that one permission does not imply another.

   The possible attribute class permissions are: read (r), write (w), search (s), and compare (c). Additionally, object permissions apply to the entry as a whole. These permissions are: add child entries (a) and delete this entry (d).

   This is the aclEntry attribute value syntax:

   ```
   subject_DN granted_rights
   ```

   The subject_DN is:

   ```
   "priv_attr_type:priv_attr_name"
   ```

where `priv_attr_type` is either access-id or group, and `priv_attr_name` is any valid DN that will represent the object (entry) to which privileges are being granted.

The `granted_rights` is specified as follows, where `obj_rights_list` is one or more elements of the set {ad}, and `attr_rights` is one or more elements of the set {rwsc}:

```
[:object:obj_rights][:normal:attr_rights][:sensitive:attr_rights][:critical
:attr_rights]
```

The following is an example of access permissions:

```
group:group:cn=OS390 IP Security, ou=Groups, o=IBM_US, c=US::normal:rsc                    1
access-id:cn=Michiyasu Takada, ou=ITSO, ou=Raleigh, o=IBM_US,c=US:object:ad                2
access-id:cn=Michiyasu Takada, ou=ITSO, ou=Raleigh,
o=IBM_US,c=US:normal:rwsc:sensitive:rwsc:critical:rsc                                      3
```

1 This means that members of the group `cn=OS/390 IP Security, ou=Groups, o=IBM_US, c=US` have permission to read, search and compare all attributes within the normal attribute access class.

2 In this case, the user corresponding to access-id `cn=Michiyasu Takada, ou=ITSO, ou=Raleigh, o=IBM_US,c=US` has permission to add and delete entries below the entry.

3 Similarly, access-id `cn=Michiyasu Takada, ou=ITSO, ou=Raleigh, o=IBM_US,c=US` has permission to read, write, search and compare both normal and sensitive attributes, and to read, search and compare critical attributes.

Although the aclEntry for access-id `cn=Michiyasu Takada, ou=ITSO, ou=Raleigh, o=IBM_US, c=US` was broken into separate aclEntry attribute values for object and user (normal, sensitive, critical) DN types (see **2** and **3**), they could be combined into one, if desired.

5.9.4.3 Propagation

ACLs and owner can be set to apply to just a particular entry or an entry and the entire subtree. Although both the entryOwner and aclEntry attributes can be propagated, their propagation is not linked in any way.

Entries on which an ACL has been placed are considered to have an explicit ACL. Similarly, if an owner has been set on a particular entry, that entry has an explicit owner. Since the two are not intertwined, an entry with an explicit owner may or may not have an explicit ACL, and an entry with an explicit ACL may or may not have an explicit owner. If these values are not explicitly present on an entry, the values are inherited from an ancestor node in the tree.

Each explicit ACL and owner applies to the entry on which it is set. Additionally, the value may apply to all descendants that do not have an explicitly set value. These values are considered propagated; that is, their values propagate downward through the directory tree. Propagation of a particular value continues until another propagating value is reached.

An object's ACL (or an object’s owner) can, therefore, be conceptually determined by the following algorithm:

Is there an explicit ACL set at the object?

If yes, then that is the object's ACL.

If no, then traverse the tree backward until an ancestor node is reached with a propagated ACL.

If no node is found with a propagated ACL, only the object owner and the administrator will be granted access to the object.

5.9.4.4 Creating ACLs and owners using LDIF format input to ldif2db

You can create ACLs and groups using the ldif2db program. By creating a file containing entries to be added that are in LDAP Directory Interchange Format (LDIF), ACLs and groups may be added to the LDAP directory using this file as input to the ldif2db program. Below is an example of an entry to be added with ACL attributes and owner attributes represented in LDIF format.

```
dn: cn=Michiyasu Takada,ou=ITSO,ou=Raleigh,o=IBM_US,c=US
objectclass: organizationalPerson
cn: Michiyasu Takada
sn: Takada
userPassword: abclmnxyz
aclSource: cn=Michiyasu Takada,ou=ITSO,ou=Raleigh,o=IBM_US,c=US
aclEntry: access-id:cn=LDAPAdm,ou=ITSO,ou=Raleigh,o=IBM_US,c=US:object:a
aclEntry:
access-id:cn=LDAPAdm,ou=ITSO,ou=Raleigh,o=IBM_US,c=US:normal:rwsc:sensitive:
rsc:critical:s
aclEntry: group:OS390 IP
Security,ou=Groups,o=IBM_US,c=US::normal:rsc:normal:rsc:sensitive:sc
ownerSource: cn=Michiyasu Takada,ou=ITSO,ou=Raleigh,o=IBM_US,c=US
entryOwner: access-id:cn=Tatsuhiko Kakimoto,ou=ITSO,ou=Raleigh,o=IBM_US,c=US
ownerPropagate: TRUE
```

This example creates an explicit ACL for the DN `cn=Michiyasu Takada,ou=ITSO, ou=Raleigh,o=IBM_US,c=US`. The ACL that is created contains the aclEntry values (specifying access permissions for this DN) for access-id `cn=LDAPAdm,ou=ITSO, ou=Raleigh,o=IBM_US,c=US` and group `OS390 IP Security,ou=Groups,o=IBM_US, c=US`.

You should note that whenever an aclEntry is present for a given DN, it must be accompanied by a corresponding aclSource whose attribute value equals that of the DN for which the ACLs are being created; if the DN for the entry does not equal the DN for the aclSource, the aclEntry values will be ignored.

The owner created for this DN is access-id `cn=Tatsuhiko Kakimoto,ou=ITSO, ou=Raleigh,o=IBM_US,c=US`, and will be propagated to children entries in the directory hierarchy, as directed by the ownerPropagate attribute being set to TRUE.

It should be noted that whenever an entryOwner is present for a given DN, it must be accompanied by a corresponding ownerSource whose attribute value equals that of the DN for which the entryOwner is being created. If the DN for the entry does not equal the DN for the ownerSource, the entryOwner will be ignored.

Instead of using the ldif2db utility to manage ACL entries, you can use the `ldapcp` command-line utility. For detailed information about how to use the `ldapcp` command, see *OS/390 Security Server LDAP Server Administration and Usage Guide*, SC24-5861.

5.10 OS/390 policy agent security

Before we can talk about security provided by the OS/390 policy agent, we should review the purposes of policy agent in general.

5.10.1 Quality of Service (QoS)

Current IP traffic service is characterized by:

- Best effort packet delivery
- No differentiation for users or applications when forwarding IP packets (some routers do analyze IP and TCP/UDP headers and make priority decisions based on port numbers - most often based on incompatible proprietary solutions)
- No service differentiation when discarding IP packets
- No consistent way to measure and report end-to-end or network service

This represents a problem in that the only way to provide "good" service is to have enough bandwidth to handle peak traffic at any time of day during which service-critical applications are used. Yet we need to ensure that the extra capacity is used by any single application or user (group) that needs it.

On the one hand we need a way to treat users differently. In terms of bandwidth and network delay, certain users may need better service levels than others. For example, the sales force may be given a higher priority in terms of bandwidth and delay, while the secretarial workload may be able to tolerate somewhat lower network speeds and availability in certain scenarios.

On the other hand, the network performance of certain applications (vs. user groups) may need to be managed in a special fashion. Our example in Figure 110 on page 203 illustrates how a company might have multiple webservers, only one of which should enjoy high-priority attention and network performance. There is a lower-priority corporate webserver to service the network, followed by a personal webserver that requires the lowest priority.

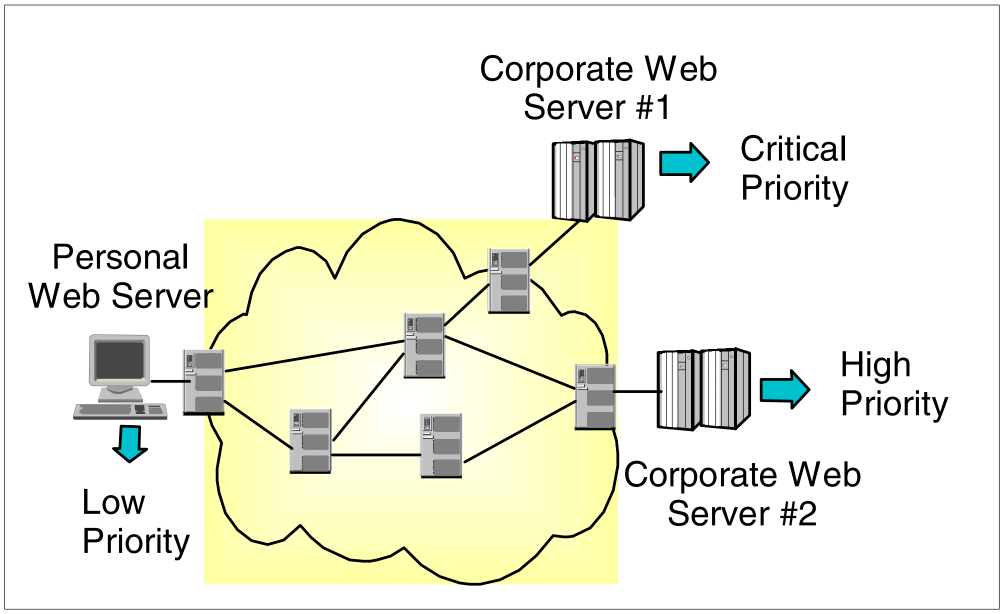


Figure 110. Same application treated differently

A variation on these themes is to provide different levels of service for different applications and user groups at different times of day. As you can see in Figure 111 on page 204, the TN3270 traffic for the query department enjoys the highest level of service between 8AM and 5PM daily, although the accounting department follows closely behind. The IT department is given the highest service level for FTP over any of the other departments. However, on the whole, FTP is prioritized below other application types. The Query department never demands a high level of service for any of the application types during off-shift hours.

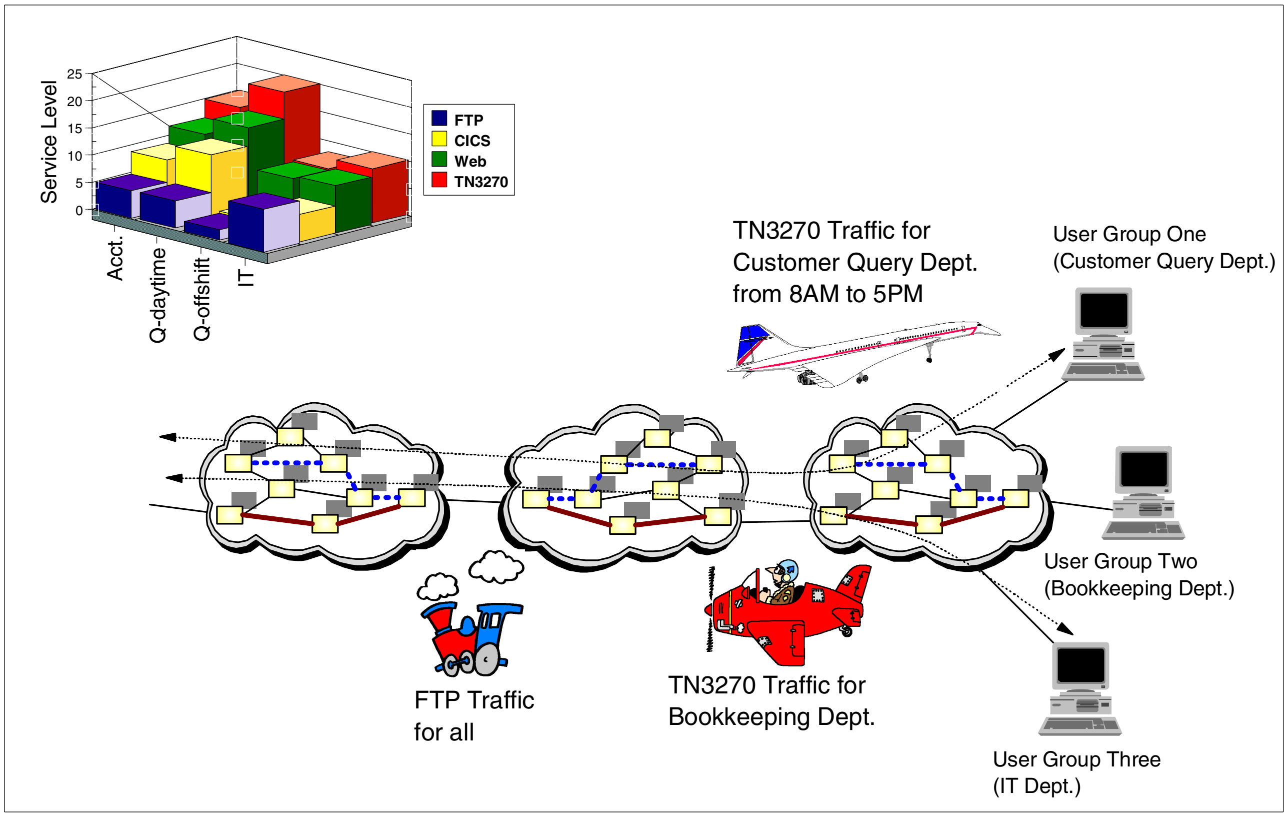


Figure 111. Levels of service based on users, applications, time of day

Business units and IT define in business terms the required service for applications, users, transaction volumes, response times, and availability and then establish Service Level Agreements (SLAs). Service Policies are encoded according to emerging standards and stored in an LDAP directory or a flat file.

The overall level of service provided to users and applications is termed *Quality of Service*, or *QoS*. There are two categories of QoS: *Differentiated Services*, which defines broad classes of service available to groups of users and categories of applications, and *Integrated Services*, which reserves resources for specific applications along the entire data path, end-to-end. We generally think of Integrated Services in terms of Resource ReserVation Protocol, or the RSVP protocol, defined in RFC 2205.

5.10.2 Policy agent and QoS

In Figure 112 we attempt to illustrate how Service Level Agreements can provide the information to define policies that are stored in either an LDAP server or a flat file. According to *IBM Communications Server for OS/390 IP Configuration Guide*, SC31-8725, "Policies are an administrative means to define controls for a network, in order to achieve the QoS levels promised by SLAs, or to implement security or resource balancing decisions." Network components may cooperate with each other to interpret and apply these policies to network traffic. The presence of a monitoring tool to compare actual network performance with desired network performance and possibly adjust values in the network dynamically to meet the desired expectations completes the picture.

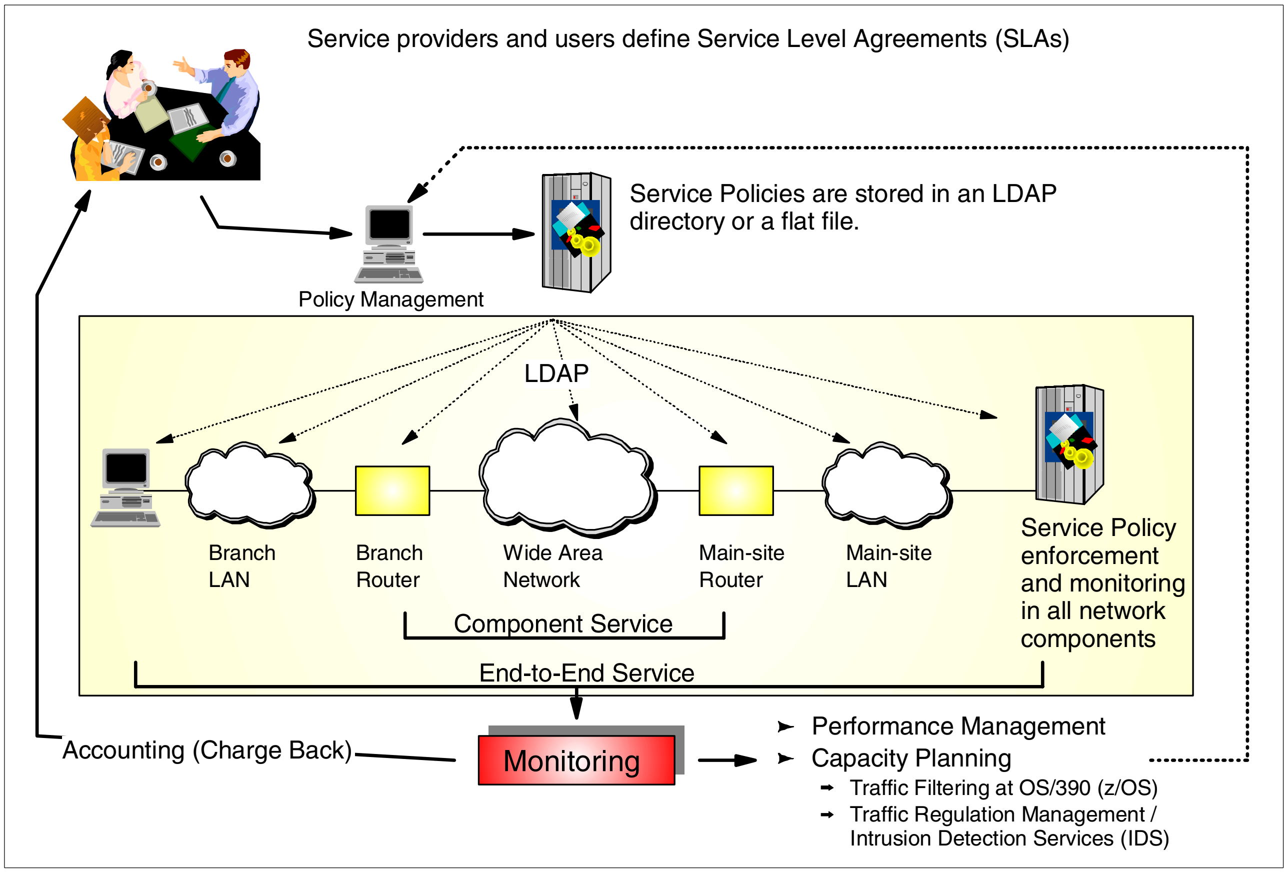

Service providers and users define Service Level Agreements (SLAs)
Service Policies are stored in an LDAP directory or a flat file.
Policy Management
LDAP
Branch LAN
Branch Router
Wide Area Network
Main-site Router
Main-site LAN
Service Policy enforcement and monitoring in all network components
Component Service
End-to-End Service
Accounting (Charge Back)
Monitoring
Performance Management
Capacity Planning
Traffic Filtering at OS/390 (z/OS)
Traffic Regulation Management / Intrusion Detection Services (IDS)

Figure 112. Performance management and traffic filtering

If network traffic filtering, traffic regulation, and *Intrusion Detection Services* (IDS) can be introduced as well, Figure 112 gives an overview of the complete capability of policy agent and its associated daemons in OS/390 (z/OS) as it has evolved over time. (Refer to 5.10.2.1, “Evolution of policy agent” on page 207.)

The policy agent in CS for OS/390 IP can be configured to set the bits in the Differentiated Services byte of the IP header. Prior to RFC 2474 the DS byte was known as the *Type of Service* or *TOS* byte. See Figure 113 to understand how bits 0-2 of the DS byte (TOS) can be set to designate a priority or "precedence" and how bits 3-5 can be used to specify other Quality of Service settings like delay, bandwidth, and reliability.

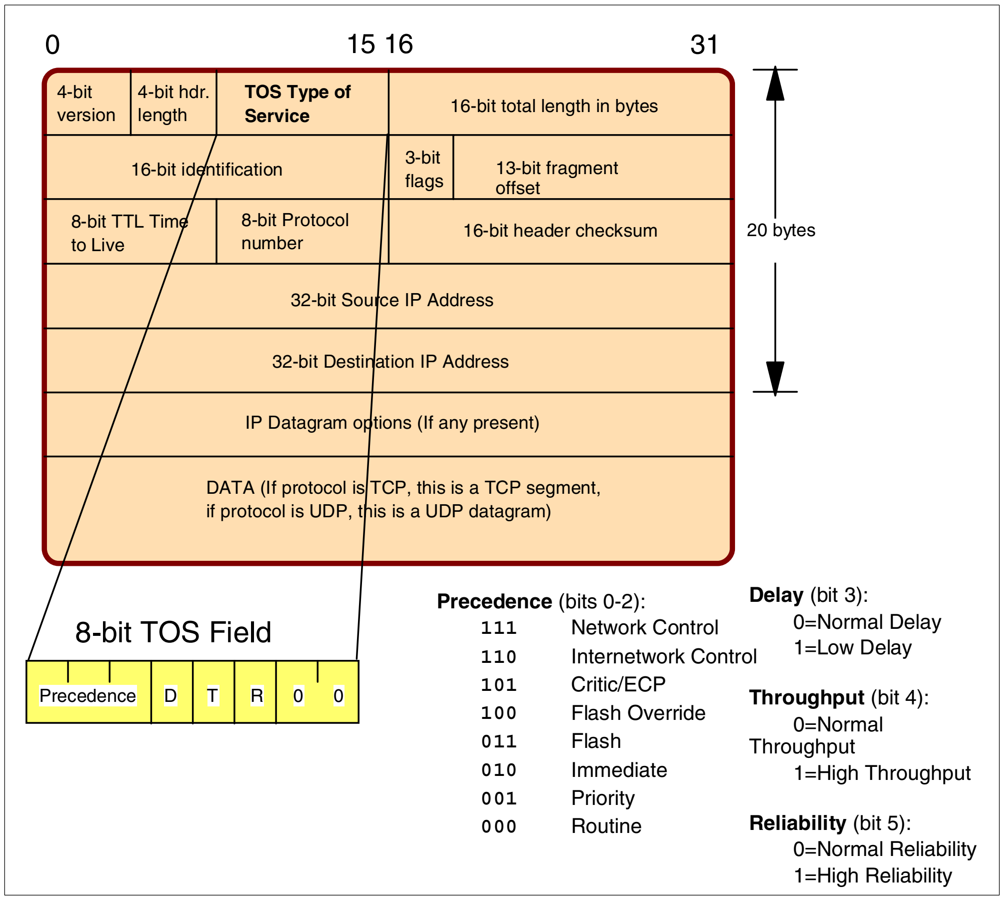
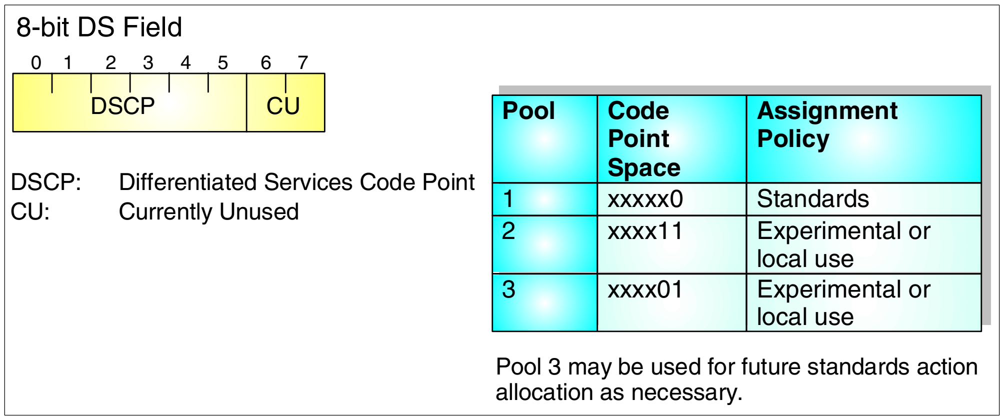


Figure 113. Type of Service field (TOS) - RFC 791 and RFC 1812

IP nodes along the network path may have been configured to act upon the settings of the DS byte after they have examined the contents of the IP header. In Figure 114 on page 206 you see how RFC 2474 has redefined the DS byte.



DSCP: Differentiated Services Code Point
CU: Currently Unused

Pool	Code Point Space	Assignment Policy
1	xxxxx0	Standards
2	xxxx11	Experimental or local use
3	xxxx01	Experimental or local use

Pool 3 may be used for future standards action allocation as necessary.

Figure 114. Differentiated services (DS) field in RFC 2474

The six bits of the DS field are used as a Differentiated Services Code Point (DSCP) to select the traffic class that a packet experiences at each node. The two-bit currently unused (CU) field is reserved and can be assigned in the future.

The DS byte specifications supersede the TOS byte definitions of RFC 1349, which changed the TOS value specification originally defined in RFC 791.

A DSCP maps to a Per Hop Behavior (PHB) in nodes within a DS domain. The eight DSCPs with xxx000 code points map to PHBs that are compatible with the behavior that is defined by the old TOS precedence bit setting. A DSCP of 000000 maps to a TOS 0 (default) behavior, which generally means: *best effort*.

In order to service packets containing DS headers, each router in a DS-capable network will maintain a number of output queues, each with varying priorities associated with them. Packets will be dispatched on the most appropriate queue depending on the DS byte setting.

Although DS is most often equated with the settings in the DS byte of the IP header, in fact Differentiated Services has a much broader meaning. It also includes limiting TCP connections, setting TCP connections rates and delay, and even controlling mean and peak traffic rates.

IP networks may need to provide for more than Quality of Service in order to differentiate among users and applications. Traffic balancing, performance monitoring, and security are additional concerns and policy agent in CS for OS/390 IP contains features to address all these areas.

For complete definitions of the TOS field, TOS byte, or DS field refer to the following RFCs:

- RFC 791 - *Internet Protocol*
- RFC 1349 - *Type of Service in the Internet Protocol Suite* (Obsoleted by RFC 2474)
- RFC 1812 - *Requirements for IP Version 4 Routers*
- RFC 2474 - *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*

5.10.2.1 Evolution of policy agent

Policy agent in OS/390, known also by its daemon name of *PAGENT*, has evolved since its introduction in OS/390 V2R6. At that release level the policy agent and RSVP agent including RSVP APIs were made available via the Web server. It was possible to set the TOS bits in the IP packet header and any defined filtering was performed at the IP layer. Downstream routers which examined the DS byte could enforce the policy designated for each packet.

At OS/390 V2R7, policy agent was integrated into CS for OS/390. Policies could be defined in the LDAP and not solely in a flat file. Filtering function moved to the TCP/UDP/RAW socket layers to improve performance. Policy agent became capable of defining the precedence bits in the IP packet; once the policies were installed in the CS IP stack, the stack could apply the precedence bits on behalf of traffic destined to use the Queued Direct IO (QDIO) interfaces of the OSA-Express.

At OS/390 V2R8, the Service Level Agreement (SLA) SNMP subagent provided the ability to monitor performance with the introduction of the SLAPM-MIB (defined in RFC 2758) and associated Trap capability. The RSVP agent was integrated into CS for OS/390; the RSVP agent communicates with RSVP agents on other platforms and reserves resources on certain types of outbound interfaces based upon RSVP policies defined in the policy agent and installed in the IP stack. The policy agent was enabled to negotiate RSVP information with the RSVP agent through a new policy agent API. Policies can be updated dynamically from either an LDAP server or a local policy file.

With OS/390 V2R10 IP, the policy agent provided automatic QoS enforcement by actually adjusting the TOS byte of lower priority connections to a "best efforts" scheme if high priority connections did not meet the SLA. This feature is called "traffic shaping." The Sysplex Distributor function of CS for OS/390 V2R10 IP takes advantage of policy agent definitions and their enforcement by the target IP stacks by using the information from Workload Manager (WLM) to determine the optimal sysplex image to which a specified traffic type should be distributed. QoS may be applied to non-SSL WebSphere Application Server V3.02 traffic based upon a scan of the URI at this release level.

Finally, a new stack function, Traffic Regulation and Management (TRM) and a new policy type, Traffic Regulation (TR), together with a new definition schema, syslogd logging support, and new commands for displaying active policies were added at OS/390 V2R10 IP. Traffic regulation (TR) policies provide a means to ensure that certain TCP traffic types, hosts, or networks do not monopolize system resources. Traffic Regulation Management daemon (TRMD) polls the stack and logs into syslogd the events relating to TRM and any violations of the policies set. With the LDAP server as a possible repository of the policy information, this level of OS/390 also introduces SSL security between the LDAP server and the policy agent. The LDAP server may use SSL protocols to feed the policy information to the policy agent, preventing eavesdropping by a third party.

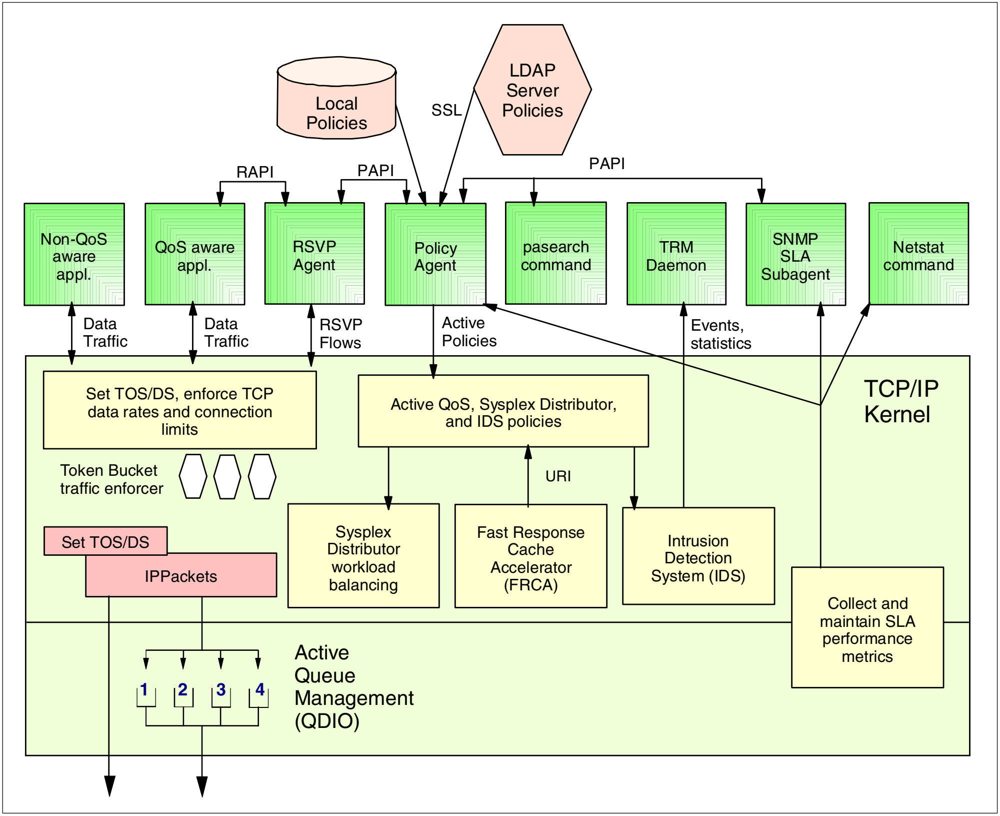
z/OS V1R2 introduces an enhanced real-time *Intrusion Detection Services* (IDS) via IDS policy and with new capabilities of the Traffic Regulation Management daemon (TRMD). At this level, Traffic Regulation is provided not only for TCP traffic but also for UDP. Additionally, scan detection and attack notification are available. Messages about possible security violations can not only be logged but also be shipped to the MVS console. The LDAP Version 3 Core Policy Schema is required for this IDS function. Other features include VLAN priority tagging for the bridges and switches on directly connected LANs and the application of QoS to SSL-enabled WebSphere Application Server V4.0 traffic.

5.10.2.2 Summary of policy agent functionality

Recall that policy agent defines the policies (DS, RSVP, TR and SD), but that the IP stack itself enforces the policies. In Figure 115 on page 209 you recognize many of the features of policy agent in CS for OS/390 IP that have been presented in a review of its evolution:

- The policy agent can define outbound priority queue values in the DS (TOS) byte for QDIO mode of the OSA-Express and for VLAN priority tagging with enforcement by the IP stack,
- Outbound packets for the non-QoS-aware applications can take advantage of the Differentiated Services settings that the IP stack has obtained from the policy agent,
- The RSVP agent can obtain RSVP policies from the policy agent in order to apply them to QoS-aware (Integrated Services) applications,
- The policy agent may read the defined policies from a flat file or from an LDAP server with which it may have established SSL communications,

- The policy agent installs and maintains these policies in the IP stack, making them the active service policies,
- The defined policies may assist with Sysplex Distributor workload balancing, with IDS, and with QoS settings for specific traffic types including WebSphere Application Server flows when Fast Response Cache Accelerator (FRCA) is enabled to scan Uniform Resource Identifiers (URIs),
- The installed policies are displayable with the `pasearch` command,
- The traffic regulation management functions of V2R10 and higher provide an additional means to regulate the sharing of system resources and, at z/OS V1R2, to perform Intrusion Detection Services (IDS) via policy agent definitions. Reports on Traffic Regulation and on IDS are provided by means of a daemon called Traffic Regulation Management Daemon,
- The SLA Performance Monitor MIB (SLABM MIB) subagent provides monitoring and the SLAPM MIB subagent communicates with the SNMP agent to provide GET and SET support for the various MIB tables and objects and even for trap generation,
- Sysplex Distributor policies can be used to limit the number of SD target stacks for a given kind of traffic and provide load-balancing across the sysplex.



Local Policies
SSL
LDAP Server Policies
RAPI
PAPI
PAPI
Non-QoS aware appl.
QoS aware appl.
RSVP Agent
Policy Agent
pasearch command
TRM Daemon
SNMP SLA Subagent
Netstat command
Data Traffic
Data Traffic
RSVP Flows
Active Policies
Events, statistics
TCP/IP Kernel
Set TOS/DS, enforce TCP data rates and connection limits
Active QoS, Sysplex Distributor, and IDS policies
Token Bucket traffic enforcer
URI
Set TOS/DS
IPPackets
Sysplex Distributor workload balancing
Fast Response Cache Accelerator (FRCA)
Intrusion Detection System (IDS)
Collect and maintain SLA performance metrics
1
2
3
4
Active Queue Management (QDIO)

Figure 115. Policy agent and OS/390 IP stack control

The service policies defined to the policy agent are installed in the IP stack with which a particular version of the policy agent is associated. The policies managed by policy agent become active when the TCP/IP stack has completed initialization. Policies stored in the HFS or in LDAP are refreshed dynamically if the policy agent configuration file resides in an HFS and has been modified. An interval specified on the TCPIMAGE statement in the configuration file also triggers refreshes of the policies at regular intervals; this allows policies that have been stored in MVS files to be modified and installed dynamically in the stack. A `MODIFY,procname,REFRESH` command is available at z/OS V1R2 to accomplish this task as well.

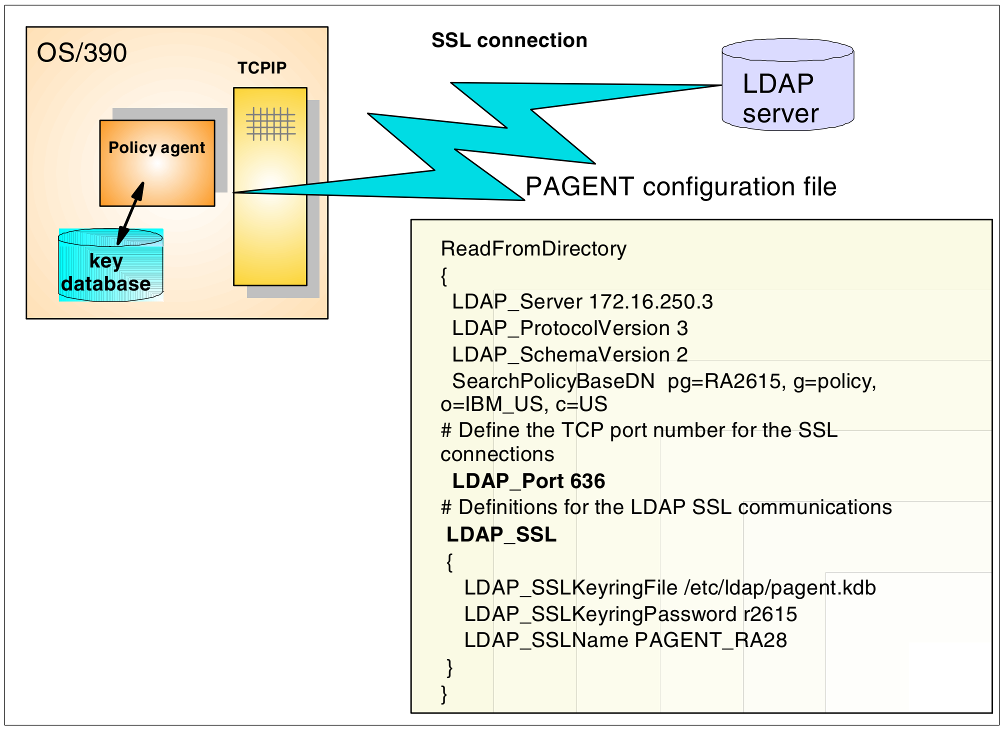
5.10.3 Policy agent and security

It is outside the scope of this book to explore in depth all the facets of policy agent, so we will limit ourselves to those policy agent functions that relate to security in an IP network. If you need detailed information on subjects like Differentiated Services, Integrated Services, Sysplex Distributor traffic balancing, performance monitoring, and how policy agent provides these services, please consult *IBM Communications Server for OS/390 IP Configuration Guide*, SC31-8725, *IBM Communications Server for OS/390 IP Configuration Reference*, SC31-8726, and *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228.

In the subsequent discussions we turn to the subject matter of this book and examine security as provided by the filtering in policy agent and its partner, Traffic Regulation Management Daemon (TRMD).

5.10.3.1 SSL with LDAP and policy agent

In reviewing the features of CS for OS/390 V2R10 IP, we pointed out that an SSL connection between policy agent and the LDAP server provides for secure communications when the LDAP server transmits policies to the policy agent. See Figure 116.



```
ReadFromDirectory
{
  LDAP_Server 172.16.250.3
  LDAP_ProtocolVersion 3
  LDAP_SchemaVersion 2
  SearchPolicyBaseDN  pg=RA2615, g=policy,
o=IBM_US, c=US
# Define the TCP port number for the SSL
connections
  LDAP_Port 636
# Definitions for the LDAP SSL communications
 LDAP_SSL
 {
    LDAP_SSLKeyringFile /etc/ldap/pagent.kdb
    LDAP_SSLKeyringPassword r2615
    LDAP_SSLName PAGENT_RA28
 }
}
```

Figure 116. SSL connection to LDAP server (V2R10)

Note the simple coding in the CS for OS/390 IP policy agent (PAGENT) configuration file that defines this secured sockets connection:

```
LDAP_Port 636
# Definitions for the LDAP SSL communications
 LDAP_SSL
 {
    LDAP_SSLKeyringFile /etc/ldap/pagent.kdb
    LDAP_SSLKeyringPassword r2615
    LDAP_SSLName PAGENT_RA28
 }
```

However, security for policy agent can take on a much broader meaning than providing an SSL connection between the agent and the LDAP server.

5.10.3.2 Policy agent and security filtering

Although service policies are often thought of in terms of performance, they may also be used to enforce certain types of security, specifically to enforce traffic filtering.

Security filtering is an often overlooked feature of policy agent and the Traffic Regulation Management Daemon that was introduced in CS for OS/390 V2R10 IP. One frequently assumes that filtering technologies which limit or block connections reside only in firewall code, when, in fact, other servers may provide screening and filtering techniques of their own. For example, you have already read in earlier discussions that there are access controls introduced with OS/390 V2R10 IP; some of these function essentially as screening mechanisms, preventing certain types of traffic flow to and from the stack. Policy agent and Traffic Regulation Management can play a role here as well.

Note: Policy agent vs. firewall

Firewall filtering can be used to block or limit traffic that enters, exits, or traverses the node on which it is defined. Policy agent filtering affects only traffic whose endpoint is on the CS for OS/390 IP node. That is, forwarded traffic is generally not affected by these filters. Bear in mind that the Sysplex Distribution function views all participating z/OS nodes as one logical node. Therefore, the Sysplex Distributor (SD) function, in which the SD may choose to route traffic to specific distribution targets based upon an `OutboundInterface` parameter, can route packets whose endpoint is not the CS for OS/390 IP.

Furthermore, the packet blocking in policy agent should not be viewed as a replacement for the robust filtering available in the OS/390 Firewall Technologies. Policy agent filtering is fast and easy to implement. It could easily represent one of the first phases of a security implementation, to be augmented later by firewall filtering and IPSec.

What determines if a policy is enforced by the stack or not? We have a simple *if-then* condition testing. If the policy rule is true then the actions specified are applied. Two things determine rule truth:

1. Time specification determines when the rule is active
2. Traffic filters (conditions) determine the description of mapped packets

The actions may provide any of the following functions:

1. Differentiated Services functions (defined as `Scope=DataTraffic`)
   - TCP connection limits
   - Maximum and minimum TCP connection rates, TCP maximum delay
   - Committed access bandwidth (mean rate and peak rate) control/enforcement
   - Type of Service (TOS) byte (also known as DS field - 6 bit value) setting, and mapping to S/390 Queued Direct I/O (QDIO) device priorities and to VLAN user priority
2. Integrated Services (RSVP) functions (defined as `Scope=RSVP`)
   - Token bucket mean rate (r) limits
   - Token bucket depth (b) limits
   - Peak rate (p)
   - Minimum policed unit (m)
   - Maximum packet size (M)

   Note: RSVP policies in policy agent can be used to limit only the values requested for (r) and (b), as well as limiting the total number of RSVP reserved flows. The RSVP daemon enforces these policies and not the IP stack. The remaining three values, (p), (m), and (M) are set by the QoS-aware application. For more information on RSVP, see RFC 2205.
3. Sysplex Distributor target stack distribution
   - Outbound interface designation
   - PolicyPerfMonitorForSDR
4. Traffic Regulation Management actions (defined as `Scope=TR`)
   - Statistics
   - Log
   - Limit

In this redbook we are focusing on security. Therefore, we home in on connection limits within Differentiated Services and on TR policy in the policy agent. The actions revolving around RSVP and Sysplex Distributor functions are relegated to the background for our purposes.

How are policies and actions defined? If they are stored in the LDAP server, you may be using a Version 1, Version 2, or Version 3 LDAP schema to define them. The PAGENT configuration file must indicate which schema should be retrieved in a statement called ReadFromDirectory. *Version 1* applies to policy definitions using the schema that was available prior to CS for OS/390 V2R10 IP, *Version 2* refers to the schema introduced with OS/390 V2R10 IP, *Version 3* refers to the schema introduced with z/OS V1R2. (Version 1 remains valid with OS/390 V2R10 IP for migration and compatibility purposes; Versions 1 and 2 remain valid with z/OS V1R2.)

If a flat file is the repository for policy and action definition, a Version 1 and/or a Version 2 format is available to you depending on whether you are working with a pre-OS/390 V2R10 system or not. However only a subset of the full standards-based Version 2 schema is supported in the flat file configuration, whereas LDAP supports the entire set. There is no Version 3 schema for flat file definitions. You may combine policies defined in the LDAP with policies defined in a flat file. CS for OS/390 V2R10 IP also introduced the TRM policies within the Version 2 flat file syntax to install TR policies into the stack and to enable logging and polling by the Traffic Regulation Management daemon. However, z/OS V1R2 requires the Policy Core LDAP Schema (PCLS), also known as the Version 3 policy schema, for PAGENT if new TRM and IDS functionality is to be exploited. (The V1 and V2 flat files are still supported for previous capabilities.) The LDAP Version 3 schema is, as of the publication timeframe of this redbook, on track to become a standard. It has been extended to include z/OS-specific QoS and Intrusion Detection Services schemas.

Regardless of whether you use LDAP as your policy repository or a flat file, two sets of definitions are created to define the policies and their related actions. To simplify this discussion the following explanations of describing policies and actions describes only the flat file semantics. (Should you need more detail on the LDAP schema, please refer to *IBM Communications Server for OS/390 IP Configuration Reference*, SC31-8726 and *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228.)

If you are defining policies in a flat file with the Version 1 semantics, a *ServicePolicyRules* statement defining particulars like IP address(es), protocols, ports, direction of traffic flow, and time frame is mapped to a *ServiceCategory* statement which defines the type of service to give, including connection limits. If you are using Version 2 semantics, a *PolicyRule* statement is mapped to a *PolicyAction* statement. See Table 14.

Table 14. Policy agent terminology

	Policy Agent Definition Version 1	**Policy Agent Definition Version 2**
Defining Rules	ServicePolicyRules	PolicyRule
Defining Actions	ServiceCategories	PolicyAction

Note in Figure 117 how the Rule cross-references the Action or Category.

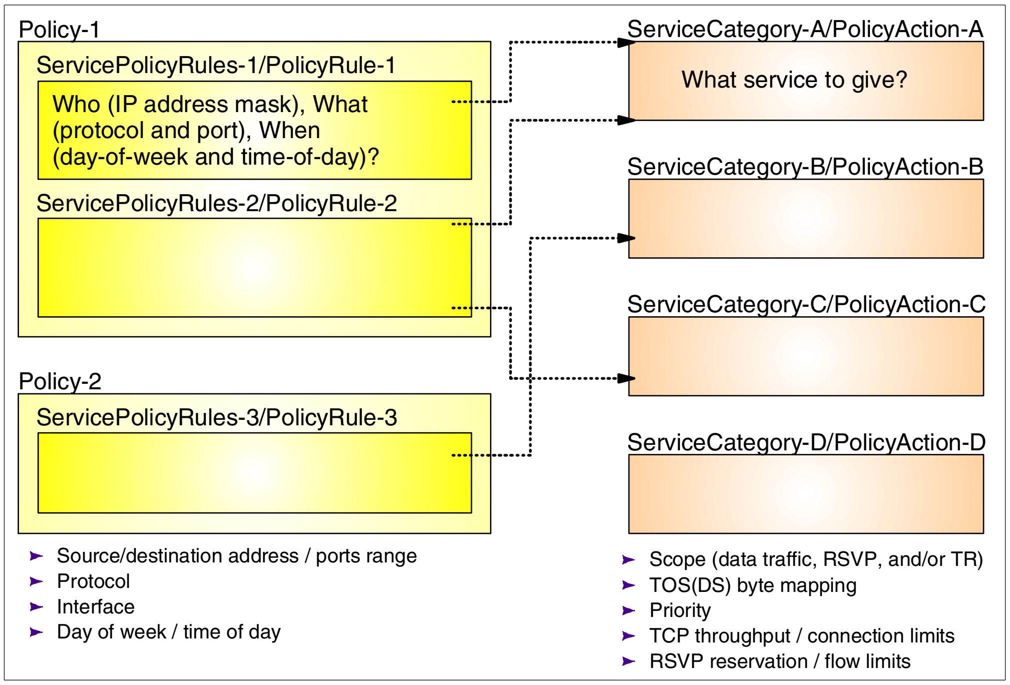

Policy-1
ServicePolicyRules-1/PolicyRule-1
Who (IP address mask), What (protocol and port), When (day-of-week and time-of-day)?
ServicePolicyRules-2/PolicyRule-2
Policy-2
ServicePolicyRules-3/PolicyRule-3
➤ Source/destination address / ports range
➤ Protocol
➤ Interface
➤ Day of week / time of day
ServiceCategory-A/PolicyAction-A
What service to give?
ServiceCategory-B/PolicyAction-B
ServiceCategory-C/PolicyAction-C
ServiceCategory-D/PolicyAction-D
➤ Scope (data traffic, RSVP, and/or TR)
➤ TOS(DS) byte mapping
➤ Priority
➤ TCP throughput / connection limits
➤ RSVP reservation / flow limits

Figure 117. Mapping of rules and actions to each other

Figure 117 on page 214 depicts this relationship between the rule (the "if condition") and the action. This relationship is valid regardless of the definition type used (i.e., Version 1 or Version 2). Table 15 and Table 16 on page 215 illustrate the types of parameters that can be specified as filter rules and actions in PAGENT through CS for OS/390 V2R10 IP:

Table 15. Policy rules in OS/390 V2R10 IP: V1 ServicePolicyRules and V2 PolicyRule

	Scope			
Rule	**DS 1**	**RSVP**	**TR 2**	**SD 3**
PolicyScope (DataTraffic, RSVP, Both)	V1	V1		
Direction (Incoming, Outgoing, Both)	V1	V1		
InboundInterface	V2	V2		V2
OutboundInterface	V2	V2		
Permission (Allowed, Blocked)	V1	V1		
ProtocolNumber	V1	V1		
ProtocolNumberRange	V2	V2		V2
Interface	V1	V1		
SourceAddressRange	V1, V2	V1, V2		V2
DestinationAddressRange	V1, V2	V1, V2		V2
SourcePortRange	V1, V2	V1, V2		V2
DestinationPortRange	V1, V2	V1, V2	V2	V2
DaysOfWeekMask	V1	V1		

	Scope			
Rule	**DS 1**	**RSVP**	**TR 2**	**SD 3**
DayOfWeekMask	V2	V2	V2	V2
TimeOfDayRange	V1, V2	V1, V2	V2	V2
MonthOfYearMask, DayOfMonthMask	V2	V2	V2	V2
ConditionTimeRange	V2	V2	V2	V2
ServiceReference	V1	V1		
PolicyActionReference	V2	V2	V2	V2
PolicyRulePriority	V2	V2	V2	V2
ApplicationName, ApplicationData	V2	V2		V2
Note: 1 Scope=DS (Differentiated Services) defines the policies for DataTraffic. 2 With the TR column, we are pointing out that a rule with the Version 2 parameters indicated may reference an action with Scope=TR. 3 Likewise, with the SD column, which signifies that a sysplex distributor policy may be defined, we are indicating that a rule may be built with these parameters in order to screen for the type of traffic that should be distributed across the sysplex. In fact, there is no "Scope=SD" to define such a rule.				

Notice in Table 15 how many of the rules apply to traffic regulation (TR), which we discuss later in this redbook.

Table 16. Policy actions in OS/390 V2R10 IP: V1 ServiceCategories and V2 PolicyAction

	Scope			
Action	**DS**	**RSVP**	**TR**	**SD**
PolicyScope (DataTraffic, RSVP, TR, Both)	V2	V2	V2	V2
MaxRate, MinRate, MaxDelay	V1, V2	V1, V2		
Interface	V1	V1		
OutBoundInterface				V2 1
OutgoingTOS	V1, V2	V1, V2		
MaxConnections	V1, V2	V1, V2		
DaysOfWeekMask	V1	V1		
TimeOfDayRange	V1	V1		
FlowServiceType (ControlledLoad, Guaranteed)		V1, V2		
MaxRatePerFlow		V1, V2		
MaxTokenBucketPerFlow		V1, V2		
MaxFlows		V1, V2		
Permission (Allowed, Blocked)	V2	V2		
DiffServInProfileRate	V2			
DiffServInProfilePeakRate	V2			

	Scope			
Action	DS	RSVP	TR	SD
DiffServInProfileTokenBucket	V2			
DiffServInProfileMaxPacketSize	V2			
DiffServExcessTrafficTreatment (Drop, BestEffort)	V2			
DiffServOutProfileTransmittedTOSByte	V2			
TypeActions (Statistics, Log, Limit)			V2	
TotalConnections			V2	
Percentage			V2	
TimeInterval			V2	
Logging Level			V2	

Note:
1 The parameter OutBoundInterface may be used strictly to define sysplex distributor interfaces. It is used by SD to find a set or a subset of targets to which incoming requests should be forwarded. The network performance data of the QoS policy at the target nodes, and the performance of the host/application as measured by WLM, determines which is the best OutBoundInterface over which to forward the packets.

As you compare the Version 1 parameters with the Version 2 parameters, notice how some of them have shifted from the rule to the action in Version 2. For example, PolicyScope and Permission have both moved to the action definition from their original rule location in Version 1. Please understand as well that this table does not reflect the parameters for rules in z/OS V1R2.

The TR column in Table 16 on page 215 is used for a different purpose from that in Table 15 on page 214. Here we are indicating the parameters that are strictly for the use of TR policies.

There is no policy scope parameter for Sysplex Distributor (SD) actions, because SD policy is a subset of “Scope=DS”, that is, Differentiated Services or QoS policy. However, we list an SD column here because a rule may be built with this parameter in order to select traffic for the Sysplex Distribution function. Although the SD column in this chart refers to the Sysplex Distributor node itself, in fact, TR policies coded at the SD target nodes will influence the selection of target stacks by the distributor node. Read more about this in *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 1: Configuration and Routing*, SG24-5227.

5.10.3.3 Examples of security filtering with policy agent
Assume you want to limit the number of inbound CICS connections from remote port 2001 in order to ensure that other TCP/IP processes, for example, bulk processing of FTP, are able to be handled efficiently during the nighttime. The definitions accomplishing this policy appear in Figure 118 on page 217.

```
ServicePolicyRules CICS-nighttime
{
  Direction            Inbound
  Protocol             TCP
  DestinationPortRange 2001       #CICS
  DaysOfWeek           0111110    #Mon-Fri
  TimeOfDay            18:30-24:00 0-6:30
  ServiceReference     cics-night
}

ServiceCategories cics-night
{
  MaxRate      1024     #1024 Kpbs (1Mbps)
  MaxConnections 100
}
```

Figure 118. Limiting CICS connections: policy agent Version 1 definition

Observe how the parameter `ServiceReference` in the ServicePolicyRules statement identifies the ServiceCategories with which the specific ServicePolicyRule should be associated. For example, the `ServiceReference` inside ServicePolicyRules `CICS-nighttime` points to ServiceCategories `cics-night`. An interesting point about this example is that both actions - MaxRate and MaxConnections - are enforced outbound. When the one hundred and first connection request (SYN) arrives inbound at this OS/390 system, the IP stack will recognize that it is not allowed to respond *`outbound`* to the request and it will reject it.

QoS rules vs. blocking rules

Rules that intend to set QoS, RSVP, or SD policies, including connection limits, apply only to traffic flowing outbound. Policy agent does not set QoS or RSVP values for inbound traffic. The one exception to this statement is when the SD node views itself together with the target stacks logically as one entity and will examine inbound traffic that is *in transit* to a distributed stack under the aegis of the sysplex distribution function.

Rules that intend to *block* traffic apply to both inbound and outbound traffic, whether or not the traffic originates at CS for OS/390 IP.

Note in Figure 118 on page 217 that we have used a Version 1 policy definition. It is imperative to understand that the Version 1 policy agent terms “source” and “destination” refer respectively to “local” and “remote” from the viewpoint of CS for OS/390 IP. Version 2 terminology applies a significantly different meaning to the same terms.

Version 1 and Version 2: “source” vs. “destination”

You have just read that the Version 1 policy agent terms "source" and "destination" refer to "local" and "remote." Version 2 terminology applies a significantly different meaning to the same terms.

With Version 2, the “source” indicates the source of the data flow with “destination” signifying the target of the data traffic. So, in some cases the source would be CS for OS/390 IP and in others the source would be the remote host.

Suppose you now want to limit, but not block, concurrent FTP connections from a specific network to a mere fifty employing Version 2 syntax. See Figure 119 on page 218. In other words, you want to define a QoS policy for TCP traffic that does not completely filter out (“block”) FTP traffic but just restricts it.

```
PolicyAction ftpaction
  {
  MaxConnections 50               # Limit FTP concurrent connections to 50.
  MaxRate 400                     # Limit FTP connection throughput to 400 Kbps.
  OutgoingTOS 01000000            # the TOS value of outgoing FTP packets.
  }

PolicyRule ftprule
  {
  ProtocolNumberRange   6
  SourcePortRange       20 21
  DestinationAddressRange    172.16.232.1 172.16.252.254
  policyactionreference ftpaction
  }
```

Figure 119. Limiting inbound FTP connections: policy agent Version 2 definition

The PolicyAction named `ftpaction` allows a maximum of fifty connections across *both* FTP ports while regulating throughput to 400 kbps. The PolicyRule named `ftprule` references this PolicyAction and is invoked when packets identified as FTP (TCP protocol #6) need to flow from port 21 (the well-known FTP control port) or from port 20 (the well-known FTP data port) to remote addresses ranging from 172.16.232.1 through 172.16.252.254.

Notice that the data connection, typically initiated by the FTP server on port 20, will not be established if a remote connection with port 21 has been rejected due to the connection limit. Likewise, if fifty clients establish control connections with port 21, there are consequently no connections remaining to service port 20 requests. Therefore MaxConnections of fifty when indicating both FTP ports effectively limits the remote network to twenty-five users doing productive work. Note that, if there were a second FTP server at this node running on alternate ports, for example, 2020 and 2021, connections to this FTP server would not be restricted by this PolicyRule.

The filter rule functions even in a *firewall-friendly* implementation which allows the remote FTP client to initiate the connection to a data port other than the well-known port 20. (According to the RFC 1579 for firewall-friendly FTP, an ephemeral port is used for the server-side data connection.) Although there is no limitation on the inbound use of such an ephemeral port, the restrictions placed on responses coming from the source (and control) port 21 to the destination network limit the total number of FTP connections.

Rule of thumb: QoS policy

Since QoS policies are enforced only for outbound traffic and not for inbound traffic, a good rule of thumb is to indicate the SourcePortRange and the DestinationAddressRange in the PolicyRule.

You may question the specification of a port *range* for FTP connections, given the fact that the MaxConnections parameter applies to the connections *across* the port range. Therefore you might consider coding the policy rule as follows in order to simplify the understanding of the true limits put in place:

```
PolicyAction ftpaction
  {
  MaxConnections 50                # Limit FTP concurrent connections to 50.
  MaxRate 400                      # Limit FTP connection throughput to 400 Kbps.
  OutgoingTOS 01000000             # the TOS value of outgoing FTP packets.
  }

PolicyRule ftprule
  {
  ProtocolNumberRange   6
  SourcePortRange       21 21
  DestinationAddressRange    172.16.232.1 172.16.252.254
  policyactionreference ftpaction
  }
```

Figure 120. Limiting inbound FTP connections to port 21: policy agent Version 2 definition

In Figure 120 on page 219 we have specified the SourcePortRange as "21 21," thus making it quite clear that only fifty concurrent FTP users will be allowed.

Can you completely stop certain types of traffic using policy agent? The following set of definitions illustrates how to accomplish this for inbound telnet traffic from network 192.168.5.0.

```
PolicyAction telnet-block
  {
  Permission Blocked               # Do not permit inbound telnet
  }

PolicyRule telnetin-block23
  {
  DestinationPortRange            23 23
  SourceAddressRange              192.168.5.1     192.168.5.254
  policyactionreference           telnet-block
  }
```

Figure 121. Blocking inbound Telnet traffic: policy agent Version 2 definition

This definition causes the stack to discard all packets from subnet 192.168.5.0/24 for Telnet at Port 23. Yet there might be other telnet servers at other ports available on the IP stack, including a UNIX telnet server. Perhaps it is not important to block access to those telnet servers because you have other security mechanisms in place for them. However, if it is critical to inhibit all attempts to reach any telnet server at OS/390, you would need to include a PolicyRule for each of them.

Rule of thumb: blocking traffic

Recall from earlier notes that QoS policies are applied to traffic flowing outbound from CS for OS/390 IP. However, blocking policies are applied to both inbound and outbound traffic flow. Generally speaking, if we want to block inbound traffic to CS for OS/390 V2R10 IP and we are using Version 2 policy syntax, we would name a CS for OS/390 IP “destination port” with a “source address” of the remote host or (sub)network. Likewise, if we want to block outbound traffic to a remote destination, we would define a “destination port” at the remote site and a CS for OS/390 IP “source address.” However, a distinction is made as to how policy agent applies this general rule:

- For UDP or RAW socket flows, the origination of the request is irrelevant. The direction of the traffic flow determines whether a particular blocking rule applies or not. No other factors are considered.
- For actions specifying a Permission of *blocked*, TCP traffic is handled differently depending on whether the connection request is inbound or outbound.
  - When an inbound request is received, an inbound rule is checked to see if the SYN should be accepted. If it is, then the outbound rule is checked to see if the connection is allowed. If both the inbound and outbound rules indicate that it is all right to accept the connection, then a `tcb` (TCP connection control block) is built. Any other QOS rules, for example, TOS settings or similar, may now be applied to the outbound connection. If the inbound rule permits the connection but the outbound rule does not, the tcb is not built and the connection request is rejected.
  - If an outbound TCP SYN request is generated and there is no outbound blocking rule in effect, the `tcb` is created and any outbound QoS rules are applied. Possible inbound blocking rules are ignored. When the SYN/ACK arrives back at the CS for OS/390 IP server, a tcb with an assigned outbound QoS already exists and there is no further checking to see if an inbound blocking rule is in effect.

Instead of limiting or preventing inbound connections, your security may require eliminating the capability of TSO users to become Telnet clients and telnet to remote destinations. Here is our last example of a possible security scenario that has implemented policy agent.

```
PolicyAction telnet-block
  {
  Permission Blocked              # Do not permit outbound telnet
  }

PolicyRule telnetout-block23
  {
  DestinationPortRange          23 23
  SourceAddressRange            192.168.8.1        # SourceVipa Address
  policyactionreference         telnet-block
  }
```

Figure 122. Blocking outbound Telnet traffic: policy agent Version 2 definition

This example assumes that you know the source address of outbound packets. The use of SOURCEVIPA simplifies the designation of the source address for packets flowing outbound. With this policy no TSO client may telnet outbound to a remote telnet server. Obviously if you are not using SOURCEVIPA and if you have multiple stack interfaces, this complicates the coding of the SourceAddressRange. However, you might find that all your stack interfaces can be supernetted, in which case you may specify a supernet that encompasses the address range. Alternatively, if you use the LDAP schema you may specify a set of addresses in the fashion of “address1 OR address2 OR address3” or even specify all local addresses.

A warning about SourceAddressRange is useful here. If, in our example in Figure 122 on page 221, you had specified a SourceAddressRange of 0.0.0.0, you would have blocked not only outbound telnet traffic from the IP stack, but also all inbound workstation telnet traffic!

Policy agent is not intended to be a foolproof method of restricting access to the system. For example, you can see in Figure 122 that a remote Telnet server listening on a port other than the well-known port 23 may still receive connection requests from a CS for OS/390 IP telnet client.

Policy agent as security

We cannot emphasize this enough: To put this description of policy agent as a security mechanism in perspective, PAGENT should be considered as just one more "line of defense." You may choose to use it as such behind an external firewall, or local firewall filtering technology, or even CS for OS/390 V2R10 IP access control capabilities described elsewhere in this book.

5.10.4 Displaying active policies

At V2R8 the `NETSTAT SLAP` command provides information about the service policies as you see in Figure 123.

```
EZZ2350I MVS TCP/IP NETSTAT CS V2R8       TCPIP NAME: TCPIPA       16:41:39
EZZ2840I Policy:    RSVPappl
EZZ2841I   Profile:   rule39a3
EZZ2842I     Scope:            RSVP              Direction:       Both
EZZ2843I     Permission:       Allowed           Protocol:        0
EZZ2844I     LocalInterface:   0.0.0.0           Position:        0000000000
EZZ2845I     SourceIpFrom:     0.0.0.0           SourceIpTo:      0.0.0.0
EZZ2846I     SourcePortFrom:   000000            SourcePortTo:    000000
EZZ2847I     DestIpFrom:       0.0.0.0           DestIpTo:        0.0.0.0
EZZ2848I     DestPortFrom:     008000            DestPortTo:      008001
EZZ2849I     ServiceClass:     srv39a3
EZZ2850I       Scope:          RSVP              OutgoingTOS:     C0
EZZ2851I       Priority:       0                 Interface:       0.0.0.0
EZZ2852I       MaxRate:        0000000500        MinRate:         0000000300
EZZ2853I       MaxDelay:       0000000000        MaxConn:         0000000000
EZZ2854I     Performance Information:
EZZ2855I       FirstActTime:   15:07:04          LastMapTime:     00:00:00
EZZ2856I       TotalBytesIn:   0000000000        TotalBytesOut:   0000000000
EZZ2857I       BytesInDiscard: 0000000000        BytesOutDiscard: 0000000000
EZZ2858I       TotalInPackets: 0000000000        TotalOutPackets: 0000000000
EZZ2859I       ActConnMap:     0000000000        MaxConnLimit:    0000000000
EZZ2860I       AcceptConn:     0000000000        DeniedConn:      0000000000
```

Figure 123. Output of NETSTAT SLAP

At V2R10 the `NETSTAT SLAP (onetstat -j)` command has been altered to display only the policy statistics. See Figure 124.

```
MVS TCP/IP onetstat CS V2R10       TCPIP Name: TCPIPA          1
PolicyRuleName:  ftprule1
  FirstActTime:    17:07:24      LastMapTime:      17:16:24
  TotalBytesIn:    0000000992    TotalBytesOut:    0000001950
  BytesInDiscard:  0000000000    BytesOutDiscard:  0000000000
  TotalInPackets:  0000000040    TotalOutPackets:  0000000045
  ActConnMap:      0000000000    MaxConnLimit:     0000000002
  AcceptConn:      0000000004    DeniedConn:       0000000067
  OutBytesInProf:  0000000000
PolicyRuleName:  ftprule
  FirstActTime:    17:07:24      LastMapTime:      17:26:44
  TotalBytesIn:    0000000000    TotalBytesOut:    0000000000
  BytesInDiscard:  0000000000    BytesOutDiscard:  0000000000
  TotalInPackets:  0000000000    TotalOutPackets:  0000000000
  ActConnMap:      0000000000    MaxConnLimit:     0000000000
  AcceptConn:      0000001084    DeniedConn:       0000000000
  OutBytesInProf:  0000000000
```

Figure 124. (o)netstat -j at V2R10

A V2R10 UNIX command, `pasearch`, displays policy information with more detail than was previously available.

```
MVS TCP/IP pasearch CS V2R10              TCP/IP Image:      TCPIPA
  Date:                 08/08/2000        Time:  19:07:50

policyRule:             ftprule
  Version:              2                 Status:            Active
  Distinguish Name:     pr=ftprule,pg=TCPIPC,g=policy,o=IBM_US,c=US
  Group Distinguish Nm: pg=TCPIPC,g=policy,o=IBM_US,c=US
  Priority:             2                 Sequence Actions:  Mandatory
  ConditionListType:    CNF               No. Policy Action: 1
  policyAction:         ftpaction
   ActionType:          QOS               Action Sequence:   255
  Time Periods:
   Day of Month Mask:   1111111111111111111111111111111
   Month of Year Mask:  111111111111
   Day of Week Mask:    0111110  (Sunday - Saturday)
   Start Date Time:     Fri Jan  1 00:00:00 1999
   Start Date Time UTC: Fri Jan  1 05:00:00 1999
   End Date Time:       Tue Jan  1 00:00:00 2002
   End Date Time UTC:   Tue Jan  1 05:00:00 2002
   From TimeOfDay:      08:00             To TimeOfDay:      22:00
   From TimeOfDay UTC:  12:00             To TimeOfDay UTC:  02:00
   TimeZone:            Local
  Condition Summary:                      Negative Indicator:  OFF
   RouteCondition:
    InInterface:        0.0.0.0           OutInterface:      0.0.0.0
   HostCondition:
    SourceIpFrom:       0.0.0.0           SourceIpTo:        0.0.0.0
    DestIpFrom:         0.0.0.0           DestIpTo:          0.0.0.0
   ApplicationCondition:
    ProtocolNumFrom:    6                 ProtocolNumTo:     6
    SourcePortFrom:     20                SourcePortTo:      21
    DestPortFrom:       0                 DestPortTo:        0
    ApplicationName:
    ApplicationData:
```

Figure 125. pasearch command output (part 1)

The V2R10 UNIX command, `pasearch -f ftprule`, generated the output in Figure 125. `pasearch -f` requests a display of any policy rules that match the indicated name. Variations of this command display policy rules and policy actions.

The default for `pasearch` is `pasearch -e`, meaning that all policy entries - both the PolicyAction and the associated PolicyRule - will be returned. The command will execute against policy rules coded using any syntax version. See the continuation of the command in Figure 126.

```
Qos Action:           ftpaction
    Version:            2                Status:            Active
    Distinguish Name:   pa=ftpaction,pg=TCPIPC,g=policy,o=IBM_US,c=US
    Scope:              DataTraffic      OutgoingTOS:       01000000
    Permission:         Allowed
    MaxRate:            400              MinRate:           0
    MaxDelay:           0                MaxConn:           5
    Routing Interfaces: 4
      Interface Number: 1                Interface:         172.16.233.5
      Interface Number: 2                Interface:         172.16.233.29
      Interface Number: 3                Interface:         172.16.233.40
      Interface Number: 4                Interface:         0.0.0.0
    RSVP Attributes
     ServiceType:       0                MaxRatePerFlow:    0
     MaxTokBuckPerFlw:  0                MaxFlows:          0
    DiffServ Attributes
     InProfRate:        0                InProfPeakRate:    0
     InProfTokBuck:     0                InProfMaxPackSz:   0
     OutProfXmtTOSByte: 00000000         ExcessTrafficTr:   BestEffort
    TR Attributes
     TotalConnections:  0                LoggingLevel:      0
     Percentage:        0                TimeInterval:      0
     TypeActions:       0
```

Figure 126. pasearch command output (part 2)

At z/OS V1R2, the `pasearch` output changes to reflect the new functions. Existing TR policy defined in the policy agent configuration file will continue to work as it does in V2R10. However, `pasearch` output will display it as IDS policy rather than QOS policy.

5.10.4.1 Security for policy displays

The `pasearch` command normally requires superuser authority. However, at z/OS V1R2, a new RACF class profile limits the users who may see all aspects of policy when they execute the `pasearch` command. It is possible to set the SERVAUTH class profile to distinguish among users allowed to view only certain policies. For example, certain users may be authorized to view only QoS policies, while others view only IDS policies. Sample profiles for SERVAUTH are:

- EZB.PAGENT.systemname.tcpimage.QoS
- EZB.PAGENT.systemname.tcpimage.IDS
- EZB.PAGENT.systemname.tcpimage.*

5.10.5 PAGENT logging

Prior to V2R10, policy agent logs to an HFS file identified in the environment variable PAGENT_LOG_FILE. At V2R10 and higher the implementer has a choice: PAGENT may log in an HFS or in the syslogd log file.

5.11 Traffic Regulation Management (TRM) security

Flooding attacks in the internet have become prevalent in recent years. A network might be flooded with ICMP, UDP, or TCP packets. In an attempt to deal with them, many companies simply disable certain services temporarily. Other companies deploy a multi-tiered line of defense, with tiers of routers limiting traffic rates to upstream targets for certain packet types or preventing directed broadcasts altogether. Another way to deal with the attacks is to discard suspected packets once they exceed what is deemed a legitimate threshold. Traffic Regulation and Management (TRM) support, first introduced with CS for OS/390 V2R10 IP, employs the former approach of discarding suspected packets once they exceed a certain threshold. TRM is a function of the IP stack, but it receives its threshold information from TR policies that you code in Policy Agent (PAGENT).

Still in its infancy, TRM in OS/390 V2R10 tackles only Traffic Regulation for TCP connections. Beginning with z/OS V1R2, TRM is enhanced to include other types of attacks as well as UDP port monitoring. At this stage it is considered to be an OS/390 *Intrusion Detection Services* (IDS). Although it can be difficult to distinguish between a denial-of-service (DoS) attack and valid but unusually heavy traffic, TRM can be used to limit and report on such connections. z/OS V1R2 provides real-time notification at the MVS console of detected intrusions. The TRM function in the IP stack can deny new connections that are over the specified limit. The Traffic Regulation Management daemon (TRMD) polls the stack and logs TR activity and limit violations to syslogd. The IDS functions in the stack can also send messages to the z/OS console, as see in Figure 127.

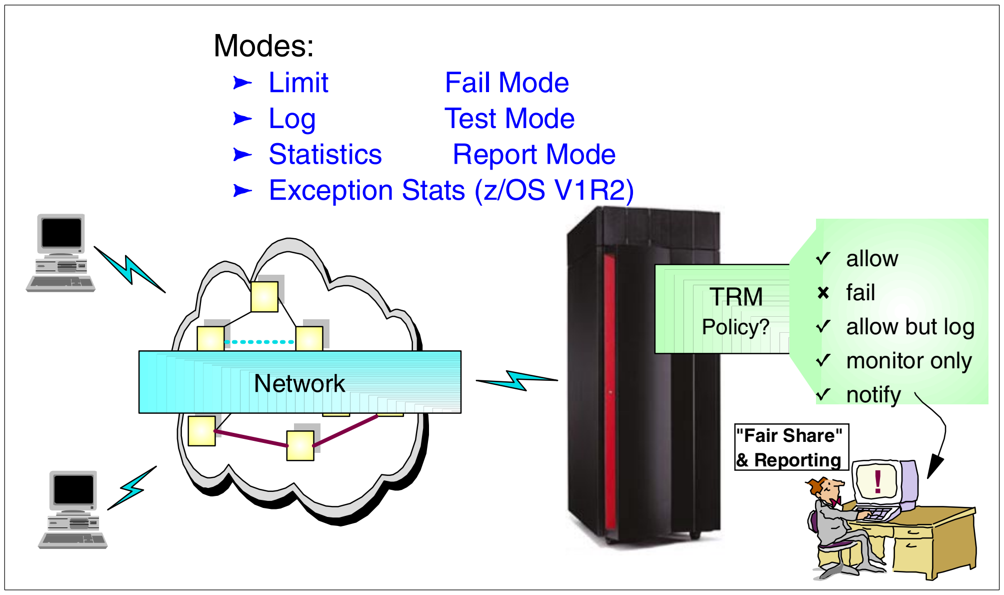


Figure 127. TRM function

TRM offers enhancements to the functions already available in the connection-limiting parameters of the DS policies of policy agent. TR policies permit the regulation of connections to prevent overconsumption by individual clients. Policy agent installs the TR policies in the stack, enabling the IP stack to control the number of TCP connections from specific hosts to a designated port and, at z/OS V1R2, to control the UDP flows. In this sense, TRM appears to overlap with the capabilities already available in the DS policies of policy agent. However, there is an important difference.

Specifically, TR policies for TCP, like the DS connection-limiting policies of policy agent, regulate the number of allowable connections by imposing a hard limit on the permitted number of connections. Unlike DS, TR policies for TCP set restrictions on the *percentage* of available resources that may be consumed for particular connection types given the current state of the CS for OS/390 V2R10 IP stack; in this fashion it helps prevent certain users from monopolizing the network, attempting to ensure that the network resources are *fairly shared*. In addition, the Traffic Regulation Management daemon (TRMD) provides a reporting mechanism to alert the system administrator to overconsumption of resources that may be a sign of a denial-of-service attack.

Figure 128 illustrates the algorithm applied by TRM in order to regulate traffic for TCP. TRM applies a formula to each new connection request based on a percentage of remaining available connections and the number already held by the requesting client. The algorithm used is biased towards maximizing the number of clients but will allow earlier clients to obtain more connections than later clients.

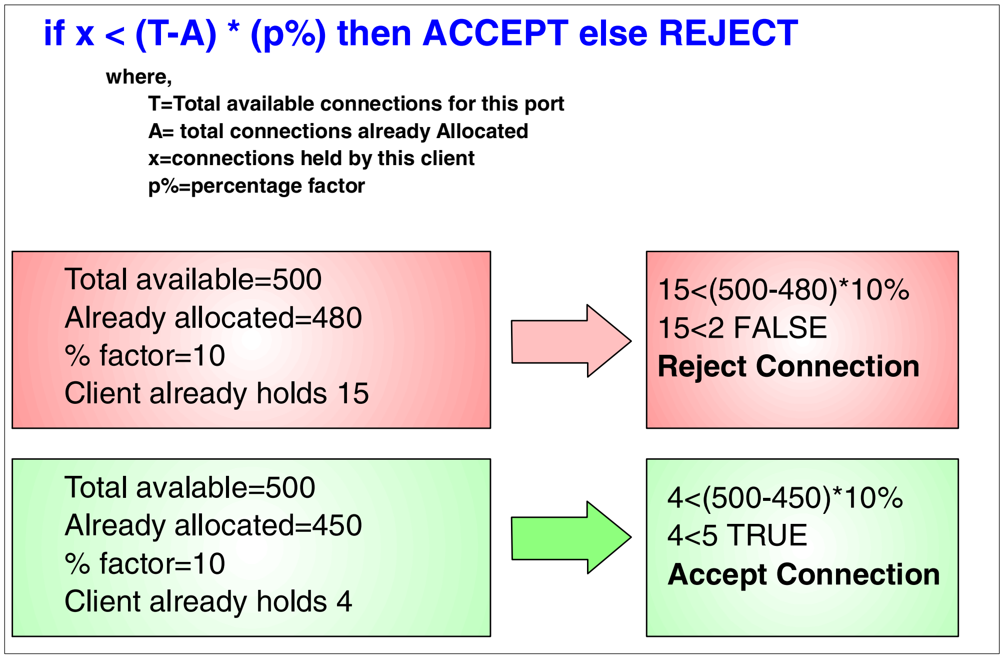


Figure 128. Connection regulation with TRM

In the first example of Figure 128 the assumptions are that 500 connections are available that will be monitored by TRM. 480 of those connections have already been allocated. 20 connections remain to be allocated; of these, 10% (2 connections), may be allocated to the next client making a connection request. However, the client making the request already has 15 connections, a number greater than the 2 he should be permitted. Therefore, his connection request is rejected. However, note that the client does not lose any of his existing connections.

In the second example of Figure 128 there are now only 50 connections remaining (500-450=50). 10% of these, that is, five, could be made available to the next client. The client in this scenario already holds 4 connections; since he is allowed up to 5, this connections request is honored.

TR policies are set in the policy agent and installed in the IP stack, which enforces them, as shown in Figure 129. Both PAGENT and the TCP/IP stack must have been started prior to running TRMD, the process that polls the stack and logs data pertaining to TR policies and any violation thereof.

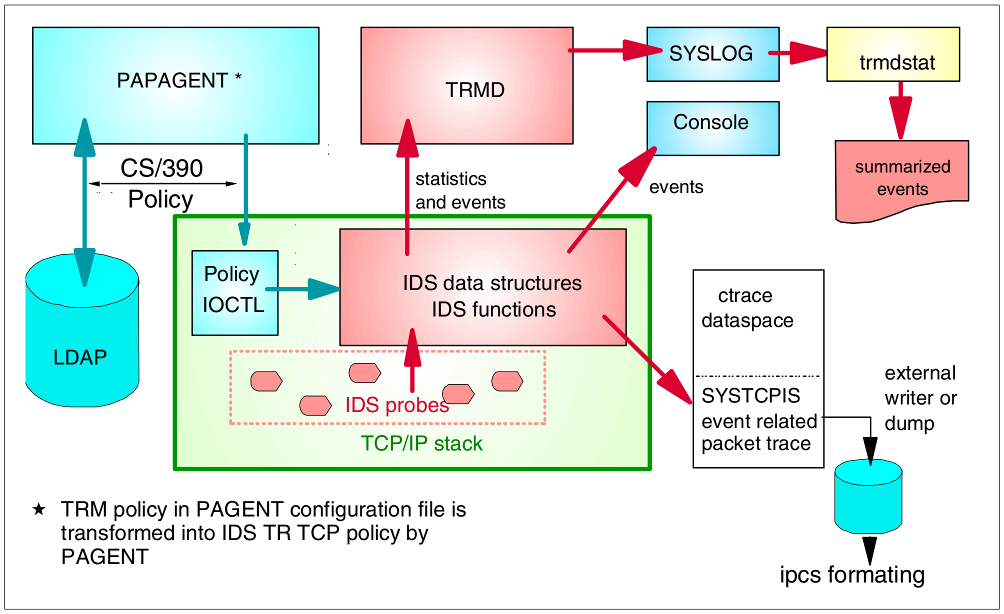
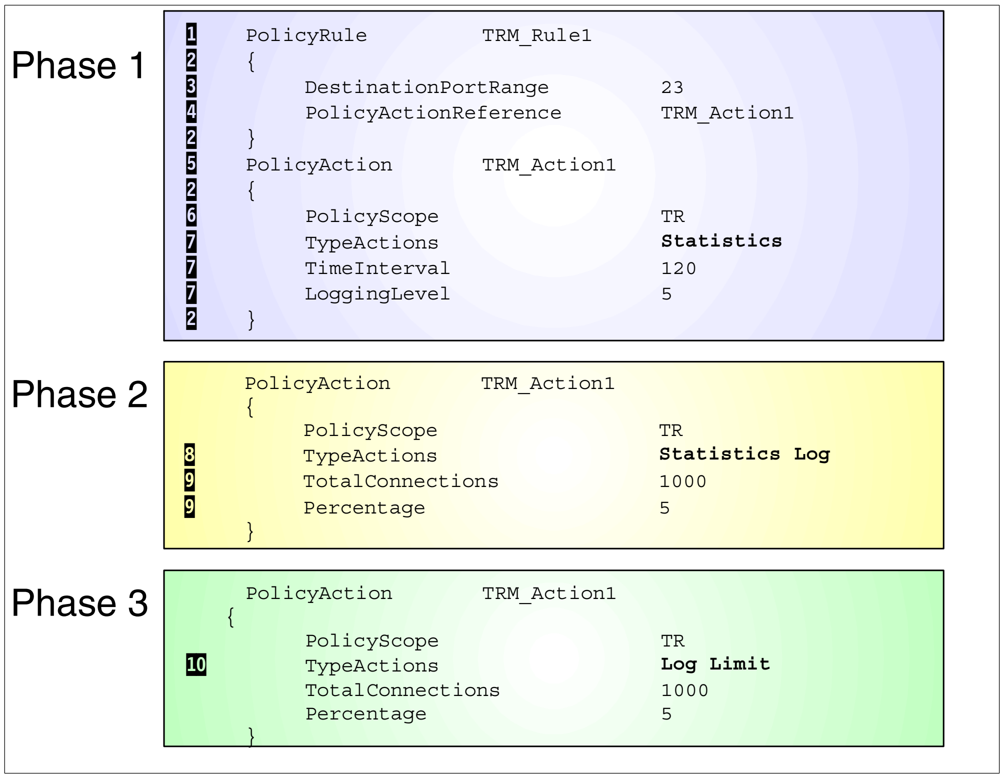


Figure 129. TRM system structure z/OS V1R2

Figure 129 shows the TRM implementation as it exists in z/OS V1R2. The functions introduced at that release are event logging by IDS to the MVS console, and when requested, event tracing of suspicious activity, which can subsequently be formatted with IPCS. TRMD continues to log to syslogd. Recall that the z/OS V1R2 IDS functions, such as monitoring for SYN attacks, scan, and UDP flooding are available only when the policies have been defined in LDAP Version 3 schema. However, older schema types are still supported as are flat file policies for the older TR policies available in OS/390 V2R10 IP.

Figure 130 on page 228 represents a strategy for implementing TRM policies for TCP. Phase 1 shows the definitions to implement statistics collection on the IP traffic throughput. Ideally this will give an idea of what the ideal network traffic performance should look like. Note the PolicyScope of TR and the action to collect and log statistics.

Phase 2 still reflects a PolicyScope of TR. However now we *theoretically* limit the connections to 1000 as long as 5% of the remaining connections are still available to other processes; we log any connections that surpass these limits. The excess connections represent those that would have been rejected had we implemented this policy *in practice*. Phase 2 gives us a chance to test out a policy without actually enforcing it to determine how realistic that policy is.


Phase 1
Phase 2
Phase 3

```
1    PolicyRule          TRM_Rule1
2    {
3        DestinationPortRange          23
4        PolicyActionReference         TRM_Action1
2    }
5    PolicyAction        TRM_Action1
2    {
6        PolicyScope                   TR
7        TypeActions                   Statistics
7        TimeInterval                  120
7        LoggingLevel                  5
2    }
```

```
     PolicyAction        TRM_Action1
     {
         PolicyScope                   TR
8        TypeActions                   Statistics Log
9        TotalConnections              1000
9        Percentage                    5
     }
```

```
     PolicyAction        TRM_Action1
    {
         PolicyScope                   TR
10       TypeActions                   Log Limit
         TotalConnections              1000
         Percentage                    5
     }
```

Figure 130. TR policy definition at CS for OS/390 V2R10 IP

If we have determined after an examination of the log that the PolicyAction specified in Phase 2 is realistic, we then implement that action in Phase 3 by changing the parameter `TypeActions` from `Statistics Log` to `Log Limit`. For a display of the reports produced by trmdstat at CS for OS/390 V2R10 IP, please see Figure 131 on page 229.

Although our examples in Figure 128 on page 226 and Figure 130 on page 228 describe in some detail how to implement TR policy for TCP, similar concepts apply to TR policy for UDP at z/OS V1R2. Whereas with TCP traffic we deal with connections, UDP traffic is measured in *queue length*. A summary report on the results of logging UDP traffic appears later in this document; see Figure 132 on page 231.

You may have observed that there is a policy agent TR parameter called `TotalConnections` (Figure 130) and a policy agent DS parameter called `MaxConnections` (Figure 119 on page 218). Both define a limit to the number of connections. In fact they are unrelated. `TotalConnections` defines the total number of connections allowed on a local port; `MaxConnections` defines the maximum number of connections allowed to a subnet or host. In one case only does `MaxConnections` take precedence over the percentage of `TotalConnections` rule: when the `MaxConnections` is applied to a *remote host* accessing a *specific OS/390 port*, and its value is larger than the number of connections allowed to a single host under a TR policy, the connection request will be granted to the host as long as long as TR is not in a *constrained* state. According to *IBM Communications Server for OS/390 IP Configuration Guide*, SC31-8725, "TR is considered in a constrained state when 90% of the total number of connections (TotalConnections) are already held. After a constrained state is entered, TR remains in a constrained state mode until 12% of the total number of connections are again available."

Keep in mind that a policy with scope of TR may exist independently of a policy with a scope of DS. For example, you may code a TR policy regulating FTP traffic. However, you may have no DS policy defining FTP traffic in any fashion. Likewise, you may code a DS policy for telnet with no TR policy regulating this type of traffic. Nevertheless, you might consider ensuring that certain users, the system programmers or operators for example, may gain access to telnet by establishing a DS policy specifically for telnet traffic from their assigned IP addresses. Recall that a DS policy defined for a host takes precedence over any percentage regulated maximums defined in a TR policy. Note that, even if the DS policy allows the source address additional connections, the `Total Connection` limit will never be exceeded.

Note

Policies with a scope of *TR* may be defined within a flat file only and not within an LDAP server.

5.11.1 TRM reporting

5.11.1.1 CS for OS/390 V2R10 IP

To produce a report based on the log contents, the UNIX `trmdstat` command was introduced in CS for OS/390 V2R10 IP. `trmdstat` invokes a scan of the log indicated on the command (`/tmp/syslogd.lo`g). The sample log report in Figure 131 reflects the output of the report after the Phase 2 policy has been activated.

```
trmdstat -C -D /tmp/syslogd.log
CS for OS/390 trmdstat CS V2R10    Wed Aug  9 15:03:18 2000

Log Time Interval    : Aug  9 14:18:12  - Aug  9 15:01:13
Stack Time Interval  : Aug  9 18:18:11  - Aug  9 19:00:50
TRM Records Scanned  : 28
Port Range           : ALL

Port Number      Connections Refused     IP Address
-----------      -------------------     ----------
No records to display

Port Number      Connections Would Have Been Refused        IP Address
-----------      -----------------------------------        ----------
      23                   9
                                2                           9.24.106.113
                                7                           9.24.106.31

TRMD Started                 : Aug  9 11:43:28
Policy might have changed    : Aug  9 11:44:11
TRMD Ended                   : Aug  9 13:34:17
```

Figure 131. TRM activity at CS for OS/390 V2R10 IP: report generation

The report indicates that nine connections in total would have been refused had the policy been put into effect per the `TypeActions` parameter coded in the Phase 3 implementation of the policy: two from IP address 9.24.106.113 and seven from

IP address 9.24.106.31. The `trmdstat` command is thoroughly documented in *OS/390 IBM Communications Server User's Guide,* SC31-8514.

For further information on implementing policy agent and TRMD, please see *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228, *IBM Communications Server for OS/390 TCP/IP 2000 Update Technical Presentation Guide*, SG24-6162, *IBM Communications Server for OS/390 IP Configuration Guide*, SC31-8725, and *IBM Communications Server for OS/390 IP Configuration Reference*, SC31-8726.

5.11.1.2 Communications Server in z/OS V1R2 IP

With this level of IP, TRM becomes part of the Intrusion Detection Services (IDS) function. In addition to TCP traffic regulation, IDS supports UPD traffic regulation, real-time attack event notification and scan detection. Messages can be sent by the IDS function to the MVS console to indicate that attempts have been made to infiltrate z/OS IP. To take advantage of this and other new z/OS IP features for TRM, it is necessary to define the policy rules and actions in an LDAP server with the Version 3 schema.

Traffic regulation displays with trmdstat at V1R2

```
trmdstat -I /tmp/dablog.log
trmdstat for z/OS CS V1R2          Wed Jan 31 15:51:45 2001

Log Time Interval     : Jan  9 12:16:24  - Jan  9 12:20:54
Stack Time Interval   : Jan  9 16:09:06  - Jan  9 17:20:36
TRM Records Scanned   : 3307
Port Range            : ALL

Traffic Regulation - TCP
-----------------------------------------------
Connections would have been refused :         0
Connections refused                 :         0

Constrained entry logged            :         0
Constrained exit logged             :         0
Constrained entry                   :         1
Constrained exit                    :         1

QOS exceptions logged               :         0
QOS exceptions made                 :         0

Traffic Regulation - UDP
-----------------------------------------------
Constrained entry logged            :         0
Constrained exit logged             :         0
Constrained entry                   :         0
Constrained exit                    :         0
SCAN Detection
-----------------------------------------------
Threshold exceeded                  :         0
Detection delayed                   :         0
Storage constrained entry           :         0
Storage constrained exit            :         0

ATTACK Detection
-----------------------------------------------
Packet would have been discarded    :         0
Packet discarded                    :       593

FLOOD Detection
-----------------------------------------------
Accept queue expanded               :         0
SYN flood start                     :         0
SYN flood end                       :         0

440 ATTACK messages lost at 01/09/2001 16:08:26.49

TRMD Started                : Jan  9 10:53:42
TRMD Ended                  : Jan  9 11:05:14
TRMD Started                : Jan  9 12:16:22
```

Figure 132. TRMDSTAT report at z/OS V1R2

The IDS summary report in Figure 132 illustrates how various types of intrusion attacks would be reported with the z/OS V1R2 functionality. There are also other detail reports available as well, depending on the trmdstat options selected:

- Scan detection reports to reveal if a source host has been attempting to access multiple unique resources (ports or interfaces) over a specified period of time

- Attack reports to notify of malformed packets, IP fragments, attempted use of restricted IP protocols or options, UDP perpetual echoes, ICMP redirect packets, outbound Raw packet flow attempts.
- TCP SYN flooding.

Figure 133 indicates the number of times the UDP port receive queue became constrained, how many times the UDP port receive queue constraint ended, and the number of seconds the UDP port receive queue constrained condition existed. Finally, it shows how many datagrams would have been discarded if the queue size TR policies had been enforced.

```
trmdstat -U /tmp/tstlog.log
trmdstat for Z/OS CS V1R2          Wed Nov  8 09:00:20 2000

Log Time Interval    : Aug 21 08:32:09  - Aug 21 10:32:09
Stack Time Interval  : Aug 21 14:32:09  - Aug 21 16:33:09
TRM Records Scanned  : 71
Port Range           : ALL

                            UDP TR Summary

                               Constrained State          Datagrams
   IP Address      Port   Entered     Exited   Duration   Discarded
----------------- ------ ---------- ---------- ---------- ----------
 05.16.17.18        2001          1          1        100        155
 05.16.17.18        5001          2          2        200        310


                               Constrained State          Datagrams
   IP Address      Port   Entered     Exited   Duration    Would
                                                          have been
                                                          Discarded
----------------- ------ ---------- ---------- ---------- ----------
 05.16.17.18        1001          1          1        100        155
 05.16.17.18        2001          2          2        200        310


TRMD Started                 : Aug 21 10:32:09
```

Figure 133. UDP traffic regulation events

NETSTAT IDS displays at V1R2

A new version of the `NETSTAT` command is also available with Communications Server in z/OS V1R2. The "`NETSTAT,IDS`" command (in UNIX, "`netstat -k`") displays the installed policies from the IP stack's perspective. In contrast, the `pasearch` command displays the policies of which PAGENT is aware.

5.12 OS/390 and z/OS syslog daemon security

The syslog daemon reads and logs system messages to the MVS console, log files, SMF, other machines, or users. If syslogd is not started, application data may appear on the MVS console.

You may wish to have remote systems from remote, critical UNIX servers, send their log messages to the syslog daemon on OS/390 (Z/OS) if there are important messages that you need to monitor from the mainframe.

Syslog facility names are specified in /etc/syslog.conf with associated log files which are stored in the Hierarchical File System (HFS). Depending on how the application has been coded, debug messages may go to one file and trace messages to another. The documentation for each server should indicate where the log debug messages, error messages, information messages and warning messages should be written. Some will log all of them to the same syslogd destination.

syslogd may be configured either to forward messages to another syslogd on another host instead of logging them to files, to send them directly to a specified user or to all users that are logged in on the system, or to send messages to the MVS console.

Note in Figure 134 how the syslogd process creates a UDP socket and binds to port 514 during initialization. This port is reserved to OMVS in the CS for OS/390 V2R10 IP stack and is also set aside in the /etc/services file. Local processes create an AF_UNIX socket to communicate with syslogd for logging purposes. An AF_INET socket is used for logging by and to remote servers. Herein lies a security exposure.

.

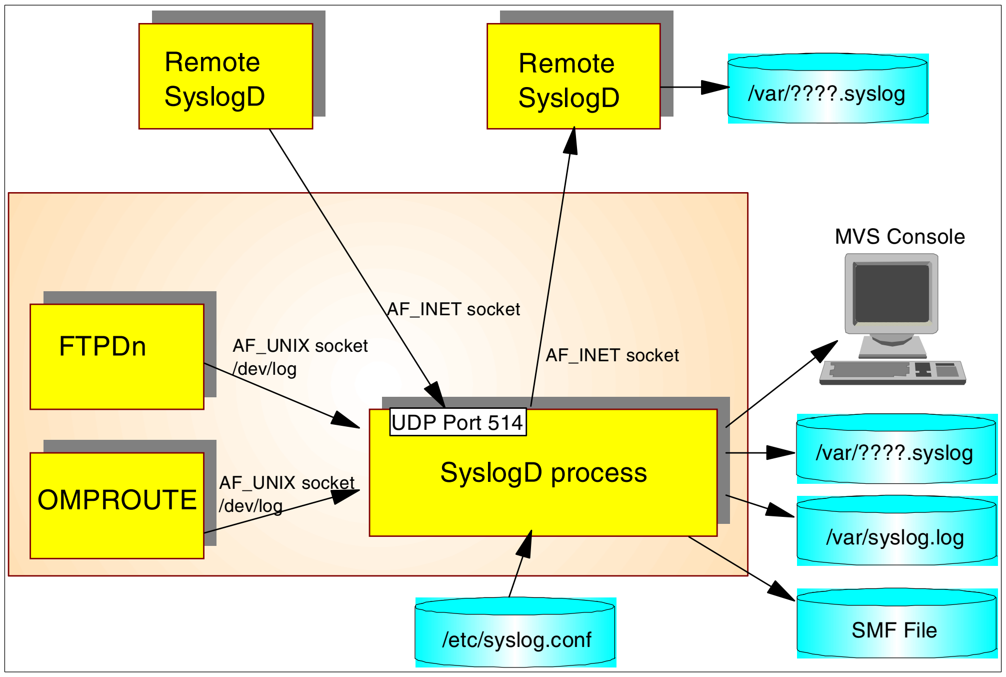


Figure 134. Syslog daemon processing

You may not wish to have remote syslogd servers log into your log files. Log files can grow notoriously large and impact performance on your system. Furthermore, should the log files fill up the HFS designated for your logging, all further logging is stopped and you may lose critical information about system activity. One safeguard against this eventuality is to ensure that you have set aside enough space in a separate HFS that has been allocated specifically for logging; this HFS is mounted at the logging directory mountpoint. (Prior to CS for OS/390 V2R10 IP the traditional mountpoint was the /tmp directory; with CS for OS/390 V2R10 IP the preferred mountpoint is the /var directory in order to follow the convention used by other UNIX platforms.) Another safeguard against excessive consumption of the logging HFS is to manage your log files, archiving and deleting them periodically so that the HFS does not fill up.

Such management procedures for syslogd are documented in *IBM Communications Server for OS/390 IP Configuration Guide*, SC31-8725 and *IBM Communications Server for OS/390 IP Configuration Reference*, SC31-8726. Yet another solution to excessive logging is to limit the remote users who may log into your syslogd log files. Remote syslogd users may represent a security threat to your system, either because they overload the system with messages or because they send bogus messages that cause disruption to the system operation.

If you intend to receive log data from or send log data to remote syslogd servers, you should place the syslog service in the services definition file or dataset so that an AF_INET socket may be built. Prior to CS for OS/390 V2R10 IP logging by remote syslogd servers was prevented by eliminating this reference to the syslog service at UDP port 514 in the /etc/services file. However, note that the omission of a service reference for syslogd also prevented OS/390 from logging messages at remote syslogd servers. The behavior of syslogd with respect to the /etc/services file changes in V2R10 and a new startup option, `-i`, is introduced to control the receipt of messages from remote syslog daemons.

5.12.1 syslogd isolation

Note the new options (`-i, -c, -u`) that may be specified when initializing syslogd on CS for OS/390 V2R10 IP:

```
syslogd [-f conffile] [-i][-u[-c[-d][-m interval] [-p logpath]
```

As you can see from Table 17, only the `-i` and `-u` options relate to security enhancements and what is called *syslogd isolation*. The term, syslogd isolation, refers to the capability of syslogd to isolate messages into separate logs with more granularity than was previously possible and to restrict communication with remote syslog servers more easily.

Table 17. Syslog daemon startup options

Option	Option Description
-c	Create log files and directories automatically
-d	Run syslogd in debugging mode
-f	Configuration file name
-i	Do not receive messages from the IP network
-m	Number of minutes between mark messages. The default value is 20 minutes
-p	Path name of OS/390 UNIX character device for the AF_UNIX datagram socket. The default value is /dev/log. This option is not used frequently. If you incorrectly use the -p option, syslogd will not function properly.
-u	For records received over the AF_UNIX socket (most messages generated on the local system), include the userid and jobname in the record.

CS for OS/390 V2R10 IP allows the syslogd on OS/390 to log at remote servers while prohibiting the remote servers to log at CS for OS/390 V2R10 IP itself. The `-i` command-line option at syslogd initialization tells syslogd not to receive messages via UDP (i.e., from syslog daemons on other hosts in the network). So, with `-i`, syslogd is still able to send messages to other systems, as in the following example:

```
*.panic       @bigiron.raleigh.ibm.com
```

If `-i` has been specified at syslogd startup and no remote destinations are named in the syslog.conf file, syslogd will not make use of AF_INET sockets at all.

We mentioned previously that it was typically necessary to set aside UDP port 514 for use by AF_INET sockets. However, this admonition is no longer necessary beginning with OS/390 V2R10, although it is still standard practice to include the entry in the /etc/services file. The following table displays which ports are bound to under various startup conditions for syslogd.

Table 18. syslogd behavior with/without -i startup option

	Startup with -i	**Startup without -i**
Remote host configured in /etc/syslog.conf	syslogd binds to ephemeral port, not to udp/514	syslogd binds to port udp/514
Remote host not configured in /etc/syslog.conf	syslogd does not bind to any port, neither an ephemeral port nor udp/514	syslogd binds to port udp/514

5.12.1.1 syslogd record selection criteria

With CS for OS/390 V2R10 IP there is more granularity in the criteria that may be used to select where log messages are written. You may now configure the CS for OS/390 V2R10 IP syslogd to accept messages based on the user ID and jobname associated with the running task issuing the syslog system call.

For example, prior to CS for OS/390 V2R10 IP, messages were routed to various log files defined in the syslogd configuration file (syslog.conf) by means of their facility classes and priority codes. See a sample of the facility classes and priority codes in Table 19 and Table 20.

Table 19. syslogd facility names

Facility Name	**Facility Function**
kern	OS/390 UNIX kernel messages
user	Mail generated by a user; default facility used by anyone who does not fall into one of the other categories
mail	Mail system messages
use	Usenet system messages
uucp	UUCP messages
daemon	Various server messages; used by the various servers (FTPD, rshd, and rexecd, OSNMPD, SNMP Subagent, TRMD)
auth	Authorization messages

Facility Name	Facility Function
authpriv	Same as auth
cron	CRON system messages
lpr	Printing system messages
local0-7	Facility names meant for local use; TelnetD uses local1 to log its messages; Firewall uses local0 and local4
mark	Logging of MARK messages
*	Placeholder that is used to represent any facility name

Table 20. syslogd priority codes

Priority Code	Priority Function
emerg	Emergency; system is becoming unusable
panic	Same as emerg
alert	Immediate action required
crit	Critical condition; a device or a component is becoming unusable
err(or)	Error condition
warn(ing)	Warning condition
notice	Normal but significant condition
info	Information message
debug	Debugging message
none	Placeholder used to represent none of the priorities
*	placeholder used to represent all priority codes

The old record selection criteria for syslogd messaging was:

```
facility.priority   destination
```

The facility and priority are any of the entries from Table 19 and Table 20. The destination may be a file, a userid, all logged-on users (careful!), syslogd on a remote system, or, in CS for OS/390 V2R10 IP, SMF. An example of this usage would be to send all daemon messages that are at a priority level of `err` or higher to the location of /var/log/daemon.err and all other messages of priority level of `err` or higher to /var/log/generic.err, while not duplicating `daemon` messages at that location:

```
daemon.err       /var/log/daemon.err

*.err;daemon.none  /var/log/generic.err
```

On OS/390, AF_UNIX sockets provide a mechanism for determining certain information about the sender. For AF_UNIX datagram sockets (vs. AF_UNIX stream sockets), this includes the user ID and jobname, allowing you to ensure that applications invoking the syslog system call are trusted applications. The new syslogd in CS for OS/390 V2R10 IP makes use of this additional information in order to isolate messages. It allows message selection using criteria which the application cannot forge: the user ID and jobname associated with the thread writing the message.

The new record selection criteria for syslogd messaging as of CS for OS/390 V2R10 IP is depicted as `[userid.jobname.] facility.priority` in Figure 135:

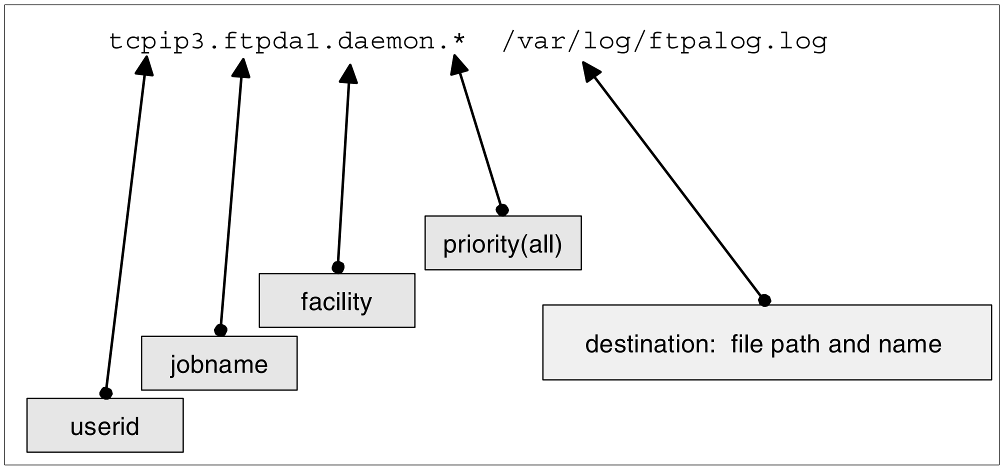

tcpip3.ftpda1.daemon.*   /var/log/ftpalog.log
priority(all)
facility
destination: file path and name
jobname
userid

Figure 135. Syntax for /etc/syslog.conf in CS for OS/390 V2R10 IP

Therefore, if you had planned to allow only those tasks associated with a specific userid to write to the log, you might create a logging rule in /etc/syslog.conf that sends only certain messages into logs managed by syslogd. For example, you see in Figure 135 that we have isolated messages from user ID `tcpip3`, jobname `ftpda1`, facility of `daemon`, and `all priority` types in /var/log/ftpalog.log.

As another example, you might code a rule that assumes that only tasks associated with the userid of MAINT or SERV may write to /var/log/messages:

```
MAINT.*.*.*;SERV.*.*.*              /var/log/messages
```

In taking advantage of this new syntax you might also consider ensuring that the userid and jobname are logged as well. This requires setting the `-u` option in the syslogd startup. We show you this syntax later in this chapter.

Even if you do not worry about illegitimate applications, selecting based on user ID and jobname is a big help in separating messages from applications that specify the same syslog facility. The previous method of categorizing messages continues to function but now we also have the possibility of separating out messages according to more criteria. For example, if you wanted to send all messages with a priority level of `err` or higher that are associated with any userid and any job whose name begins with the characters of `inetdcs` to a destination of /var/log/inetd.err, you would code as follows:

```
*.inetdcs*.*.err       /var/log/inetd.err
```

To log all other error messages to /var/log/generic.err while omitting any duplication of the already logged `inetdcs` messages you would code:

```
*.err;*.inetdcs*.*.none  /var/log/generic.err
```

5.12.1.2 User ID and jobname considerations for FTP

In Figure 135 on page 237 you saw how to isolate syslogd messages about the ftp server started task by using the record selection criteria available since OS/390 V2R10. In that figure, the jobname of `ftpda1` is derived from the parent process which forks an address space for a child process at the startup of the procedure. The actual procedure name would have been `ftpda`, with the forked address space receiving the suffix of 1: `ftpda1`. The server started task has been associated with the user ID of `tcpip3`, thus allowing us to create a logging rule as follows:

```
tcpip3.ftpda1.daemon.*  /var/log/ftpalog.log
```

This rule allows the capture of FTP server messages about the main task, but not messages pertaining to FTP user child processes. To exploit syslogd message isolation with FTP users requires an understanding of what happens when a user ftps into the system. When a user logs on to the FTP daemon, an additional process is created that runs in a forked address space separate from the main server address space. See Figure 136.

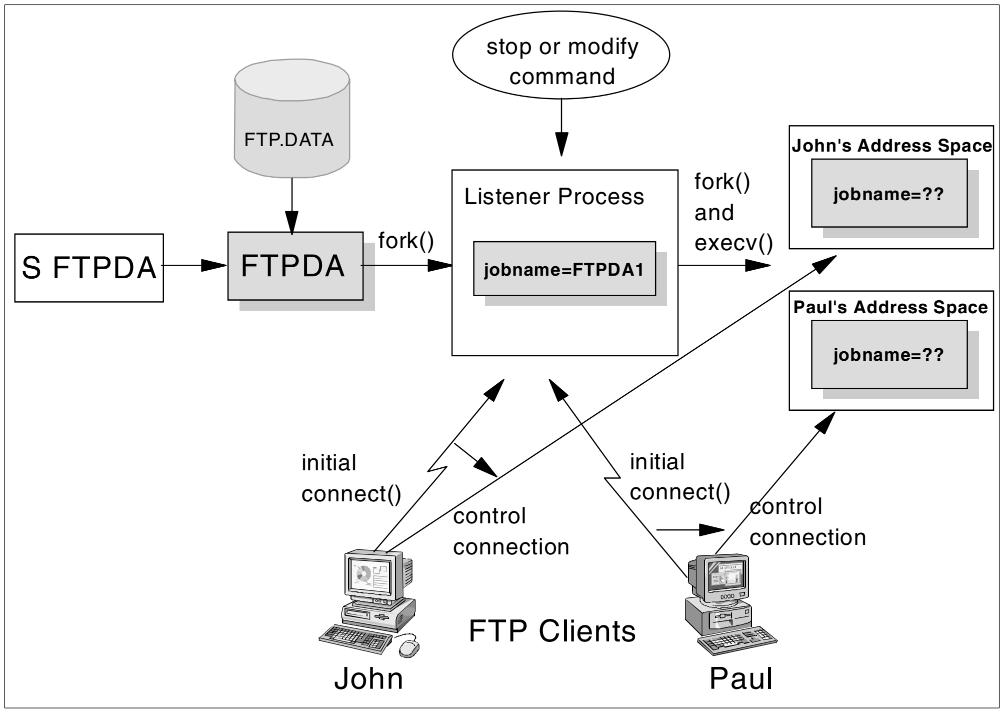


Figure 136. User address space creation

The jobname of the user's address space varies depending on the release and maintenance level of CS for OS/390 IP and on how you have exploited FTP. At OS/390 V2R5 and V2R6 IP, the FTP jobname associated with logged on users was created from the address space that was forked at the first RACF call. Assuming that the started task name was FTPDA, this user address space bore the jobname of FTPDAn, where n was a numbered suffix. The output of the `D OMVS,A=ALL` command reflects what this looks like in these versions of OS/390:

```
D OMVS,A=ALL
BPXO040I 21.57.15 DISPLAY OMVS 290
OMVS     000E ACTIVE          OMVS=(03)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
TCPIP3   FTPDA1   004F   67108883          1 1FI   21.53.05       .041
  LATCHWAITPID=         0 CMD=FTPD
GDENTE   FTPDA2   003D   67108886   67108883 1FI   21.55.13       .175
  LATCHWAITPID=         0 CMD=/usr/sbin/ftpdns 2138281240
```

Figure 137. FTP at V2R5, V2R6: user and jobname

Examine Figure 137. Note the FTP listener process of FTPDA1 as the jobname and TCPIP3 as the User ID. Also note how the FTP client process connection displays GDENTE as the user ID and FTPDA2 as the jobname.

The jobname was important to many customers for accounting and billing reasons. Prior to OS/390 V2R5, many customers had been relying on the presence of the FTP user ID in the jobname field. The behavior exhibited in Figure 137 meant unwelcome changes to their existing accounting routines. For such customers, APARS PQ14425 (V2R5, V2R6) and PQ27473 (V2R7, V2R8) were taken to ensure that the jobname field contained the user ID in lieu of the forked task name. OS/390 V2R10 base contains this function. See Figure 138.

```
D OMVS,A=ALL
BPXO040I 21.57.15 DISPLAY OMVS 290
OMVS     000E ACTIVE          OMVS=(03)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
GDENTE   GDENTE   003D   50331673   50331860 1FI   21.01.13       .178
  LATCHWAITPID=         0 CMD=/usr/sbin/ftpdns 2138211608
```

Figure 138. FTP: jobname equivalent to user ID

Note the user ID of GDENTE and the jobname of GDENTE when this behavior is in effect.

Other customers preferred to retain the jobname distinct from the user ID. For these customers, the solution offered with the previous APARs did not work. Furthermore, a return to the OS/390 V2R5 scenario was also undesirable, because this complicated Workload Manager (WLM) tuning. As a result, APAR PQ32536 was taken to allow a consistent jobname field for the FTP server for all releases from OS/390 V2R5 through V2R8. (The APAR is incorporated in the base of OS/390 V2R10.) When the environment variable _BPX_JOBNAME is passed on the PARM statement in the FTPD started procedure, every FTP user's jobname assumes a constant value.

```
D OMVS,A=ALL
BPXO040I 21.57.15 DISPLAY OMVS 290
OMVS     000E ACTIVE          OMVS=(03)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
GDENTE   FTPDA    0047   83886178   67108883 1FI   15.13.07       .175
  LATCHWAITPID=         0 CMD=/usr/sbin/ftpdns 2138436888
MATT     FTPDA    004D   83886224   67108883 1FI   15.12.20       .191
  LATCHWAITPID=         0 CMD=/usr/sbin/ftpdns 2138461464
```

Figure 139. FTP and _BPX_JOBNAME

Our example illustrates how _BPX_JOBNAME=FTPDA makes FTPDA the jobname for all FTP users: both GDENTE and MATT. Not only can customers take advantage of this in accounting routines and WLM customization, but they can also use this consistent jobname to implement syslogd isolation as you saw earlier in Figure 135 on page 237.

5.12.1.3 syslogd logging options

The ability to include user ID and job name in trace records is also a security measure. This prevents a program from being able to mimic the trace records of some other program. Specify the new option `-u` at syslogd initialization to include the user ID and jobname in trace records.

If you have specified `-u` at syslogd start, the messages will be logged as follows:

```
Feb 13 09:57:54 MPS030/USER3    SYSLOGD5 FSUM1220 syslogd: restart
Feb 13 10:02:31 cs390-1.raleigh.ibm.com/N/A      N/A      gobble
```

Without the `-u` at startup the user ID and jobname are omitted from the trace records:

```
Feb 13 09:59:43 MPS030 FSUM1220 syslogd: restart
Feb 13 10:03:24 cs390-1.raleigh.ibm.com gobble
```

Remote syslogd Servers

We don't know the user ID and jobname of messages received via UDP. Because the user ID and jobname is unknown for messages received from the syslogd of other systems, only selectors with *.* for userid.jobname will match such messages. In the following example, the first and second rules will match a daemon.info message received via UDP, but the other rules will not.

```
daemon.info       /var/log/a

*.*.*.info             /var/log/b

u*.*.daemon.info /var/log/c

*.j*.daemon.info  /var/log/d
```

The syslogd function in CS for OS/390 V2R10 IP includes the capability to write to SMF records (Type 109), a feature that was available in prior releases only in a separate version of the syslogd, one that was delivered with the Security Server. Some customers may have viewed this as a security feature that enabled them to track down certain types of security violations by examining SMF reports. At the OS/390 V2R10 release level there is only one version of syslogd and such customers preserve the capability to log to SMF if they use the following syntax in the syslog.conf file and have permitted the syslogd’s user ID to the facility BPX.SMF:

```
userid.jobname.facility.priority    $smf
```

It is expected that most administrators will not consider SMF an appropriate place for syslogd messages. The function is provided purely to aid customers who used this feature on previous releases.

For more information on syslogd and its many features, please consult *IBM Communications Server for OS/390 IP Configuration Guide*, SC31-8725, *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228, and *IBM Communications Server for OS/390 TCP/IP 2000 Update Technical Presentation Guide*, SG24-6162.

5.13 Express Logon Feature (ELF)

Express Logon Feature (ELF) was introduced in IBM Communications Server for OS/390 V2R10 IP Services. ELF allows a user on a workstation, with a TN3270 client and an X.509 certificate, to log on to an SNA application *without* entering a user ID or password.

The ELF function has been added to the TN3270 server on CS for OS/390 V2R10 IP with APAR PQ47742.

Express Logon Feature allows users to:

- Reduce the time administrators spend maintaining user IDs and passwords.
- Reduce the number of user IDs and passwords that users have to remember.
- Remove a potential security risk of users writing down user IDs and passwords, or sharing them with someone else.

5.13.1 General requirements

The following IBM products support the Express Logon Feature at the time of writing this redbook:

- Host On-Demand (HOD) V5.0 as a TN3270 client.
- TN3270 servers (at least one of the following is required)
  - Communications Server for OS/2 6.1
  - Communications Server for Windows NT 6.1.1 PTF
  - Communications Server for AIX 6.0.0.1 PTF
- OS/390 V2R10 with the following required components:
  - Communications Server for OS/390 V2R10 plus APAR PQ41276 - provides SNA and TCP/IP transport, and the Digital Certificate Access Server (DCAS) which supports the Express Logon Feature by interfacing to the TN3270 servers and RACF.
  - Security Server for OS/390 V2R10 (RACF) - provides general security services and services for digital certificates and PassTicket.
  - RACF APAR OW44393 is required when using one of the following:
    - TSO with Generic Resources and PTKTDATA class profiles.

- Applications with shared user IDs that could access the application simultaneously. RACF requires the PTKTDATA profile to specify `APPLDATA('NO REPLAY PROTECTION')`.

What's new in z/OS V1R2

The TN3270E server shipped with z/OS Communications Server V1R2 supports the ELF as well, so that this functions can be implemented without using the three-tier network topology.

5.13.2 ELF in the three-tier network design with OS/390 V2R10

In OS/390 V2R10, ELF supports a three-tier and a two-tier network design. Without APAR PQ47742, because the OS/390 TN3270E server does not support the requestor role in the ELF configuration, a TN3270 server running on other platforms, such as AIX and Windows NT, has to be installed as a middle-tier server.

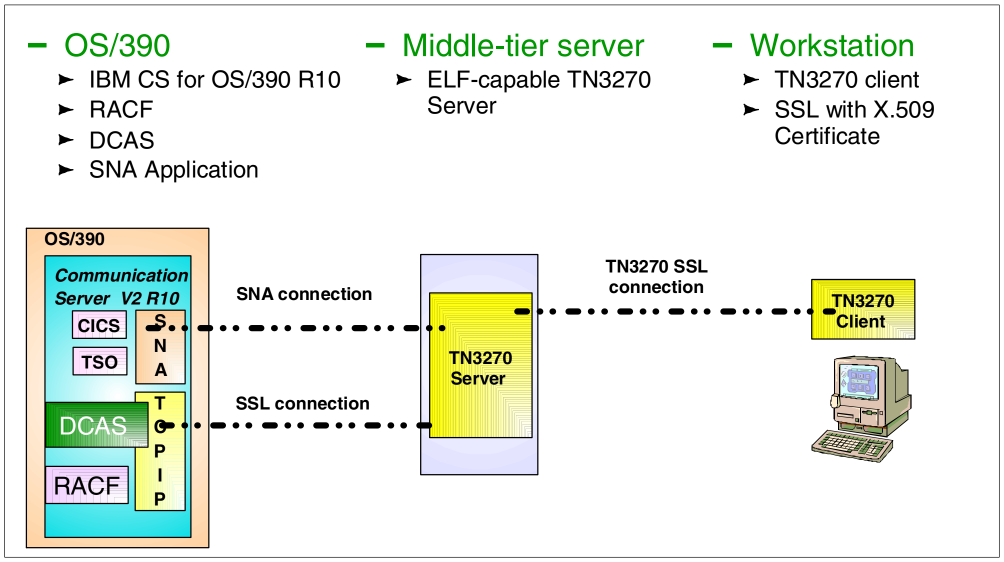


Figure 140. ELF: three-tier network design

In this scenario, the ELF design consists of the three-tier network components (see Figure 140):

- A client workstation that supports Secure Sockets Layer (SSL) connections with client authentication and an X.509 certificate.
- A *middle-tier* TN3270 server, so called because it does not reside on the host, but rather between the client and the host. This server communicates with a DCAS using an SSL connection with client authentication. It sends the user's certificate from the workstation and an application ID to the DCAS and expects to receive a user ID and PassTicket (a one-time password) in response. This is the user ID and password that will be used to logon to the SNA application.
- The Digital Certificate Access Server (DCAS) resides on the host. The DCAS uses RACF services to obtain a user ID that is associated with the certificate sent by the client. RACF also provides secured sign-on services, which the

DCAS uses to generate a *PassTicket*. A PassTicket is a RACF token similar to a password except that it is valid only for 10 minutes.

The SNA connection between the TN3270 server and SNA application can be SNA LU2, DLUR, HPR/IP (EE), or AnyNet connection.

Note: The Secure Sockets Layer (SSL) communication with client authentication is required in the configuration of the Express Logon Feature on the HOD client, TN3270 server, and OS/390 DCAS server.

5.13.3 ELF flow overview: three-tier network design

In this section, we will describe the simplified flow used in the ELF environment. In the following example, we assume that a user at a workstation wants to logon to a TSO application on the host.

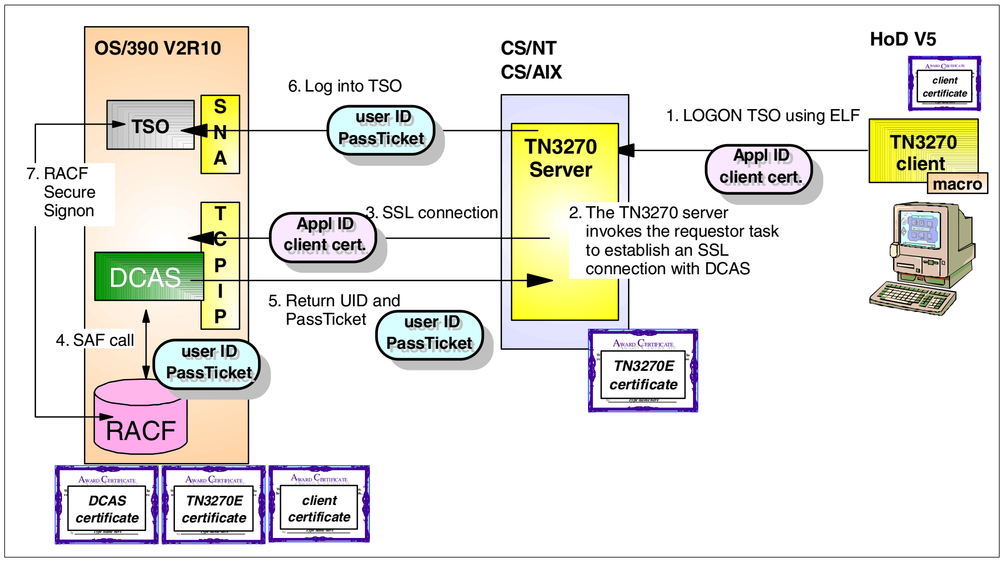


Figure 141. Simplified ELF flow: three-tier network design

Before starting the ELF procedure some configuration steps are required, the following is an overview:

On the workstation is a HOD V5 emulator session configured to use SSL client authentication. A macro should be created to record the logon to the desired SNA application. The X.509 certificate on the workstation must be extracted and stored in the keyring of the TN3270 middle-tier server. The TN3270 server's certificate must also be extracted and stored on the workstation to be used for SSL server authentication. The workstation's x.509 certificate must be extracted and defined in RACF on the host to be associated with a valid user ID for the SNA application. For ELF, the HOD V5 client has an Application ID popup window where the name of the SNA application that the user will logon to must be specified. Note that this Application ID must match the application name of the RACF PTKTDATA profile that must be defined on the host so that a PassTicket can be generated for the application. Since the SSL connection between the TN3270 middle-tier server and DCAS requires client authentication, the certificate from the TN3270 server must be extracted and stored on the host in the keyring that the DCAS is using. The DCAS certificate must also be extracted and stored in the keyring of the TN3270 server to be used during the SSL handshake.

Then the user starts the TN3270 connection. The overview of the communication flow is as follows:

1. During the SSL handshake between the TN3270 client and server, the TN3270 client certificate is passed to the TN3270 middle-tier server and authenticated. Also, during the TN3270 exchange, the ELF capability is negotiated using the New Environment (NE) Telnet option defined in RFC 1572 - *Telnet Environment Option*. The application ID (from the HOD popup window) is sent from the client to the TN3270 server under the NE Telnet option. The logon screens come to the emulator as usual, and the recorded macro plays and inserts placeholder strings in the user ID and password fields, which are $usr.id$ and $pss.wd$ respectively. The TN3270 server intercepts the placeholder strings. This prompts the TN3270 server to connect to the DCAS.
2. Then the TN3270 server invokes the requestor function to establish the SSL communication to a DCAS. The TN3270 server's certificate has to be sent to the DCAS to get SSL handshake with client authentication completed.
3. The connection between the TN3270 server and the DCAS is an SSL connection with client authentication. This handshake provides DCAS with the certificate of the TN3270 sever which (depending upon the CLIENTAUTH DCAS configuration parameter) can be used for additional security checks. After an SSL connection is successfully established, the TN3270 server sends the client's certificate and the target application ID to the OS/390 DCAS over the secure, trusted connection.
4. The DCAS makes SAF calls to convert the HOD client's certificate and the application ID to a TSO user ID and PassTicket.
5. The DCAS sends the user ID and PassTicket back to the TN3270 server.
6. Then the server inserts the user ID and PassTicket into the 3270 datastream at the macro-inserted placeholder locations and sends it to an TSO application via an SNA connection.
7. The TSO application presents the user ID and PassTicket to RACF or other compatible host access control facility, which approves them and the logon completes as usual.

The DCAS server can use either an HFS key database or RACF common keyring to authenticate the TN3270 middle-tier server.

The client certificates have to be stored in the RACF data base and associated with the user ID that the HOD client will use to log into the SNA application.

The TN3270 server acts as a client that interacts with the DCAS. Authenticating the TN3270 server involves additional levels of control in which the client must have a key database with a certificate. There are three levels of client authentication you may implement. Depending on the security level you are using, the TN3270 server's certificate may have to be stored into the RACF data base with a valid user ID as well.

For more information about the ELF implementation with OS/390 V2R10 using the three-tier network design, refer to the following documents:

- White paper *Setting up and Using the IBM Express Logon Feature* available at

  `http://www.ibm.com/software/network/commserver/library/`
- *IBM Communications Server for OS/390 TCP/IP 2000 Update Technical Presentation Guide*, SG24-6162
- *Express Logon User's Guide* available at:

  `http://www.ibm.com/software/network/commserver/library/whitepapers/csos390.html`

5.13.4 ELF enhancements: two-tier network design support

In z/OS V1R2 and OS/390 V2R10 with APAR PQ47742, the TN3270E server is enhanced to support the ELF, so that the DCAS is not required. From the TN3270 client's point of view, the flow is identical regardless of the TN3270 server's location.

Figure 142 illustrates the simplified data flows using this enhancement:

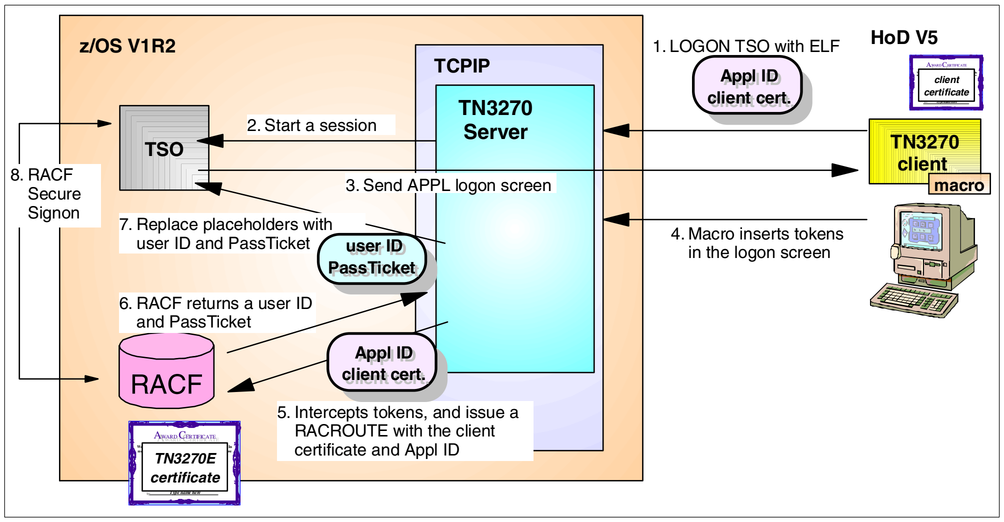

z/OS V1R2
TCPIP
TN3270 Server
TSO
RACF
TN3270E certificate
2. Start a session
3. Send APPL logon screen
7. Replace placeholders with user ID and PassTicket
user ID PassTicket
6. RACF returns a user ID and PassTicket
Appl ID client cert.
5. Intercepts tokens, and issue a RACROUTE with the client certificate and Appl ID
8. RACF Secure Signon
1. LOGON TSO with ELF
Appl ID client cert.
HoD V5
client certificate
TN3270 client
macro
4. Macro inserts tokens in the logon screen

Figure 142. Simplified ELF flow: exploiting the two-tier network design

The configuration setup required is almost the same as the three-tier model:

A macro should be created to record the logon to the desired SNA application. The X.509 certificate on the workstation must be extracted stored into the SSL keyring that is used by the TN3270 server. The TN3270 server's certificate must also be extracted and stored in the keyring on the workstation. This design also requires that the workstation certificate be extracted and associated in RACF with a valid user ID of the SNA application that the user wants to logon to. For ELF, the HoD V5 client has an application ID popup window where the name of the SNA application that the user will logon to must be specified. Note that this application ID must match the application name of the RACF PTKTDATA profile that must be defined on the host so that a PassTicket can be generated for the application.

The end user connects to the TN3270 server on the host and the TN3270 server saves the certificate sent from the client during an SSL handshake. The TN3270 client and server negotiate the ELF option using the telnet New Environment (NE) option. As in the three-tier case, the application ID is passed to the TN3270 server using the NE option.

The application's logon screen is sent to the client, and the recorded macro enters the tokens for the user ID and password ($usr.id$ and $pss.wd$ respectively). The TN3270 server intercepts the tokens, sends the client certificate and the NE application ID value to the security product, such as RACF, to retrieve the RACF user ID associated with the certificate and the PassTicket to be used for the SNA application. When the user ID and PassTicket are returned, the TN3270 server substitutes these for the tokens and forwards the logon screen to the application. (The user ID and PassTicket are requested separately.)

The application validates the user ID and PassTicket with the RACF help, and if they are authenticated the application session is established.

5.14 Routing protocol security

Establishing the routing tables in an IP network is critical to successful communication across that network. Many smaller network configurations rely strictly on static routing definitions to maintain the routing infrastructure. As with other manual processes, this method is prone to errors and is time-consuming. As a result, when the management of the network routing tables becomes too labor-intensive, a more efficient and reliable means of dealing with the network routes is desirable. This is where dynamic routing protocols play a significant role.

With dynamic routing protocols, changes to the routes occur through dynamic routing updates. They are essentially transparent to the network administrator since the updates are exchanged among the routers participating in the network. With the proper access to the physical infrastructure and wiring of an IP network, it would be possible for someone to introduce an unauthorized router into the network if dynamic routing protocols were in effect. The “rogue” router could then either interconnect the existing network with an untrusted network over which the dynamic routing updates would continue to flow or could introduce routing updates that are disruptive to the normal operation of the network.

To protect against these eventualities, both RIP Version 2 and OSPF provide for the authentication of routing update packets. Both ORouteD and OMPROUTE in CS for OS/390 V2R10 IP allow the configuration of a password for RIP V2 and/or OSPF routing updates. The password represents an authentication key. This key is exchanged with adjacent routers which must have been configured with the same password. Thus routing update packets from any unauthorized router that has infiltrated the network will be ignored unless the router has been configured with the same authentication key. Note that, both the RIP V2 and OSPF keys are transmitted in the clear as of CS for OS/390 V2R10 IP.

In z/OS V1R2, enhancements are being made to OSPF in OMPROUTE to allow MD5 authentication to be used. To implement MD5 authentication, a 16-byte MD5 hash key has to be defined on an interface or area basis. The entire packet is hashed with this key and the results, which is also referred as *message digest*, is appended to the message. The receiver then runs its copy of the MD5 key over the message and verifies that the digest it computes matches the one that was received. The MD5 algorithm is documented in RFC 1321, and the use of MD5 authentication with OSPF updates is discussed in RFC 2328.

Warning

Without MD5 authentication, the clear-text password is not a fool-proof shield against a malicious router's intrusion into a network. It may be regarded only as a safeguard against the accidental introduction of an invalid routing node into your networking infrastructure.

Figure 143 shows the ORouteD profile, in which you can specify a global authentication keyword (password) to exchange between routers in the network. Note the keyword: `RIP2_AUTHENTICATION_KEY: password`. Figure 146 on page 249 shows how the ORouteD options statement can be used to override the globally set authentication password for individual interfaces: `options interface name ip_addr key "authentication_key"`. RIP V2 in OMPROUTE has comparable parameters on the `RIP_Interface` statement. See Figure 147 on page 250: `Authentication_Key--=--key`.

```
RIP_SUPPLY_CONTGROL: RIP2M
RIP_RECEIVE_CONTROL: ANY
RIP2_AUTHENTICATION_KEY: “password”
```

Figure 143. The ORouteD profile (syntax)

OSPF in OMPROUTE designates the globally overriding authentication type (simple password or none) in the AREA statement. See Figure 144.

Note: Enhancements are being made in z/OS V1R2 to allow the authentication type to be defined by area or by interface.

```

          +-Area_Number=0.0.0.0---------------+
>>--Area--+-----------------------------------+-------------------------->
          +-Area_Number--=--ospf_area_address-+

   +-Authentication_Type=None----------------+
>--+-----------------------------------------+--------------------------->
   +-Authentication_Type--=--security_scheme-+

   +-Stub_Area=NO--------+  +-Stub_Default_Cost=1--------+
>--+---------------------+--+----------------------------+--------------->
   +-Stub_Area--=--value-+  +-Stub_Default_Cost--=--cost-+

   +-Import_Summaries=YES-------+
>--+----------------------------+-------------------------------------><
   +-Import_Summaries--=--value-+
```

Figure 144. OMPROUTE configuration file: AREA definition

Individual OSPF interfaces specify the Authentication_Key that should be used with other routers reachable through that interface. See Figure 145.

```
>>--OSPF_Interface--IP_address--=--ip_address--Name--=--interface_name--->

>--Subnet_mask--=--subnet_mask-------------------------------------------->
                                +-Destination_Addr--=--address-+

   +-Attaches_To_Area=0.0.0.0--+  +-MTU=576----------+
>--+---------------------------+--+------------------+------------------->
   +-Attaches_To_Area--=--area-+  +-MTU--=--mtu_size-+

   +-Retransmission_Interval=5-------------+
>--+---------------------------------------+----------------------------->
   +-Retransmission_Interval--=--frequency-+

   +-Transmission_Delay=1---------+  +-Router_Priority=1------------+
>--+------------------------------+--+------------------------------+---->
   +-Transmission_Delay--=--delay-+  +-Router_Priority--=--priority-+

   +-Hello_Interval=10-----------+
>--+-----------------------------+--------------------------------------->
   +-Hello_Interval--=--interval-+

   +-Dead_Router_Interval=40-----------+  +-Cost0=1--------+
>--+-----------------------------------+--+----------------+------------->
   +-Dead_Router_Interval--=--interval-+  +-Cost0--=--cost-+

   +-Subnet=NO--------+  +-Authentication_Key=nulls--------+
>--+------------------+--+---------------------------------+------------->
   +-Subnet--=--value-+  +-Authentication_Key--=--password-+
```

Figure 145. OSPF_Interface: coding for router authentication

It may also be desirable to filter out routing updates from routers that are legitimately part of the network configuration. Filtering may be implemented for several reasons: to force traffic distribution over specific routes or to eliminate certain paths from consideration for security purposes. RIP in both ORouteD and in OMPROUTE provides for options to restrict certain types of routing update broadcasts from being sent or received.

For example, note the `block`, `noreceive`, `noforward`, `ripoff`, `passive` (equivalent to `ripoff`) and `supply off` parameters in ORouteD's gateways file in Figure 146. (More information on the meaning of these parameters may be found in *IBM Communications Server for OS/390 IP Configuration Reference*, SC31-8726.)

```
>>----OPTIONS----gateway ip_addr ---------------------------------------><----
                |                  +-block-----|
                |                  +-noreceive-|
                |                  +-none------+
                +-interface.poll.interval timer_value --------------------------
                +-interface.scan.interval timer_value --------------------------
                +-interface name ip_addr block destination ---------------------
                +-interface name ip_addr forward destination fmask fmask ------
                +-interface name ip_addr forward.cond destination fmask fmask
                +-interface name ip_addr noforward destination fmask fmask -
                +-interface name ip_addr none-----------------------------------
                +-interface name ip_addr noreceive destination fmask fmask -
                +-interface name ip_addr passive -------------------------------
                +-interface name ip_addr ripon ---------------------------------
                +-interface name ip_addr ripoff --------------------------------
                +-interface name ip_addr receive destination fmask fmask ------
                +-interface name ip_addr receive.cond destination fmask fmask
                +-interface name ip_addr supply off-----------------------------
                +-interface name ip_addr supply on------------------------------
                +-interface name ip_addr key ----------------------------------
                |                              +-"authentication_key" -+
                +-interface name ip_addr nokey---------------------------------
                +-interface name ip_addr supply.control------------------------
                |                                        +-RIP1--|
                |                                        +-RIP2B-|
                |                                        +-RIP2M-|
                |                                        +-RIP2--|
                |                                        +-NONE--+
                +-interface name ip_addr receive.control-----------------------
                                                          +-RIP1-|
                                                          +-RIP2-|
                                                          +-ANY--|
                                                          +-NONE-+
```

Figure 146. ORouteD options syntax in gateways file or dataset

Note the comparable input and output filters for RIP in OMPROUTE in Figure 147: `filter--=--(filter_type,dest_route,filter_mask)`.

```
>>--RIP_Interface--IP_address--=--address--Name--=--interface_name------->

>--Subnet_mask--=--subnet_mask------------------------------------------->
                                +-Destination_Addr--=--address-+

   +-MTU=576------+  +-Receive_RIP=YES-------+
>--+--------------+--+-----------------------+--------------------------->
   +-MTU--=--size-+  +-Receive_RIP--=--value-+

   +-Receive_Dynamic_Nets=YES-------+
>--+--------------------------------+------------------------------------>
   +-Receive_Dynamic_Nets--=--value-+

   +-Receive_Dynamic_Subnets=YES-------+
>--+-----------------------------------+--------------------------------->
   +-Receive_Dynamic_Subnets--=--value-+

   +-Receive_Dynamic_Hosts=NO--------+
>--+---------------------------------+----------------------------------->
   +-Receive_Dynamic_Hosts--=--value-+

>------------------------------------------------------------------------>
   +-filter--=--(filter_type,dest_route,filter_mask)-+

   +-Send_Only=ALL----------+  +-Send_RIP=YES-------+
>--+------------------------+--+--------------------+-------------------->
   +-Send_Only--=--(values)-+  +-Send_RIP--=--value-+

   +-Send_Default_Routes=NO--------+  +-Send_Net_Routes=YES-------+
>--+-------------------------------+--+---------------------------+------>
   +-Send_Default_Routes--=--value-+  +-Send_Net_Routes--=--value-+

   +-Send_Subnet_Routes=YES-------+  +-Send_Static_Routes=NO--------+
>--+------------------------------+--+------------------------------+---->
   +-Send_Subnet_Routes--=--value-+  +-Send_Static_Routes--=--value-+

   +-Send_Host_Routes=NO--------+
>--+----------------------------+---------------------------------------->
   +-Send_Host_Routes--=--value-+

   +-Send_Poisoned_Reverse_Routes=YES-------+
>--+----------------------------------------+---------------------------->
   +-Send_Poisoned_Reverse_Routes--=--value-+

   +-In_Metric=1----------+  +-Out_Metric=0----------+
>--+----------------------+--+-----------------------+------------------->
   +-In_Metric--=--metric-+  +-Out_Metric--=--metric-+

   +-RipV2=NO--------+  +-RipV1_Routes=NO--------+
>--+-----------------+--+------------------------+----------------------->
   +-RipV2--=--value-+  +-RipV1_Routes--=--value-+

   +-Authentication_Key=nulls---+  <-----------------------+
>--+----------------------------+---------------------------------------->
   +-Authentication_Key--=--key-+   +-Neighbor--=--value-+


... etc ...
```

Figure 147. RIP_INTERFACE statement in OMPROUTE

In addition to the filters in ORouteD, the OMPROUTE RIP filters apply to both RIP V1 and RIP V2. However, be aware of the fact that RIP V1 does not support authentication. If you code that you want to accept RIP V1 packets by specifying `ANY` or `RIP1` for your `RECEIVE_RIP` value, you are agreeing to accept unauthenticated RIP V1 packets, since that is the only type there is. This is the case even if you have an authentication key coded on your RIP_INTERFACE statement, since this behavior is mandated by RFC 1723.

Note: The RECEIVE_RIP values of RIP1, RIP2, and ANY will not be supported until z/OS V1R2. It defines the version of RIP packets to be received. Prior to it, OMPROUTE can only send and receive the same version of RIP over an interface.

OSPF in OMPROUTE does not provide for such explicit filters. Instead, OSPF in OMPROUTE provides a mechanism to import only selected route types so that certain routing update packets are not disseminated throughout the network by means of the OSPF protocol. The AS_Boundary_Routing statement allows an Autonomous System Boundary router (ASBR) to determine which routes it will import from other routing protocols and therefore send to its neighbors in the form of Link State Advertisements (LSAs). This statement with its `Import` parameters is depicted in Figure 148.

```
                        +-Import_RIP_Routes=No--------+

>>--AS_Boundary_Routing--+----------------------------+----------------->
                        +-Import_RIP_Routes--=--value-+

   +-Import_Static_Routes=No--------+
>--+--------------------------------+----------------------------------->
   +-Import_Static_Routes--=--value-+

   +-Import_Direct_Routes=No--------+
>--+--------------------------------+----------------------------------->
   +-Import_Direct_Routes--=--value-+

   +-Import_Subnet_Routes=Yes-------+
>--+--------------------------------+----------------------------------->
   +-Import_Subnet_Routes--=--value-+

   +-Originate_Default_Route=No--------+
>--+-----------------------------------+-------------------------------->
   +-Originate_Default_Route--=--value-+

   +-Originate_as_Type=2--------+  +-Default_Route_Cost=1--------+
>--+----------------------------+--+-----------------------------+------->
   +-Originate_as_Type--=--type-+  +-Default_Route_Cost--=--cost-+

>----------------------------------------------------------------------><
   +-Default_Forwarding_Address--=--address-+
```

Figure 148. AS_BOUNDARY_Routing statement in OMPROUTE

For example, if an ASBR specifies Import_Rip_Routes=Yes, this information may be flooded to all neighbors as *external routes*, thus notifying them of destinations in networks running the RIP protocol. If the ASBR specifies Import_Rip_Routes=No, the other OSPF routers with which it is communicating do not learn about any of the destinations in the RIP part of the network. In this fashion the ASBR can screen out routing information that could have been obtained from non-OSPF routers and ensure that it is not propagated to its OSPF neighbors. Note that the OSPF protocol does not provide for the filtering out of specific routes to individual hosts or networks as does the filtering mechanism in the RIP protocol.

However, OSPF in OMPROUTE does permit the specification of a RANGE of subnetworks or networks attached to an Area Border Router (ABR); the RANGE may then be advertised or not in order to suppress the knowledge of that (sub)network range outside the area. See the syntax of the RANGE statement in Figure 149.

```
>>--Range--IP_address--=--address--Subnet_Mask--=--mask------------------->

   +-Area_Number=0.0.0.0--+  +-Advertise=YES-------+
>--+----------------------+--+---------------------+---------------------><
   +-Area_Number--=--area-+  +-Advertise--=--value-+
```

Figure 149. OMPROUTE RANGE statement

If you have reasons to isolate a range of (sub)networks, perhaps for security purposes, you may specify `Advertise=NO` on the RANGE statement.

For information relating to the implementation and syntax of these security features in ORouteD and OMPROUTE, please consult the *IBM Communications Server for OS/390 IP Configuration Guide*, SC31-8725 and *IBM Communications Server for OS/390 IP Configuration Reference*, SC31-8726. For a short tutorial on OSPF, please see *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 1: Configuration and Routing*, SG24-5227.

Chapter 6. SecureWay Security Server Firewall Technologies

Chapter 5, “TCP/IP application security” on page 101 already contains details of how Secure Sockets Layer (SSL), or Transaction Layer Security (TLS), can be implemented with various applications in z/OS. As Figure 150 illustrates, SSL provides secure communication by enabling applications to invoke a sockets application programming interface (API) that communicates with the Transport Layer of the protocol stack. SSL protects the entire communication path from client to server.

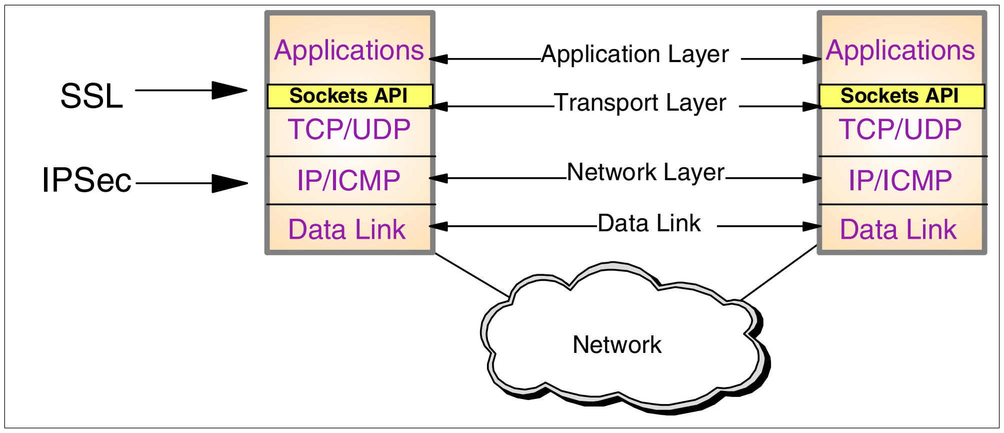


Figure 150. Protocol layers and security

SSL addresses concerns about authentication, integrity checking, and privacy. Applications taking advantage of SSL must be modified to invoke the API. Common applications capable of SSL/TLS communications are the HTTP server, theTN3270 server in CS for OS/390 V2R6 IP and higher, and the FTP server in z/OS V1R2 and higher.

Note: The TN3270 server supports SSL only, not TLS.

Figure 150 also depicts how security may be applied at the Network Layer of the protocol stack with the use of IPSec as you have already read in 2.3, “Virtual private network (VPN) and IPSec” on page 29. When IPSec is implemented to create a Virtual Private Network (VPN), any application may take advantage of it; there is no need for application modification. Application traffic passes through a channel or tunnel that provides the authentication, integrity checking, and encryption that is deemed appropriate. IPSec tunnels can even be assigned on an application basis, specifying port numbers and/or IP address information, just as one would do for normal IP filtering mechanisms. Unlike SSL, IPSec is a connectionless security protocol applied on a per-packet basis. Also unlike SSL, IPSec may protect not only TCP traffic but also UDP and RAW socket flows.

Typically we think of firewalls as the endpoints of an IPSec tunnel. However, the IPSec tunnel may be built between application endpoints as well. Alternatively, one end of the tunnel may reside in a node dedicated to firewall activity and the other end may reside at the application node. Therefore either the entire communication path may be protected or only a portion of it.

However, firewalls can provide more than access to a tunnel capable of authenticating, encrypting, and checking the integrity of the data that passes through it.   For example, you may implement a firewall to provide controlled access between a private (trusted) network and an untrusted network such as the Internet without addressing concerns about authentication, privacy and data integrity. See Figure 151.

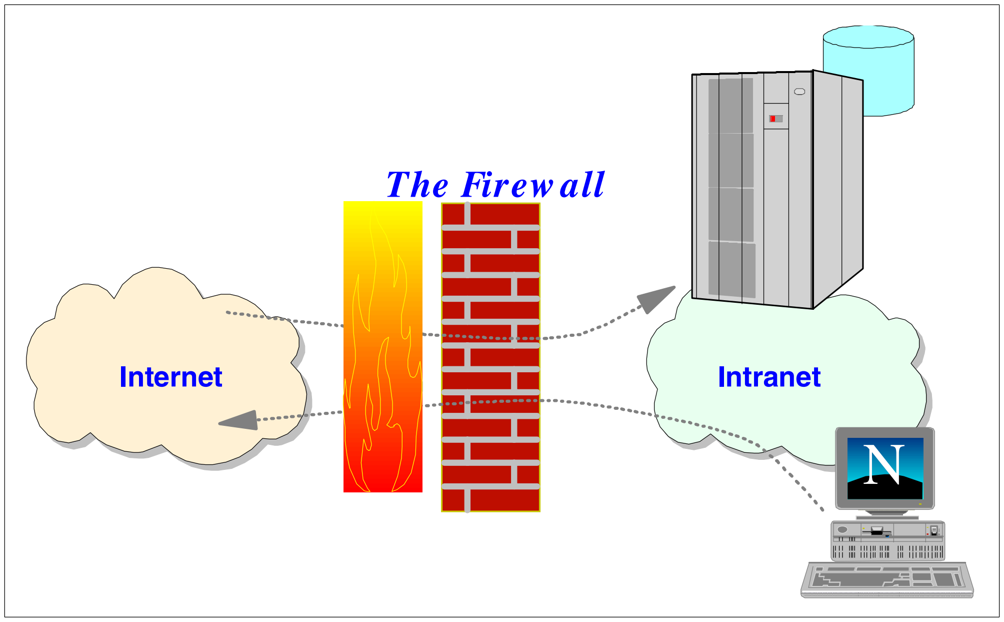


Figure 151. A firewall

The controlled access depicted in Figure 151 could encompass any of the functions provided by a firewall. Assuming that OS/390 Firewall Technologies represents this firewall, these functions are:

1. Packet filtering
2. Secure tunneling with the IPSec architecture (Virtual Private Network establishment)
3. SOCKS server
4. FTP proxy server
5. Network Address Translation (NAT) from an internal IP address to an external address and vice versa

Although Figure 151 implies that the firewall is located outside the Web server platform, in fact it may reside on the platform itself. Furthermore, there may be more than one firewall in the network, the second of which represents an additional bastion against illicit intruders. This represents a good fit for the OS/390 Firewall Technologies, whose functionality is located in two components: the OS/390 Security Server and CS for OS/390 IP.

6.1 IP security technologies

Table 21 compares and contrasts the various security technologies that are available in z/OS.

Table 21. Characteristics of IP security technologies

Solution	Access control	Encryption	Authentication	Integrity checking	Key exchange	Concealing internal addresses	Replay protection	Session monitoring	UDP support
IP Filtering	X								X
IPSec	X	X (packet)	X (packet, IP host) 1	X (packet)	X	X	X		X
SSL	X	X (application data)	X (application data, user) 1	X	X		X	X	
Policy Agent	X							X (connection & data)	X
TCP/IP Stack Security	X								X
Note: 1 The authentication function can be categorized into data origin authentication and identity authentication. IPSec supports data origin authentication for packets, SSL does it for application data. Regarding identity authentication, IPSec performs it for IP hosts, and SSL for users.									

The table is self-explanatory with the exception of a few of the fields, which we will describe next. Note that SSL provides encryption of the IP packet data whereas IPSec can encrypt not only the data but also the TCP and/or IP headers, depending on whether ESP transport or tunnelling mode is implemented. SSL provides authentication of the endpoint of the connection and as such can authenticate the server ("system") and optionally the client ("user"). Turning our attention to Session Monitoring, we see that SSL and Policy Agent are both capable of this function. SSL as a function residing at the transport layer but interfacing with the application layer must keep track of connection establishment and termination (or state information) in order to enforce its policies. The same can be said of Policy agent, which must stay aware of connection limits. Policy agent also keeps track of the data in order to enforce performance criteria and in order to set QoS for applications, including QoS for tasks running under WebSphere Application Server (WAS). This function was introduced in V2R10.

Several redbooks and IBM manuals treat the subject of OS/390 Firewall Technologies in detail: *OS/390 V2R10.0 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835 and *TCP/IP Tutorial and Technical Overview*, GG24-3376. 2.2, “Firewall concepts” on page 24 is another source of overview information on the subject.

Before we turn our attention to the two most heavily used features in Firewall Technologies, Packet Filtering and VPN/IPSec, it is helpful to review the evolution of Firewall Technologies.

6.2 Evolution of Firewall Technologies

The security functions of z/OS are found in both the base product and the SecureWay Security Server for OS/390. The Security Server, an optional element of OS/390, includes Resource Access Control Facility (RACF), Lightweight Directory Access Protocol (LDAP) server support, Firewall Technologies and the Distributed Computing Environment (DCE) Security Server. Table 22 represents which function is supported by which component(s).

Table 22. Location of security function in OS/390 and z/OS

	OS/390 or z/OS Communications Server	**OS/390 or z/OS Security Server Firewall Technologies**	**OS/390 or z/OS Security Server RACF**
IPSec	X		
IKE		X	X 1
VPN configuration commands		X	
VPN configuration GUI		X	X 2

Note:
1 The OS/390 (or z/OS) ISAKMP server utilizes SAF interface to access X.509 certificates from an External Security Manager (ESM), such as RACF.
2 Access to the configuration server is controlled by an ESM that implements the required SAF interfaces. RACF is an example of such an ESM.

OS/390 Firewall Technologies IPSec VPN provides a secure pathway between OS/390 and other IPSec VPN capable systems, routers and firewalls. This secure pathway is achieved through encryption techniques, utilizing the industry-leading S/390 hardware CMOS Cryptographic Coprocessor.

The IBM Communications Server for OS/390 IP Services provides a secure communications gateway for connecting diverse application and network environments. Included as a base element of OS/390, the Communications Server provides mission-critical, business-to-business and business-to-consumer communications across local area networks, wide area networks, intranets and the Internet.

A major enhancement to OS/390 Communication server is the possibility to create encrypted tunnels through insecure networks. This support is provided by the IPSec implementation in OS/390 Communications Server. IPSec is a world wide standard providing network layer encryption. As the Internet is a major business tool, the need for security keeps rising with time. IPSec can play an important role in this. With the support of IPSec on OS/390, the z/OS platform is able to participate in Virtual Private Networks as a powerful and secure server.

6.2.1 OS/390 V2R5

IP Security (IPSec) was introduced in the normal OS/390 V2R5 release, where it was implemented partly in the OS/390 Security Server and eNetwork Communications Server for OS/390. The configuration of the VPN was entirely manual using UNIX commands. The RFCs supported are the original IPSec standards as described in RFCs 1825 through 1829.

Note: Firewall Technologies were available as a *kit* in OS/390 V2R4 until the release of OS/390 V2R5.

At CS for OS/390 V2R5 IP the IPSec support was for manual tunnels, meaning that the key values remain static throughout the life of the tunnel. As you have seen in 2.3, “Virtual private network (VPN) and IPSec” on page 29, there are two operational modes in support of VPNs: those operating in *tunnel mode* and those operating in *transport mode*. At CS for OS/390 V2R5 IP, tunnel mode only was supported. In tunnel mode the whole IP packet is encapsulated in a new IP packet. The new IP header contains the two encryption peers performing encryption on behalf of the addresses in the original header. Tunnel mode allows also data from other resources (for example, IP addresses) to be encrypted. In OS/390 Firewall terminology tunnel mode is called an IPSec tunnel running in tunnel mode. See Figure 152.

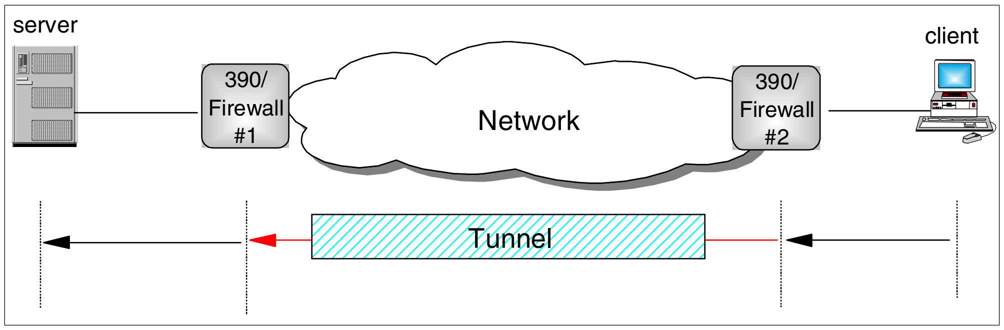


Figure 152. IPSec tunnel mode: protection between two IPSec endpoints

Here we see a typical example of an IPSec tunnel running in tunnel mode. The endpoints of the actual connection may lie beyond these endpoints as depicted in Figure 152. In tunnel mode the original IP packet is encapsulated in a new IP header and trailer.

With manual tunnels, the Security Association (SA) and cryptographic keys are generated on one of the hosts (the tunnel owner), transferred to the other host (the tunnel partner) with an out-band transport mechanism, and then imported. This procedure needs to be repeated whenever the validity of the keys has expired and new cryptographic keys need to be generated.

6.2.2 OS/390 V2R6

Recall from 2.3, “Virtual private network (VPN) and IPSec” on page 29 that IPSec tunnels have two modes, transport mode and tunnel mode. CS for OS/390 V2R6 IP provides transport mode.

It is important to know that transport mode is only supported for data flowing between the two encrypting or IPSec endpoints. The endpoints of the actual connection must lie at these endpoints. In transport mode only the protocol type of the original IP header is changed indicating that the packet is now under control of the IPSec protocol. In transport mode the payload is protected end-to-end (for example, OS/390 to client) as shown in Figure 153.

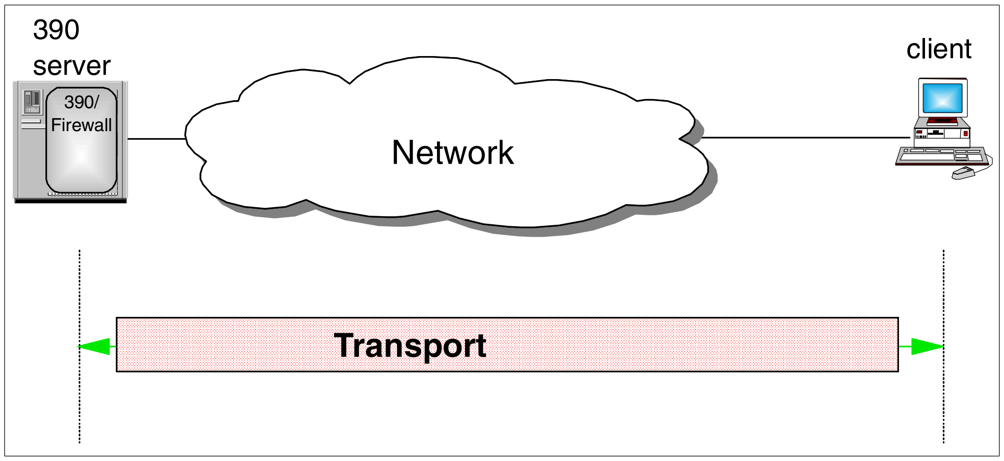


Figure 153. IPSec transport mode: end-to-end payload protection

6.2.3 OS/390 V2R7

OS/390 V2R7 brought with it the following enhancements to security In IPSec of Communications Server and in OS/390 Security Server:

- Incorporation of the IP Security RFCs 2401 through 2406 and RFC 2410. These RFCs added HMAC_MD5 and HMAC_SHA authentication, Triple-DES and ESP_NULL encryption, ESP Combined encryption/authentication, and Replay Prevention
- A configuration client GUI supported by the configuration server in Firewall Technologies

It is probably unrealistic to think that somebody can configure a large-scale IPSec network using the command-line facilities available under UNIX System Services. Therefore the use of the configuration client is recommended. For security the client uses SSL protocol, provided by the OS/390 Cryptographic Services System SSL, which must be configured on your machine. See Figure 154 on page 258.

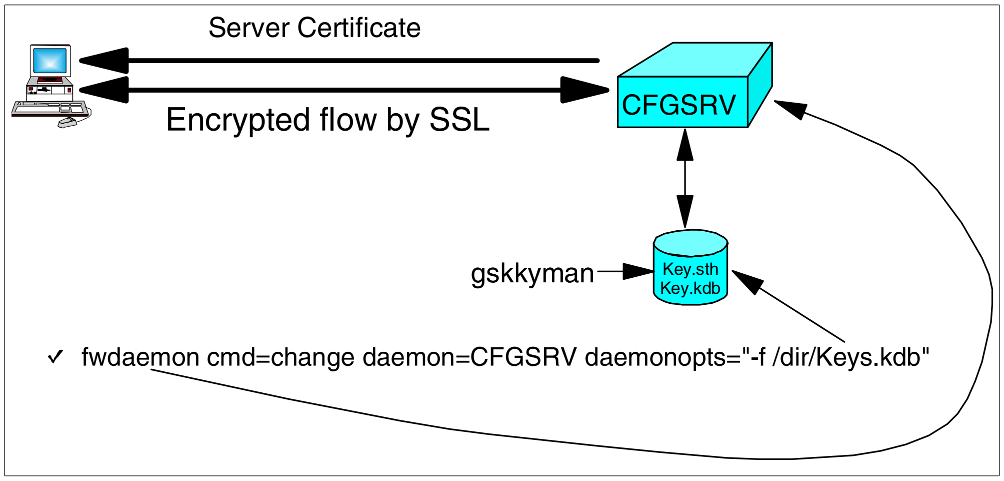


Figure 154. SSL between GUI client and firewall configuration program

The GUI is a Java-based firewall configuration client that runs on AIX, Windows NT 4.0, or Windows 95. The client, communicating with the CFGSRV task of the

Firewall, uses server-side SSL for encryption. The GSKKYMAN utility is used to create the key database and a self-signed certificate at OS/390; the key database is referenced at CFGSRV startup in the `fwdaemon` command. The certificate is used only to set up an SSL connection; it is not used for server authentication. SSL with client authentication is not provided. CFGSRV requests a valid RACF user ID and password for sign-on to the configuration server. In addition, the user must have update access to the RACF class ICA.CFGSRV. The GUI client can support more than one firewall stack.

6.2.4 OS/390 V2R8

In previous releases the Firewall administrator manually configures each system with its own keys and with keys of other communicating systems. At the older levels, if keys must be changed, the VPN tunnel must be recycled with new keys. These operations are practical for small, relatively static environments. In addition the administrator must define the encryption types and IP addresses for which encryption applies. This is a tedious and error-prone task.

At the heart of OS/390 V2R8 are security and systems management features. These include:

1. The ability to manage VPN encryption keys dynamically through the IKE protocol and its OS/390 Security Server ISAKMP daemon, and
2. Enhanced management and administration of digital certificates that represent IP hosts.

To implement a Dynamic VPN, you need OS/390 and the following products and/or subsystems:

- OS/390 V2R8
- SecureWay Communications Server for OS/390 V2R8 (with Encryption Features)
- OS/390 Firewall Technologies V2R8, which is implemented partly in the OS/390 Security Server and partly in the SecureWay Communications Server for OS/390
- OS/390 V2R8 Open Cryptographic Services Facilities (OCSF), that is required for ISAKMP only
- OS/390 V2R8 Security Server with RACF enabled, or an equivalent External Security Manager, for the security management in the OS/390 and UNIX System Services environment.

Related RFCs:

- *RFC - 2401 Security Architecture for the Internet Protocol* (Obsoletes RFC 1825)
- *RFC 2402 - IP Authentication Header* (Obsoletes RFC 1826)
- *RFC 2403 - The Use of HMAC-MD5-96 within ESP and AH*
- *RFC 2404 - The Use of HMAC-SHA-1-96 within ESP and AH*
- *RFC 2405 - The ESP DES-CBC Cipher Algorithm With Explicit IV*
- *RFC 2406 - IP Encapsulating Security Payload (ESP)* (Obsoletes RFC1827)
- *RFC 2407 - The Internet IP Security Domain of Interpretation for ISAKMP*
- *RFC 2408 - Internet Security Association and Key Management Protocol (ISAKMP)*
- *RFC 2409 - The Internet Key Exchange (IKE)*
- *RFC 2410 - The NULL Encryption Algorithm and Its Use With IPSec*

In Release 8, the exchange of encryption keys between the endpoints of an IPSec VPN can be automated and dynamically managed through Internet Key Exchange (IKE), an industry-accepted IPSec protocol for cryptographic key and security management. The key exchange is based on Diffie-Hellman. See 2.1, “Basic concepts of cryptography and digital certificates” on page 11 and 2.3, “Virtual private network (VPN) and IPSec” on page 29. Furthermore remote mobile hosts whose IP addresses are unpredictable and which previously could not take advantage of VPNs as tunnel partners themselves may now be identified with a host name or e-mail address if the IKE protocol is in use.

Using the IKE protocol IPSec peers are now able to create security associations and cryptographic keys dynamically. In general the IKE protocol allows encrypting peers to create keying material. As a result there is no need to configure each tunnel manually. IKE will take care of this. Of course everything comes with a price, with IKE support you now must create authentication information for the IKE peers. To validate the identity of the IKE servers using RSA Signature mode, each of the endpoints must have a certificate validated by a Certificate Authority. The Certificate Authority could be a known CA, or an OS/390 self-signed CA. One of the benefit to using certificates for authentication is the ensuing reduction in the amount of manual definitions. It is also possible to use pre-shared keys instead of certificates for authentication. Pre-shared keys require more manual configuration at each IP host.

With OS/390 Firewall Technologies as a component of Security Server in OS/390 V2R8, an automated system enables dynamic creation of keys and encryption types. This process facilitates the use of keys in large distributed environments with an evolving configuration and permits the creation of dynamic VPNs or tunnels. See Figure 155 for an illustration of manual tunnels versus dynamic tunnels.

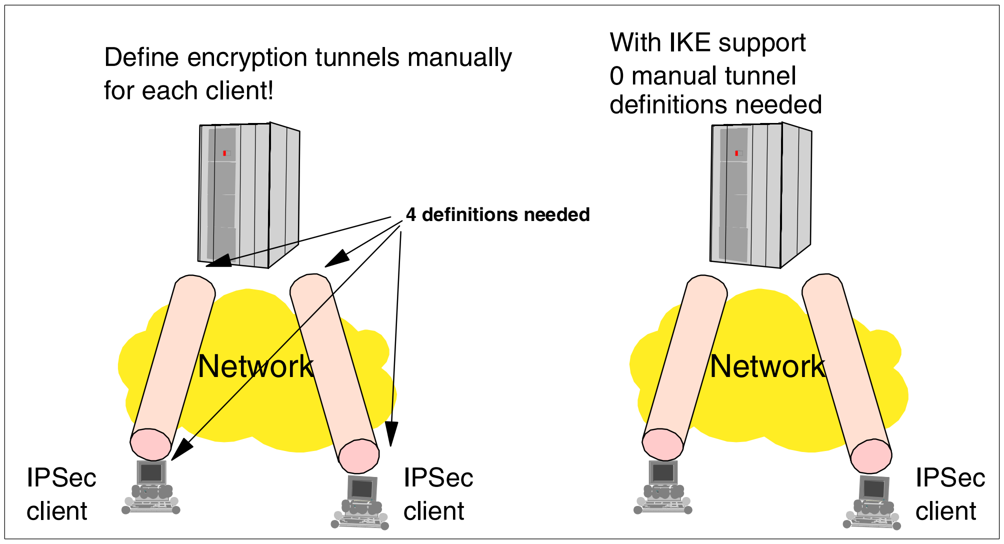


Figure 155. IKE advantages

The default automated key management protocol for IPSec is referred as the Internet Security Association Key Management Protocol (ISAKMP). ISAKMP provides a framework for negotiating Security Associations (SAs) and keying material. It defines the procedures and packet formats to establish, negotiate, modify and delete SAs, but it does not provide any specific key generation techniques or cryptographic algorithms.

The Internet Key Exchange (IKE) protocol has been designed based upon the ISAKMP protocol. IKE securely negotiates security associations and creates authenticated keying material for security associations. IP hosts that support IKE can negotiate Virtual Private Networks (VPNs) as shown in Figure 155.

6.2.4.1 Dynamic vs. manual tunnels

Although dynamic tunnels are preferable due to their inherent security strengths with the dynamic refresh of keys, there are times when a manual tunnel is still necessary. For example, dynamic VPNs do not support doing IPSec with the old RFC headers and they do not support doing IPSec with the CDMF encryption algorithm. The following encryption algorithms can be used to encrypt/decrypt the data:

- DES_CBC_8: Data Encryption Standard level of encryption (uses a 56-bit key and a 64-bit initialization vector)
- 3DES_CBC: Data Encryption Standard level of encryption (executed three times using a 24-byte key and a 32-bit initialization vector)
- ESP_NULL: No encryption

In addition, some platforms do not support dynamic VPNs and rely on the old RFC headers. In such scenarios the advanced security technologies provided by dynamic tunnels, the new RFCs, and the associated replay protection are useless and it is better to have some other security methods rather than none at all.

6.2.5 OS/390 V2R10

On-demand tunnels are introduced for IPSec in CS for OS/390 V2R10 IP. Dynamic tunnels (introduced in CS for OS/390 V2R8 IP) permitted the dynamic update and exchange of key materials as well as the dynamic enablement of one end of the VPN tunnel; however they also required that the tunnel be available prior to startup of any connections requiring its use. On-demand tunnels take advantage of dynamic tunnels in a different fashion. They allow a VPN to be created when outbound traffic flow demands it.

6.3 OS/390 firewall components

Figure 156 illustrates the individual servers that play a role in the OS/390 firewall.

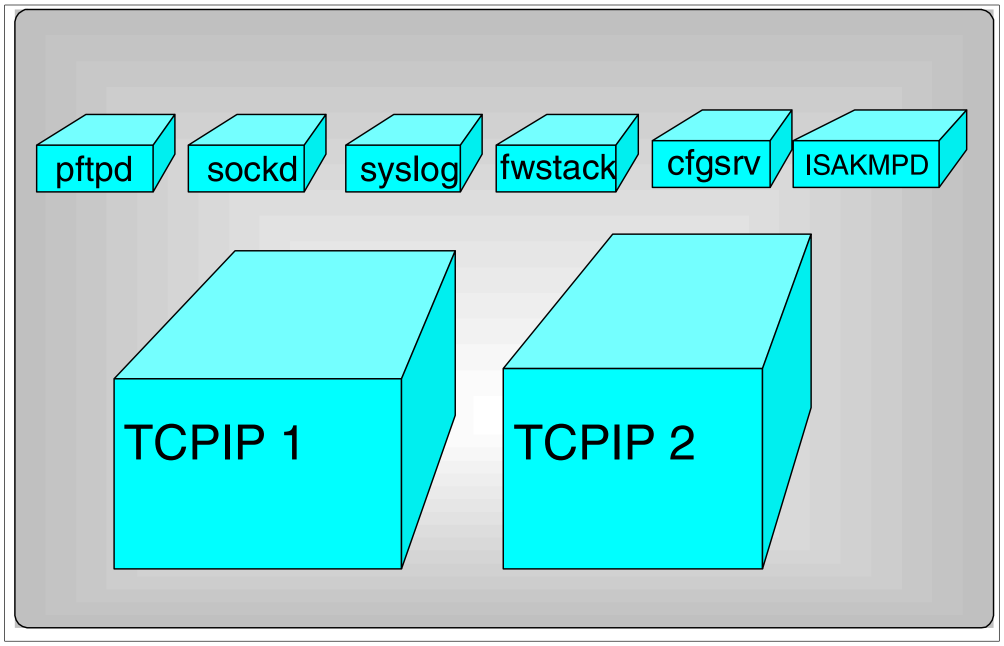


Figure 156. OS/390 firewall components

The FWKERN job of OS/390 normally starts all servers. Use the `fwdaemon` command in the UNIX System Services environment to alter and view server parameters. This command also allows you to stop and start servers.

The FTP proxy server (pftpd) and the SOCKS server (sockd) provide access from non-secure networks to secure networks. Both servers control the exchange of data between the two networks by acting as a relay between the two networks.

Prior to V2R10 a special syslogd shipped with OS/390 Firewall Technologies to provide SMF logging. Since V2R10 the SMF logging capability has been introduced into the Communication Server's syslogd and a separate Firewall logging daemon is no longer shipped.

The fwstackd server monitors and communicates with firewall TCP/IP stacks. This includes IPSec initialization and packet filter logging. The fwstackd server is configured through `fwdaemon` and started by FWKERN.

The CFGSRV task uses the SSL protocol to communicate with a configuration client. The configuration client in OS/390 V2R10 is a Java based application running on AIX and Windows 95/NT.

The ISAKMPD server implements the IKE protocol which is running over the UDP port 500 to communicate with other servers or hosts, that are usually referred as IKE peers or IPSec peers. It implements the protocols defined in RFC 2407, RFC 2408, and RFC 2409 to negotiate dynamic tunnel attributes, dynamically manage VPN encryption keys, and to establish dynamic tunnels via the `fwdynconns` command.

6.4 Screening and IP filters

The term *filtering* is usually associated with mechanisms that examine the IP and TCP headers for criteria like IP address or TCP port in order to determine whether service should be granted or denied; typically the term is applied to IP filtering performed at the IP layer of the protocol stack. Other mechanisms also exist that screen connections and packets for access. The screening criteria may be based on userid, IP stack name, TCP port number, or IP network affinity, as we have already seen in Chapter 4, “TCP/IP stack security” on page 91. Alternatively the screening criteria may revolve around IP protocol type, port number, IP addresses, direction of traffic flow, adapter, and date and time, as we have seen in the description of policy agent in 5.10, “OS/390 policy agent security” on page 202. These same mechanisms (protocol, port number, address, direction, etc.) play a role in the IP filtering which is a function of Firewall Technologies.

See Figure 157 for a diagram depicting the relationship of selected technologies to the concept of screening as a whole. The Venn diagram is not meant to be a comprehensive illustration of all the various ways of screening for access to the IP stack, the network, and their resources. If it were all-inclusive, we would need to include many other RACF and UNIX facilities, which would go beyond the intent of this section on Firewall Technologies and IPSec.

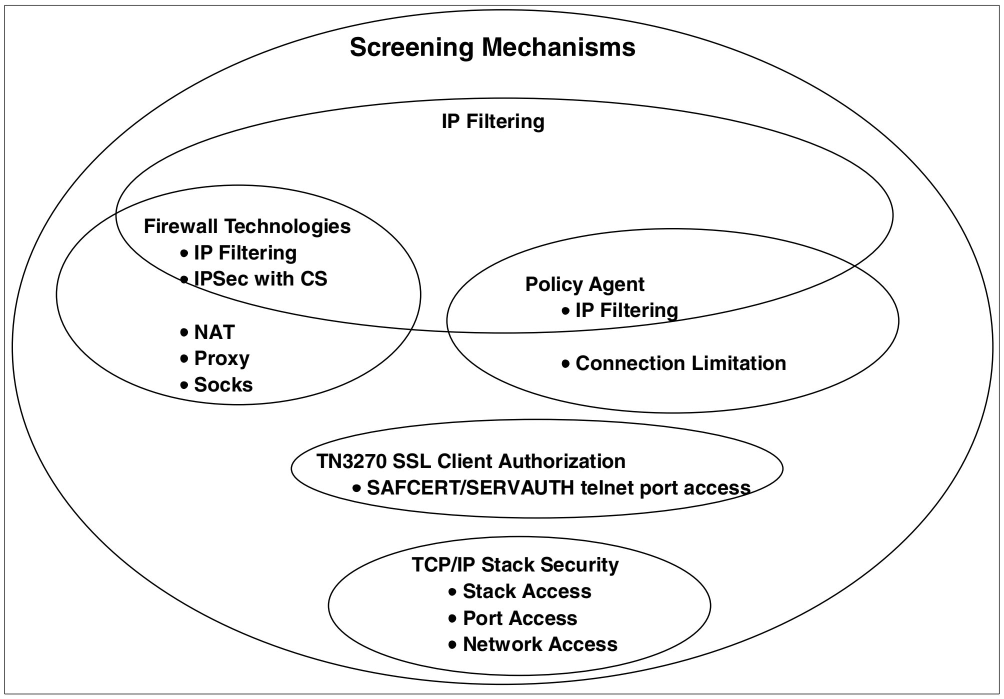


Figure 157. Selected screening mechanisms in CS for OS/390 IP

The diagram shows how certain capabilities of Firewall Technologies may be considered as IP filtering techniques; others are just methods to screen traffic. Policy agent has an IP filtering capability but it also supports a way to limit connections, which might be called a general screening technique. TN3270 SSL SERVAUTH port access and the TCP/IP stack security in general could be considered screening mechanisms, although they do not technically perform IP filtering. All of these various ways of providing screening may work in conjunction with each other in order to build multiple lines of defense. However, note in the following table what the unique differences are:

Table 23. Characteristics of screening and packet filtering technologies

Screening Mechanisms	**Packet Origination Point**	**Packet Terminatio n Point**	**Routed Packet**	**Advantages**
TCP/IP Stack Security Stack Access Port Access Network Access Telnet SERVAUTH	Yes	Yes	No	Simple, quick implementation
Policy Agent	Yes	Yes	No	Simple, quick connection blocking and connection limitation
Firewall Technologies	Yes	Yes	Yes	Robust filtering, including for forwarded or routed packets

In all these various methods, any flow that does not meet the defined rule is discarded. However a firewall may examine all IP packets: those in transit (*routed* or *forwarded* packets) and those both originating and terminating at the firewall node. See Figure 158.

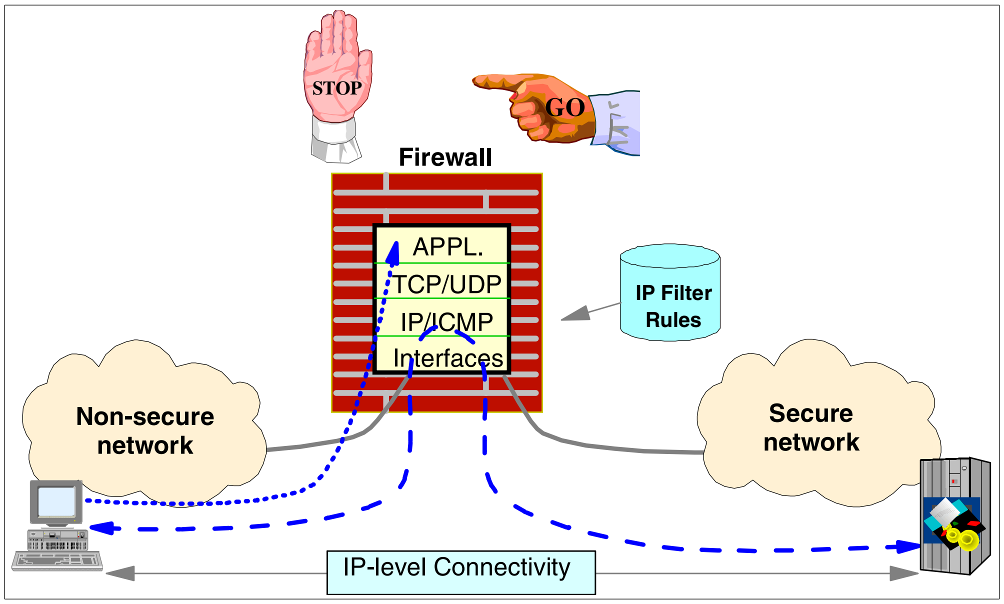


Figure 158. Firewall filtering

6.4.1 Defining IP filters

The screening provided by a firewall is used to permit or deny access to applications at the firewall or to those residing on either side of it. See Figure 159.

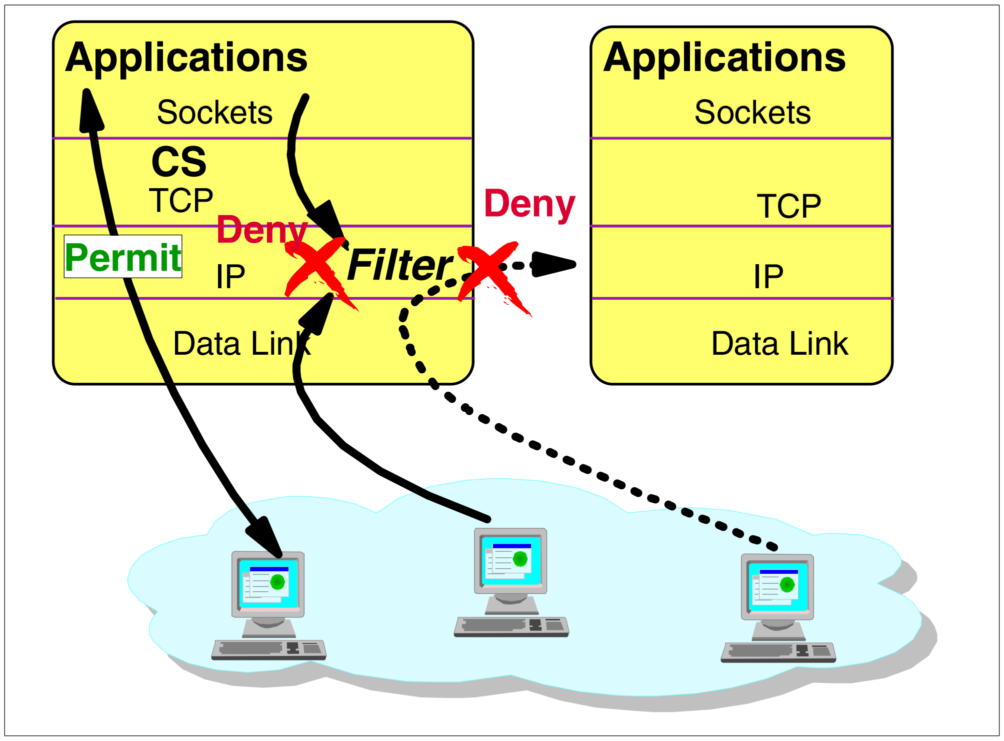


Figure 159. IP filtering with OS/390 Firewall Technologies

Suppose, for example, that you want to restrict TN3270 traffic inbound from and outbound to a specific IP network as illustrated in Figure 160.

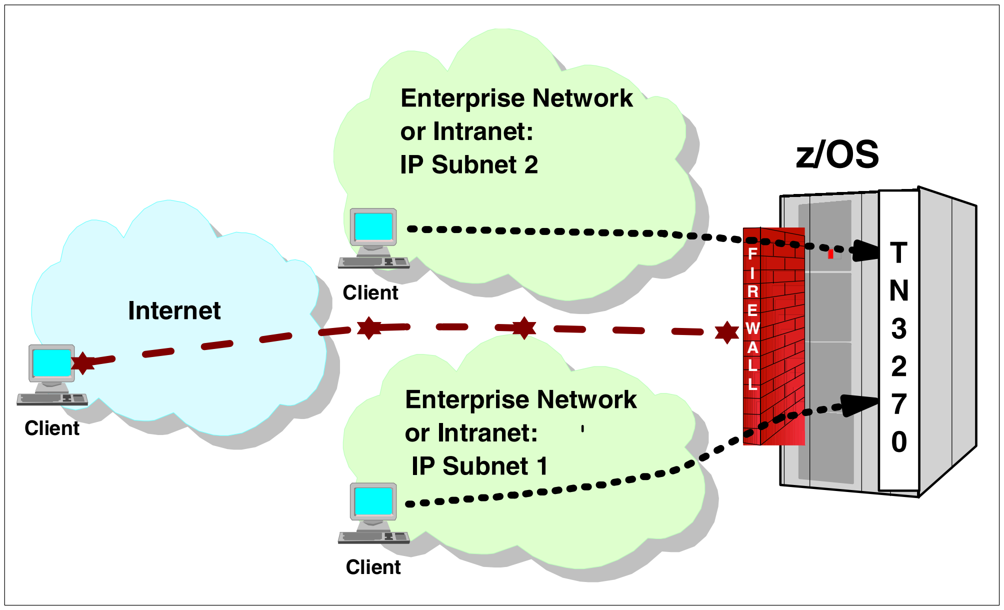


Figure 160. IP filtering example

As illustrated, anyone belonging to the corporate network should be allowed to telnet into z/OS. Anyone on the Internet should be prohibited from doing so. The interior firewall filtering capability of Firewall Technologies in z/OS examines packets to determine which ones are allowed to continue. Since we are using only an interior firewall capability, note that z/OS cycles are consumed in the process of examining traffic and determining its eligibility to continue. Some company sites might wish to combine an exterior firewall to deflect the cycles that might arrive at z/OS and then use the interior firewall as a second line of defense.

In either case, if you were to define filters with OS/390 Firewall Technologies you would define four object types to create such a screening filter, as depicted in Figure 161.

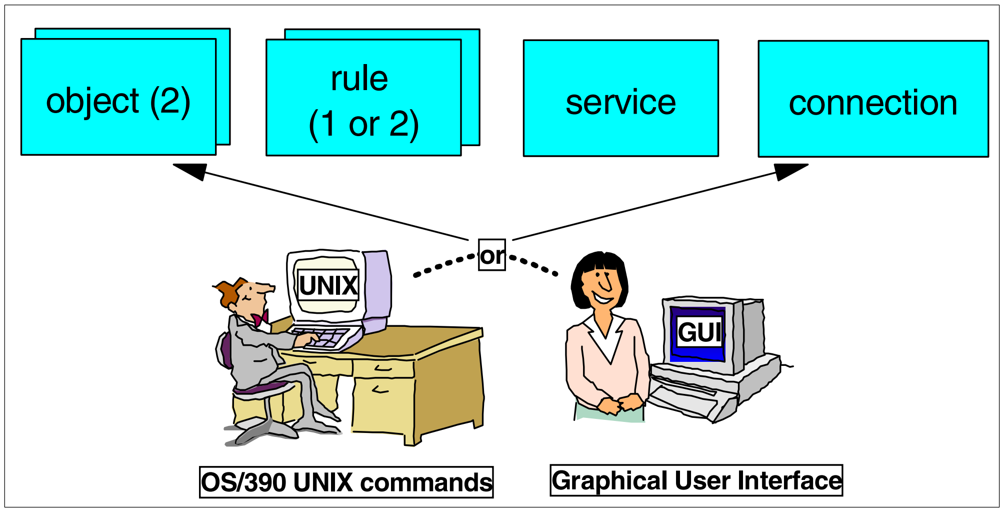
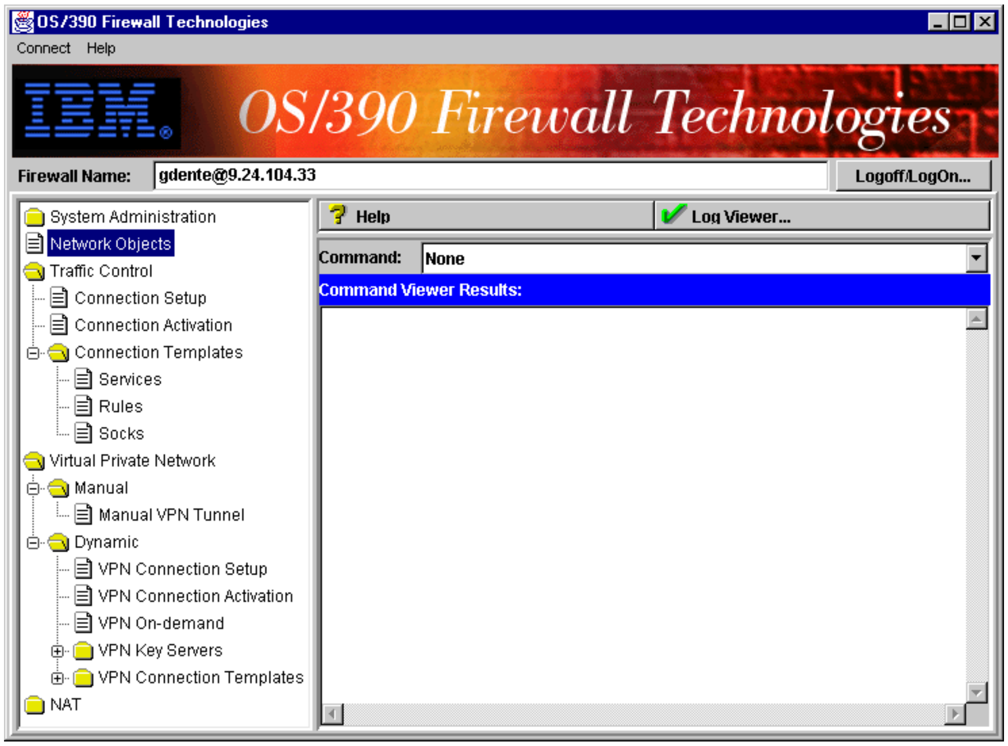


Figure 161. Basic firewall technology filter definition

Object	A network object defines a subnet, single IP address, or an IP address range. There is usually an object for the OS/390 (local) side of the connection and another object for the remote end of the connection. The UNIX command to create an object is: `fwnwobj`.
Rule	A rule defines a permit or deny for some type of higher lever IP protocol. You should define a rule for inbound traffic and a rule for outbound traffic or a single rule that is valid for both directions. The UNIX command `fwfrule` defines this.
Service	A service defines a set of rules. In fact a service is a set of allowed protocols between some source and destination IP address. The UNIX command `fwservice` creates this.
Connection	A connection defines a set of services. A connection defines which services are allowed between two IP addresses/subnets. The UNIX command `fwconns` defines this function.

Figure 162 displays the window that may be used to define these four components of a filter.



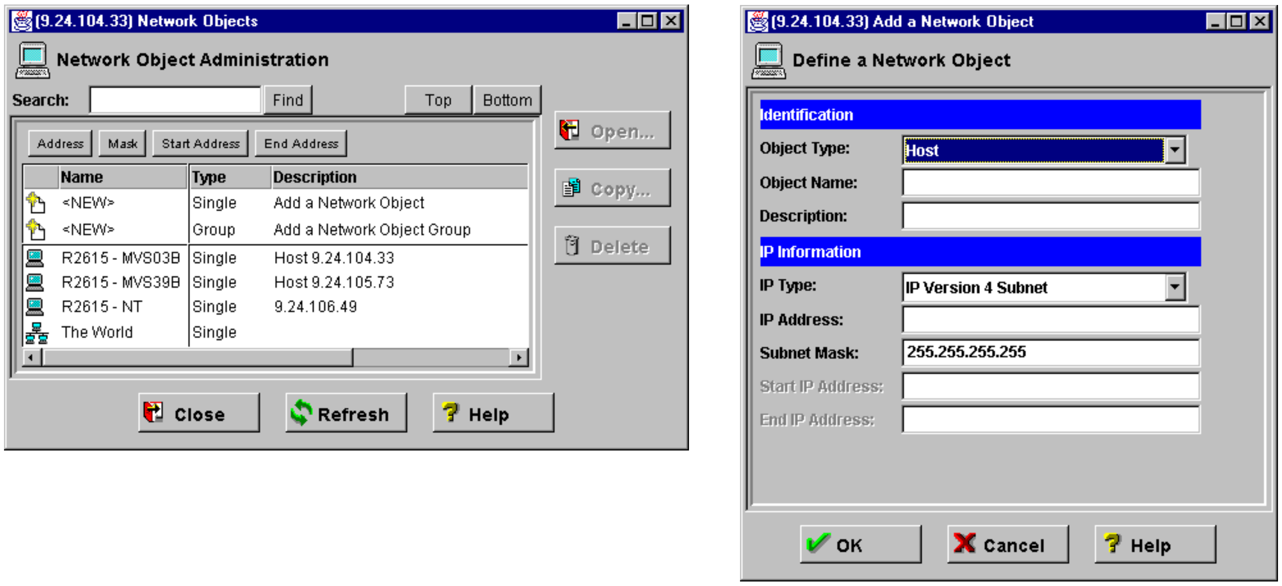
Firewall Name: gdente@9.24.104.33

Command: None

Figure 162. Firewall technologies configuration GUI

Note how objects are accessible under "Network Objects", how services and rules are defined under the category "Connection Templates" and how the connections are defined under "Traffic Control/Connection Setup".

Figure 163 displays the windows that may be used to list existing network objects and then to define the object for the filter.

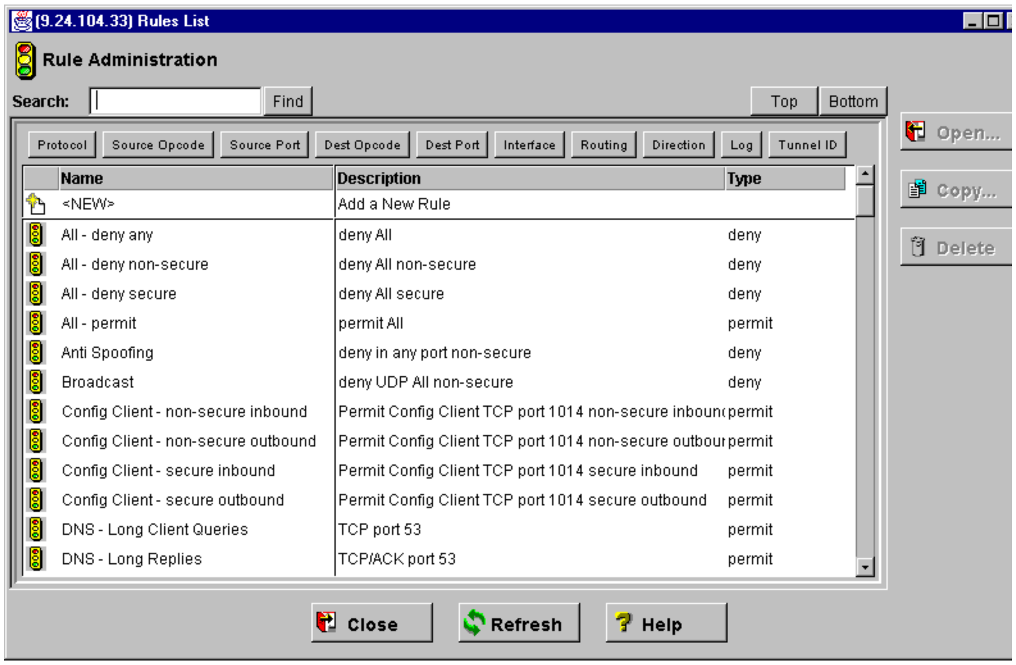


Name	Type	Description
<NEW>	Single	Add a Network Object
<NEW>	Group	Add a Network Object Group
R2615 - MVS03B	Single	Host 9.24.104.33
R2615 - MVS39B	Single	Host 9.24.105.73
R2615 - NT	Single	9.24.106.49
The World	Single	

Object Type:	Host
Object Name:	
Description:	
IP Type:	IP Version 4 Subnet
IP Address:	
Subnet Mask:	255.255.255.255
Start IP Address:	
End IP Address:	

Figure 163. Firewall object list and definition with the configuration GUI

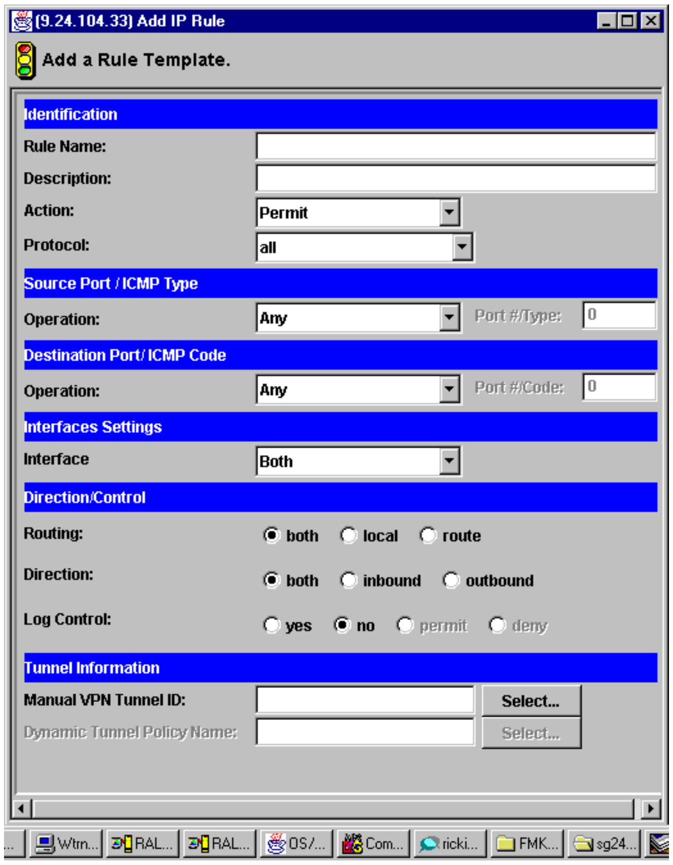
Figure 164 displays the existing rules that have been defined for this firewall.



Name	Description	Type
<NEW>	Add a New Rule	
All - deny any	deny All	deny
All - deny non-secure	deny All non-secure	deny
All - deny secure	deny All secure	deny
All - permit	permit All	permit
Anti Spoofing	deny in any port non-secure	deny
Broadcast	deny UDP All non-secure	deny
Config Client - non-secure inbound	Permit Config Client TCP port 1014 non-secure inbounc	permit
Config Client - non-secure outbound	Permit Config Client TCP port 1014 non-secure outbour	permit
Config Client - secure inbound	Permit Config Client TCP port 1014 secure inbound	permit
Config Client - secure outbound	Permit Config Client TCP port 1014 secure outbound	permit
DNS - Long Client Queries	TCP port 53	permit
DNS - Long Replies	TCP/ACK port 53	permit

Figure 164. Firewall GUI list of existing rules

Figure 165 shows how a rule may be defined that would subsequently be listed as in Figure 164.



(9.24.104.33) Add IP Rule

Add a Rule Template.

Identification

Rule Name:	
Description:	
Action:	Permit
Protocol:	all

Source Port / ICMP Type

Operation:	Any	Port #/Type:	0

Destination Port/ ICMP Code

Operation:	Any	Port #/Code:	0

Interfaces Settings

Interface	Both

Direction/Control

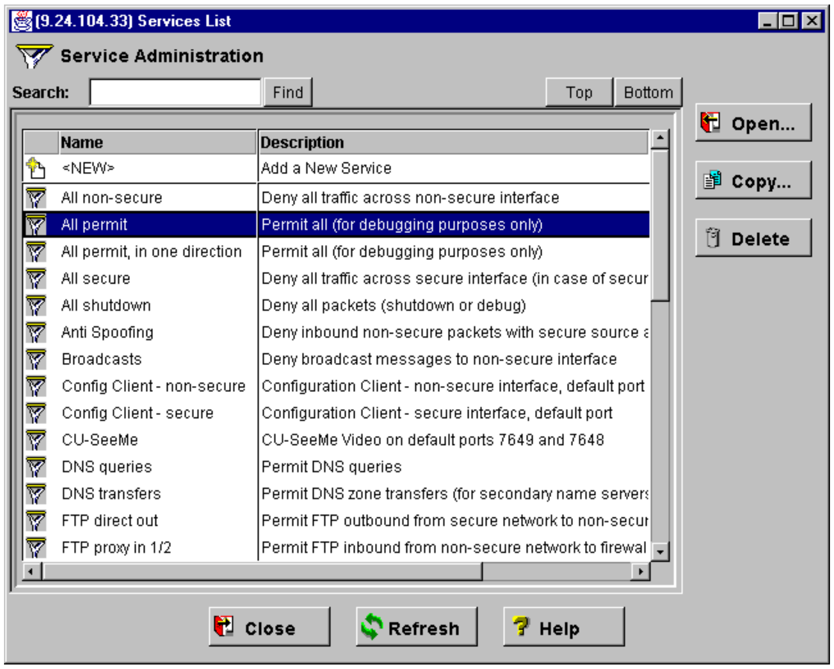
Routing:	[✓ both] local route
Direction:	[✓ both] inbound outbound
Log Control:	yes [✓ no] permit deny

Tunnel Information

Manual VPN Tunnel ID:		Select...
Dynamic Tunnel Policy Name:		Select...

Figure 165. Firewall GUI rule definition

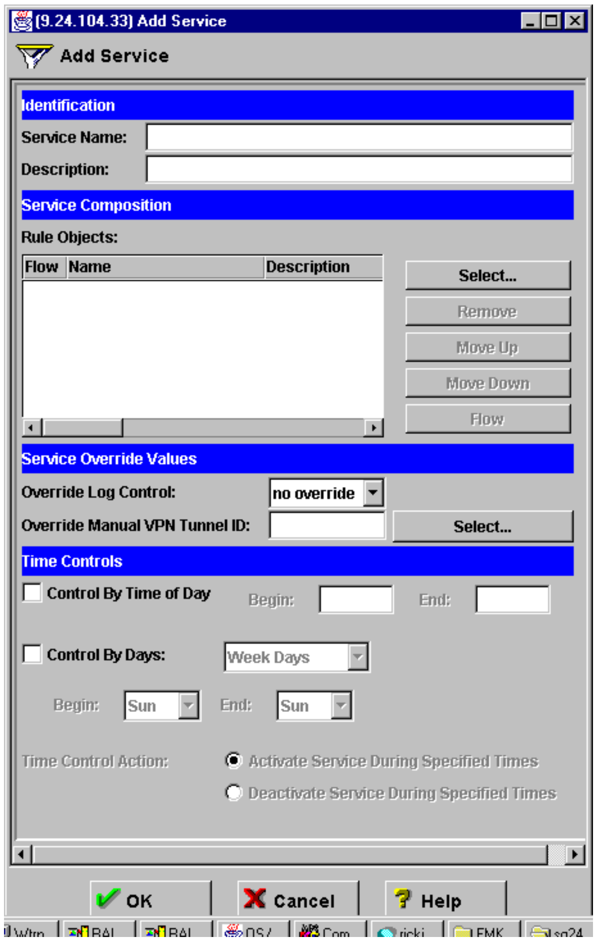
Next we see the list of services that have been created in this firewall implementation: Figure 166.



Name	Description
<NEW>	Add a New Service
All non-secure	Deny all traffic across non-secure interface
All permit	Permit all (for debugging purposes only)
All permit, in one direction	Permit all (for debugging purposes only)
All secure	Deny all traffic across secure interface (in case of secur
All shutdown	Deny all packets (shutdown or debug)
Anti Spoofing	Deny inbound non-secure packets with secure source a
Broadcasts	Deny broadcast messages to non-secure interface
Config Client - non-secure	Configuration Client - non-secure interface, default port
Config Client - secure	Configuration Client - secure interface, default port
CU-SeeMe	CU-SeeMe Video on default ports 7649 and 7648
DNS queries	Permit DNS queries
DNS transfers	Permit DNS zone transfers (for secondary name servers
FTP direct out	Permit FTP outbound from secure network to non-secur
FTP proxy in 1/2	Permit FTP inbound from non-secure network to firewal

Figure 166. Firewall GUI display of created services

If you want to add a service to the firewall configuration, you can do so easily with the window shown in Figure 167.


[9.24.104.33] Add Service

Add Service

Identification

Service Name:

Description:

Service Composition

Rule Objects:

Flow	Name	Description

Service Override Values

Override Log Control: no override

Override Manual VPN Tunnel ID:

Time Controls

Control By Time of Day Begin: End:

Control By Days: Week Days

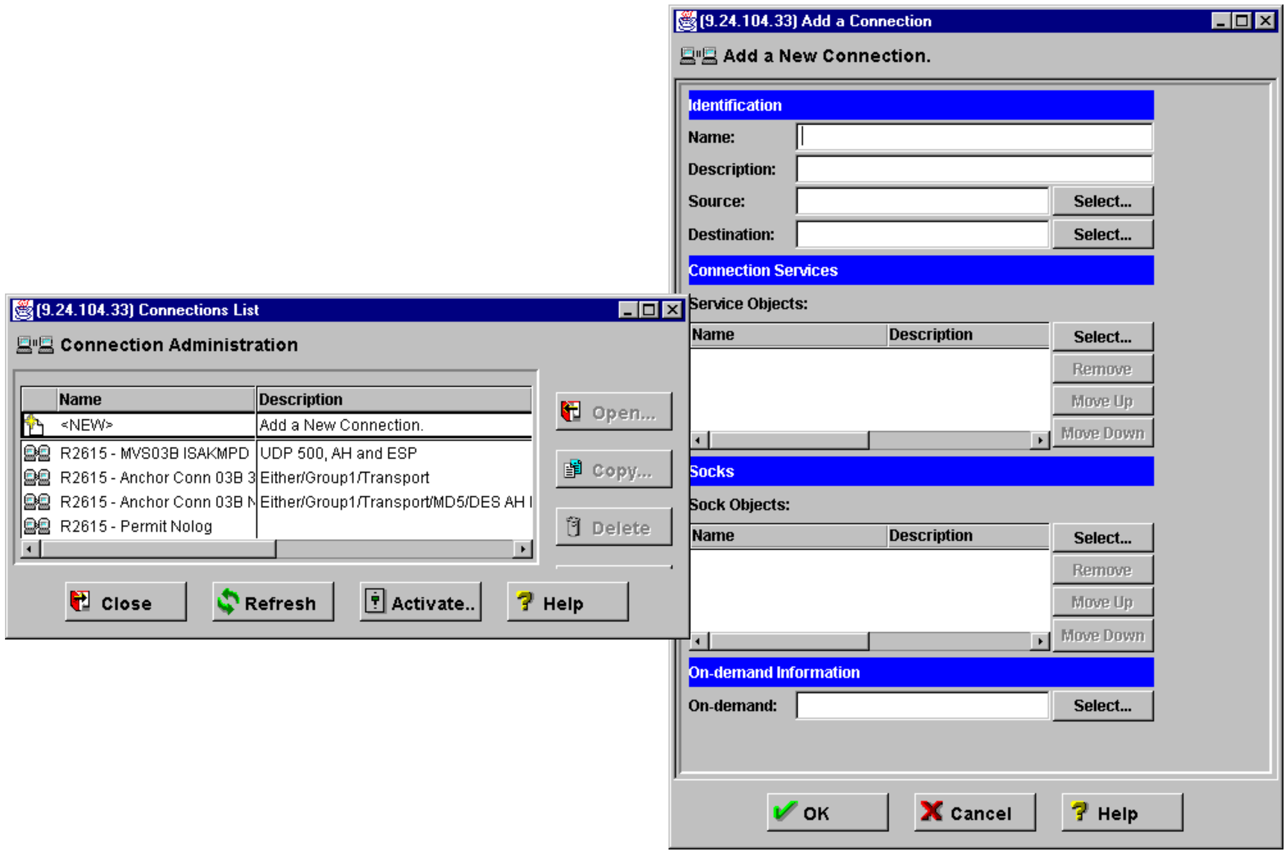
Begin: Sun End: Sun

Time Control Action: [✓ Activate Service During Specified Times] Deactivate Service During Specified Times

Figure 167. Firewall GUI definition of new service

Finally, Figure 168 on page 272 shows how to list, administer, and add new connections using the GUI.

The order of connection is important because the first matched rule is used. The IKE connection object has to be located before anchor connections.



Name	Description
<NEW>	Add a New Connection.
R2615 - MVS03B ISAKMPD	UDP 500, AH and ESP
R2615 - Anchor Conn 03B 3	Either/Group1/Transport
R2615 - Anchor Conn 03B N	Either/Group1/Transport/MD5/DES AH I
R2615 - Permit Nolog	

Name:
Description:
Source:
Destination:
Service Objects:
Name | Description
Sock Objects:
Name | Description
On-demand:

Figure 168. Firewall GUI display of existing connections and definition of new connection

6.5 IPSec in Firewall Technologies

In the last sections you saw how to use Firewall Technologies in the Security Server to define filters that permit or deny traffic leaving, arriving at, or in transit through z/OS. In this section we look at the various ways VPNs may be implemented and then we proceed to an examination of how VPNs (or tunnels) are defined.

A white paper entitled *VPN Interoperability Demonstrated for OS/390* by Linwood Overby of z/OS Communications Server Strategy and Design and Thomas Cosenza of z/OS Communications Server System Test is available at:

```
www.ibm.com/servers/eserver/zseries/networking/pdf/GM13-0029-00.pdf
```

It describes how OS/390 IPSec interoperates in a variety of configuration scenarios with the IRE SafeNet/Soft-PK VPN Client in Windows NT and Windows 95, the integrated VPN client in Windows 2000, and Cisco security gateways.

6.5.1 Virtual private network (VPN): Overview of business scenarios

The z/OS system, through the IPSec protocol, provides any or all of the following services:

- Host authentication using digital signatures
- Security key exchange

- Data encryption, data origin authentication, and message integrity using symmetric cryptography

The IP security support uses several functions available in the S/390 hardware for encrypting/decrypting as well as authenticating TCP/IP packet data in an IP tunnel. This support is provided by the combination of the Integrated Cryptographic Feature (ICRF) on the processor and the Integrated Cryptographic Service Facility/MVS (ICSF/MVS) software product. If these options are not available, the encryption and decryption will be performed by the eNetwork Communications Server software.

An IPSec Security Association (SA) corresponds to a session between two hosts. It defines the set of protocols and, with these, the negotiated algorithms and keys that are to be used when transmitting data between two hosts. An SA for data traffic is always unidirectional, so for a pair of hosts that are to communicate securely, at least two SAs, one for each direction, are needed. This differs from other protocols that make use of sessions such as, for instance, SSL: an SSL session covers the transmission in both directions.

Like IP packet filtering, IPSec is transparent to the end user; the administrator sets up the tunnel and the appropriate traffic flows across it. When an outbound packet matches on a filter rule which specifies a tunnel ID the packet is encapsulated according to the tunnel definition and sent to the partner firewall or IPSec node. When an inbound encapsulated packet is received by the partner firewall (or IPSec node), the tunnel definition is used to restore the packet to its original format in a process known as decapsulation. See Figure 169.

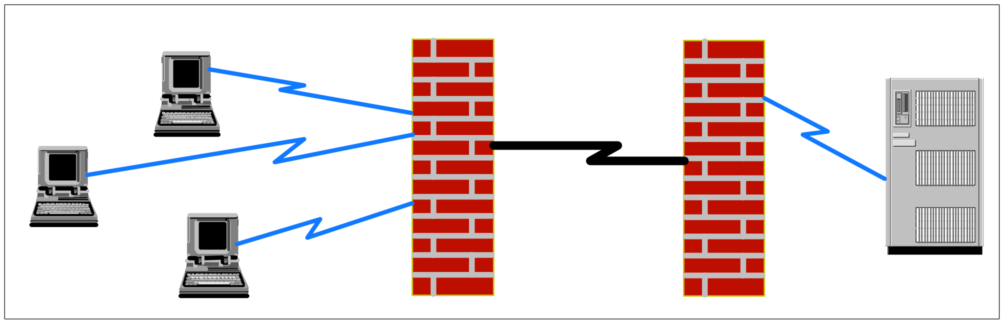


Figure 169. IPSec example

In this section we look at the three most likely business scenarios well suited to the implementation of a VPN solution:

1. Branch office connection network

2. Business partner/supplier network

3. Remote access network

The examples show the OS/390 system connected to an intranet, a demilitarized zone, or the Internet. In general, there are three types of VPN implementations that are well suited to most business needs. Determining how you will use your VPN is one of the first steps in successful planning.

Any of the scenarios can be built with manual tunnels or dynamic tunnels or even on-demand tunnels as long as both endpoints of the tunnel can support the desired technology and RFCs.

The questions that must ultimately be answered in order to create and define a strategy to protect your corporate data with IPSec are:

1. How do I want to protect data? (Manual, dynamic tunnel; encapsulation modes, data management?)
2. What data do I want to protect? (Networks, protocols, applications?)
3. How do I want to protect key exchanges? (IKE, encryption strength, etc.?)
4. Where do I want to protect the data? (Who should be the IPSec peers?)
5. How do I authenticate my IPSec peer? (Pre-shared keys, certificates?)

Again, we refer you to 2.1, “Basic concepts of cryptography and digital certificates” on page 11**,** 2.2, “Firewall concepts” on page 24**,** and 2.3, “Virtual private network (VPN) and IPSec” on page 29 for detailed information on IPSec.

6.5.1.1 Branch office connection network

The branch office scenario securely connects two trusted intranets within your organization. This is a key consideration, since your security focus is on both protecting your company′s intranet against external intruders and securing your company′s data while it flows over the public Internet. This differs from the business partner/supplier network discussed in 6.5.1.2, “Business partner/supplier network” on page 277, where the focus is on enabling your business partners’/suppliers’ access to data in your corporate intranet.

The branch office scenario allows secure communications between geographically separated branch office intranets that are part of a single corporate network. A branch office connection typically involves any of these three connection types:

1. **Connect your gateway to another gateway**
   - The connection endpoints of both systems are different from the data endpoints. The VPN protects traffic as it travels between the gateways. However, traffic on either side of the gateway within the internal networks is unaffected by the VPN. This is a common branch office connection because traffic that is routed beyond the gateway is often considered trusted.
2. **Connect your gateway to another host**
   - The VPN protects traffic as it travels from your gateway to the remote host machine. Traffic on your side of the gateway is unaffected by the VPN because you consider it trusted.
3. **Connect your host to another gateway**
   - The VPN protects traffic as it travels from your host machine to the gateway to which it is connecting. Traffic that is routed beyond the remote gateway is unaffected by the VPN.

For example, suppose corporate headquarters wants to minimize the costs incurred from communicating to and among its own branches. Today, the company may use frame relay and/or leased lines, but wants to explore other options for transmitting its internal confidential data that will be less expensive, more secure, and globally accessible. By exploiting the Internet, branch office connection VPNs can easily be established to meet the company's needs.

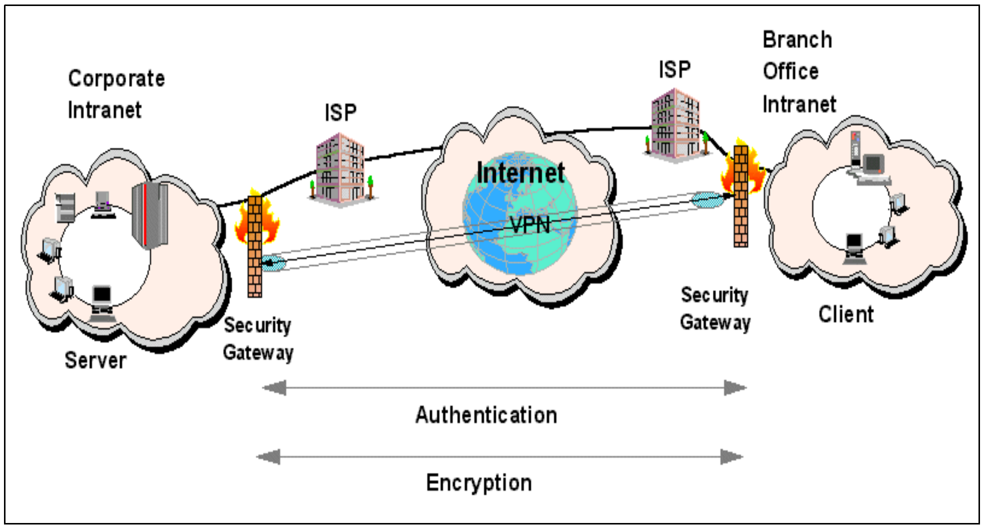


Figure 170. Branch office connection network

As shown in Figure 170, one way to implement this VPN connection between the corporate headquarters and one of its branch offices is for the company to purchase Internet access from an Internet service provider (ISP). IBM SecureWay firewalls, or routers with integrated firewall functionality, or in some cases an IBM server with IPSec capability, would be placed at the boundary of each of the intranets to protect the corporate traffic from Internet hackers. With this scenario, the clients and servers need not support IPSec technology, since the IPSec-enabled firewalls (or routers) would be providing the necessary data packet authentication and encryption. With this approach, any confidential information would be hidden from untrusted Internet users, with the firewall denying access to potential attackers.

With the establishment of branch office connection VPNs, the company's corporate headquarters will be able to communicate securely and cost-effectively to its branches, whether located locally or far away. Through VPN technology, each branch can also extend the reach of its existing intranet to incorporate the other branch intranets, building an extended, enterprise-wide corporate network.

And, as in the business partner/supplier network scenario, this company can easily expand this newly created environment to include its business partners, suppliers, and remote users, through the use of open IPSec technology.

This scenario has the following advantages:

- Using the Internet or an existing intranet reduces the cost of private lines between remote subnets.
- Using the Internet or an existing intranet reduces the complexity of installing and maintaining private lines and associated equipment.
- Using the Internet allows remote locations to be connected to almost anywhere in the world.

- Users can access to all servers and resources on either side of the VPN tunnel as though they were connected using a leased line or WAN connection.
- Industry-standard encryption and authentication methods secure sensitive information passed from one location to another.
- Encryption keys are exchanged dynamically and regularly to simplify setup and to minimize the risk of their being decoded and security being breached.
- Using private IP addresses in each remote subnet makes it unnecessary to allocate valuable public Internet addresses to each client.

In summary, a branch office has the following attributes:

- It is an extension of the corporate intranet.
- It is maintained in a geographically dispersed location.
- It is a trusted network.
- Network-to-network security is the main concern.

Planning considerations: branch office network

To set up a VPN that connects your main office to its branch offices, there are several things you need to consider. First, what level of security do you require? Obviously, you want to keep your data confidential as it travels across the public Internet, but do you need to protect data as it flows through the company intranet as well? Next, you need to understand what effect the VPN will have on system performance. For example, if you require maximum security from the VPN, then you can expect your system performance to be affected. Finally, you must develop a plan for implementing the VPN that includes information such as source and destination IP addresses (or some other means to identify your communicating systems), your authentication and encryption strategy and various other factors.

In this scenario, your company wants to establish a VPN between the corporate gateway and the gateway of a branch office. The first thing you should do is to gather the information you need to complete your configurations. The following planning checklists illustrate the type of information you will need to use.

In our case, we know that we are working with a trusted network, so we decide to protect keys ("key management") and to protect data ("data management") using a low security level in our VPN policies. Some implementers might choose to use a high security level for key management and a low security level for data management.

Table 24. Initial design worksheet for dynamic connections

This is the information needed to create dynamic VPNs	**Answers**
What is the type of connection to be created? - Gateway-to-gateway - Host-to-gateway - Gateway-to-host - Host-to-host - Gateway-to-dynamic IP user - Host-to-dynamic IP user	Gateway-to-gateway
What type of security and system performance is required to protect the keys? - Highest security, lowest performance - Balance security and performance - Lowest security and highest performance - Add your own definition	Lowest security and highest performance
What type of security and system performance is required to protect the data? - Highest security, lowest performance - Balance security and performance - Lowest security and highest performance - Add your own definition	Lowest security and highest performance

6.5.1.2 Business partner/supplier network

Industry-leading companies will be those that can communicate inexpensively and securely to their business partners, subsidiaries, and vendors. Many companies have chosen to implement frame relay and/or purchase leased lines to achieve this interaction. But this is often expensive, and geographic reach may be limited. VPN technology offers an alternative for companies to build a private and cost-effective extended corporate network with worldwide coverage, exploiting the Internet or other public network.

Suppose you are a major parts supplier to a manufacturer. Since it is critical that you have the specific parts and quantities at the exact time required by the manufacturing firm, you always need to be aware of the manufacturer′s inventory status and production schedules. Perhaps you are handling this interaction manually today, and have found it to be time consuming, expensive and maybe even inaccurate. You would like to find an easier, faster, and more effective way of communicating. However, given the confidentiality and time-sensitive nature of this information, the manufacturer does not want to publish this data on its corporate Web page or distribute this information monthly via an external report.

To solve these problems, the parts supplier and manufacturer can implement a VPN, as shown in Figure 171. A VPN can be built between a client workstation, in the parts supplier′s intranet, directly to the server residing in the manufacturer′s intranet. The clients can authenticate themselves either to the firewall or router protecting the manufacturer′s intranet, directly to the manufacturer′s server (validating that they are who they say they are), or to both, depending on your security policy. Then a tunnel could be established, encrypting all data packets from the client, through the Internet, to the required server.

An SA bundle can be used in this scenario. One SA between the two firewalls doing AH, and the other SA end to end using encryption and optionally authentication for the IP host authentication.

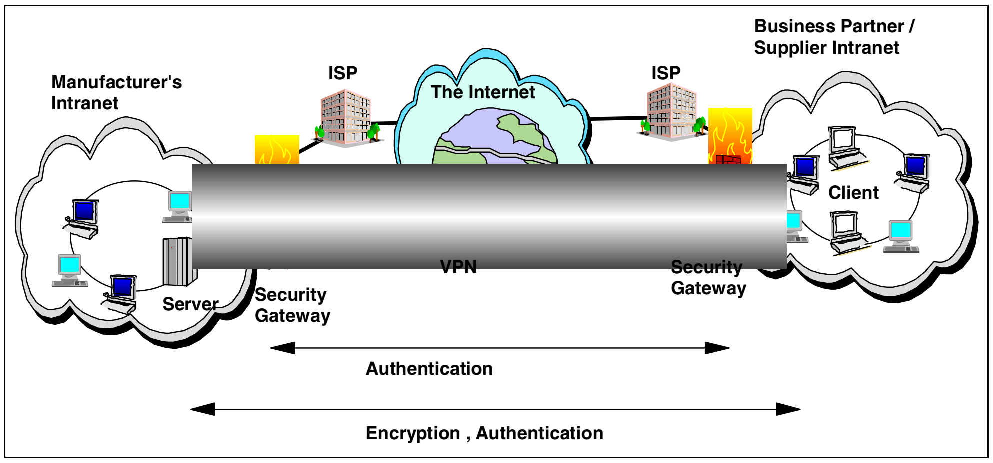


Figure 171. Business partner/supplier network

With the establishment of this VPN, the parts supplier can have global, online access to the manufacturer′s inventory plans and production schedule at all times during the day or night, minimizing manual errors and eliminating the need for additional resources for this communication. In addition, the manufacturer can be assured that the data is securely and readily available to only the intended parts supplier(s).

Again, this situation can be addressed by obtaining Internet service from an ISP. Then either an IBM firewall or IPSec-enabled router, or an IBM server with IPSec capability, can be deployed as required to protect the intranets from intruders. If end-to-end protection is desired, then both the client and server machines need to be IPSec-enabled as well.

Through the implementation of this VPN technology, the manufacturer would easily be able to extend the reach of its existing corporate intranet to include one or more parts suppliers (essentially building an extended corporate network) while enjoying the cost-effective benefits of using the Internet as its backbone.

And, with the flexibility of open IPSec technology, the ability for this manufacturer to incorporate more external suppliers is limitless.

This allows secure communications between your company's intranet and your business partner's intranet. For example, our scenario simulates secure communication between a parts supplier and a manufacturing business. A business partner connection typically involves connecting a host to another host. In a host-to-host connection, the VPN, operating in transport mode, protects traffic as it travels from your host machine to the remote host machine. This is a common business partner connection because traffic along all paths of the connection are often considered untrusted.

When a user wishes to send information to his counterpart or connect to an application, the user initiates the information exchange for ftp, telnet, or any other service. The IP stack at the source checks the tunnel definition and sets up the authentication header and/or encrypts the data, then encapsulates each packet.

Packets are sent across the network to the destination. At the destination, the IP stack decapsulates the packet, then decrypts and/or authenticates the data.The information is sent to the destination application or user.

In summary, a business partner scenario has the following attributes:

- Partners do not trust each other's intranet
- Concerns include host-to-host security and access to the private intranet
- VPN is implemented client to server, and if bundling gateway to gateway

Planning considerations: business partner network

To set up a VPN that connects your company to its business partners, there are several things you need to consider. First, what level of security do you require? Obviously, you want to keep your data confidential as it travels across the public Internet, but do you need to protect data as it flows through both company's intranets as well? Next, you need to understand what effect the VPN will have on system performance. For example, if you require maximum security from the VPN, then you can expect your system performance to be affected. Finally, you must develop a plan for implementing the VPN that includes information such as source and destination IP addresses (or some other means to identify your communicating systems), your authentication and encryption strategy, and various other factors.

In this scenario, your company wants to establish a VPN between an OS/390 host in your parts division and a client workstation in the manufacturing department of your business partner. The first step is to gather the information required to complete the definitions you need. The following planning checklists illustrate the type of information you will need.

In this case we are going to connect to an untrusted network, so we do not know how dangerous that could be. We choose to use the highest security and lowest performance specifications at the key management and data management levels. Table 24 on page 277 shows the main planning parameters and the key policy planning worksheet you might for such a scenario.

See Appendix F, “VPN planning worksheets” on page 439 for blank copies of these worksheets that you can use when doing your own planning.

6.5.1.3 Remote access network

A remote user, whether at home or on the road, wants to be able to communicate securely and cost-effectively back to his/her corporate intranet.

Although many still use expensive long-distance and toll-free telephone numbers, this cost can be greatly minimized by exploiting the Internet. For example, you are at home or on the road, but need a confidential file on a server within your intranet. By obtaining Internet access in the form of a dial-in connection to an ISP, you can communicate with the server in your intranet and access the required file.

The remote user dials in to his or her ISP by using whatever procedures and protocols the ISP requires. For example, many ISPs require remote hosts to use the Point-to-Point (PPP) protocol, and to identify themselves by an account name and a password. Then, the ISP typically assigns the remote user a dynamic IP address. Because the server to which the user wants to connect has no way of knowing his or her IP address in advance, the remote host must always initiate the connection.

One way to implement this scenario *securely* is to use a VPN IPSec-enabled remote client and firewall, as shown in Figure 172. The client accesses the Internet via dial-up to an ISP, and then establishes an authenticated and encrypted tunnel between itself and the firewall at the intranet boundary. With IKE, mobile users with shifting IP addresses can participate as IKE peers in a VPN.

By applying IPSec authentication between the remote client and the firewall, you can protect your intranet from unwanted and possibly malicious IP packets. And by encrypting traffic that flows between the remote host and the firewall, you can prevent outsiders from eavesdropping on your information.

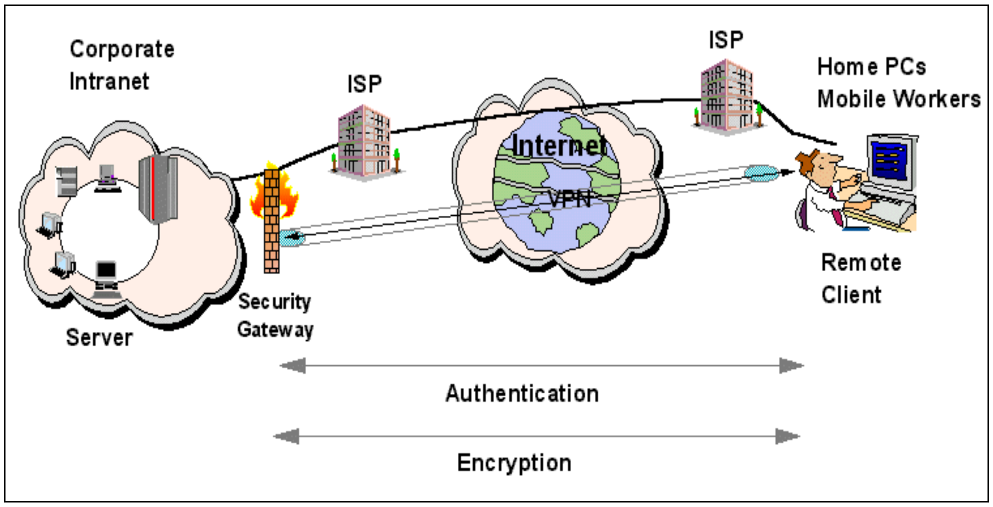


Figure 172. Remote access network

This allows remote users to initiate secure communications between a remote host and its corporate network.

A dynamic IP user connection typically involves these three connection types:

1. **Allow dynamic IP users to connect to your gateway**
   - The VPN protects traffic as it travels from the remote user to your gateway. Traffic that is routed beyond your gateway is unaffected by the VPN.
2. **Allow dynamic IP users to connect to your host**
   - The VPN protects traffic as it travels from the remote user to your host machine.
3. **L2TP connection**
   - The VPN protects traffic as it travels from the remote user to your gateway, or host machine. The remote user uses the Layer 2 Tunnel Protocol to create the connection.

We have used the dynamic VPN capabilities of VPN extensively in our scenarios. While the work involved may seem significant, the savings in subsequent management of keys and VPN connections are much greater.

Planning considerations: remote access network

The principles of planning for IPSec in this case are the same as in “Planning considerations: business partner network” on page 279, with one essential difference. Because the IP address of the remote workstation is not known (it is usually assigned dynamically by the ISP), the aggressive mode of IKE negotiation must be used. Aggressive mode uses only three messages to set up the association (compared to six in main mode), but does not protect the identities of the partners. Aggressive mode is described in 2.3.1.2, “Negotiating Security Associations (ISAKMP and IKE)” on page 31.

6.5.2 Defining tunnels for a VPN in z/OS

This section is not meant to be a comprehensive description of how to define IPSec under z/OS. Other redbooks and manuals define this process in detail, For example:

- *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 1: Configuration and Routing*, SG24-5227
- *SecureWay CS OS/390 V2R8 TCP/IP: Guide to Enhancements*, SG24-5631
- *A Comprehensive Guide to Virtual Private Networks, Volume I: IBM Firewall, Server and Client Solutions*, SG24-5201
- *A Comprehensive Guide to Virtual Private Networks, Volume III: Cross-Platform Key and Policy Management,* SG24-5339
- *OS/390 V2R10.0 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835.

Also consult Appendix E.5, “Configuration for three VPN scenarios” on page 376.

Furthermore, the many definitions involved in setting up certain tunnel types can be daunting. We recommend the use of the Firewall Technologies GUI to simplify the definition process.

We have already mentioned the existence of a white paper entitled *VPN Interoperability Demonstrated for OS/390* available at:

```
www.ibm.com/servers/eserver/zseries/networking/pdf/GM13-0029-00.pdf
```

This white paper describes several scenarios of different platforms building VPNs with z/OS.

Figure 173 on page 282 shows the main window of the Firewall Configuration Graphical User Interface from which you can reach the windows to define both manual tunnels and dynamic tunnels, including on-demand tunnels.

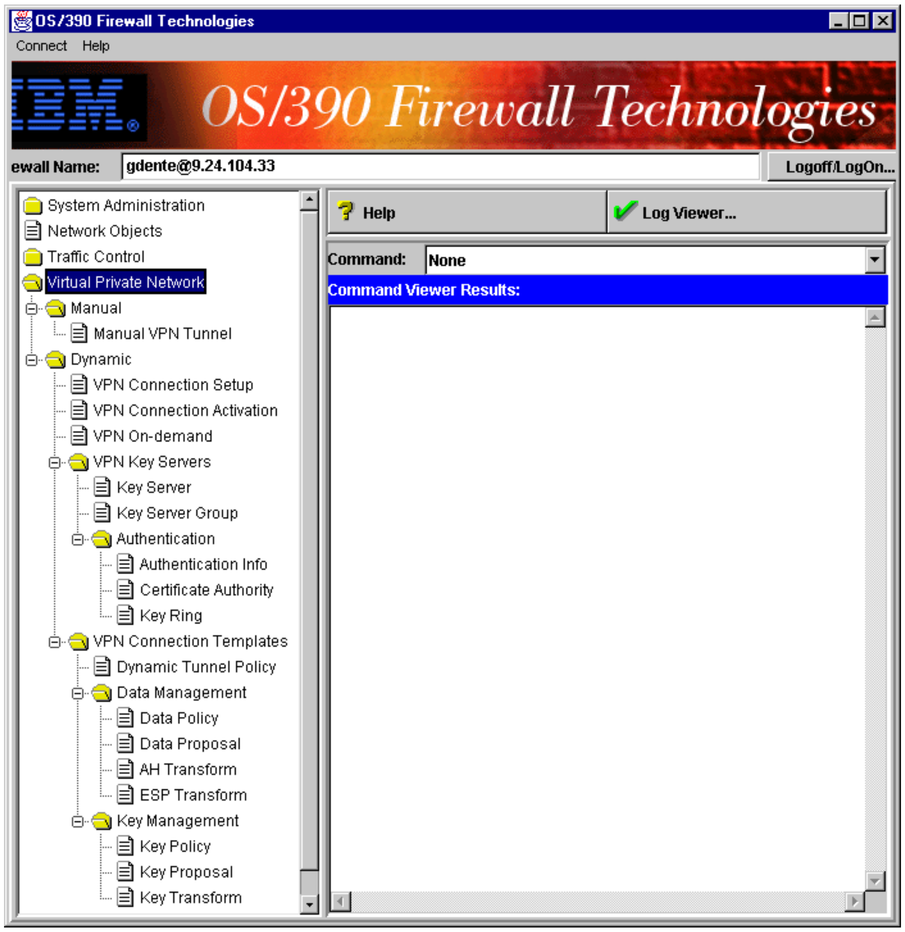


Figure 173. Firewall Technologies main window: expanded view for VPN

What follows are reviews of the types of definitions required without the accompanying detail on syntax or parameters. If you need more detail, please remember that step-by-step instructions for performing a VPN setup are contained in Appendix E, “Firewall Technologies configuration for VPN support” on page 347. Blank worksheets to plan for your own VPN setup are in Appendix F, “VPN planning worksheets” on page 439.

6.5.2.1 Manual tunnel definition for VPN

A manual tunnel requires manual administration of the tunnel policy at both ends of the tunnel. Keys may not be refreshed unless the tunnel has been deactivated and then reactivated with a new key configuration. A manual tunnel may operate in either transport or tunnel mode.

Suppose, for example, that you want to set up a tunnel through which only certain types of traffic may flow. The tunnel will extend from your z/OS system to the firewall that protects you from the Internet, as illustrated in Figure 174.

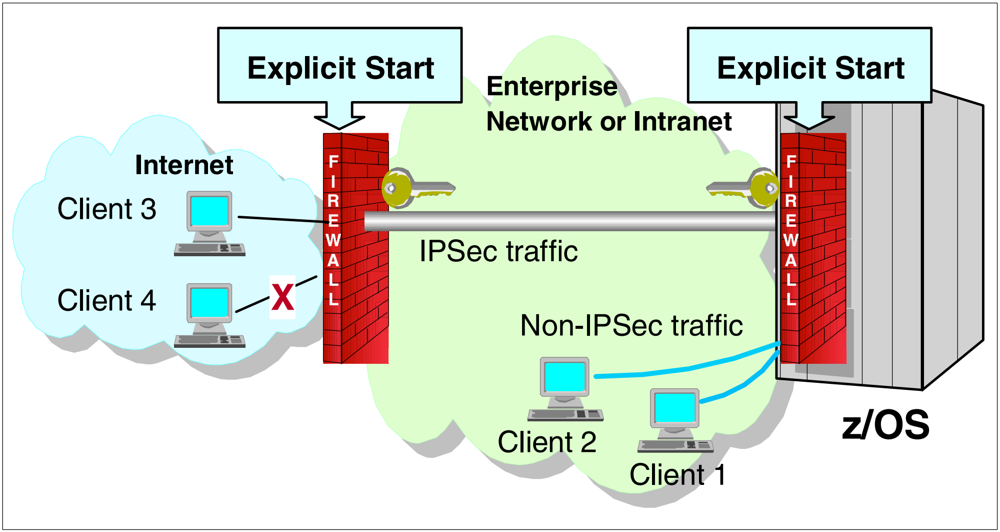


Figure 174. A manual tunnel example

Client 3's traffic will be allowed to pass through the firewall and will take advantage of the authentication and encryption provided by the IPSec tunnel built between the external firewall and Firewall Technologies at z/OS. Client 4's traffic will be prohibited from getting past the firewall into the Enterprise network. Clients 1 and 2 belong to the Enterprise Network and do not represent a security exposure. Therefore, they are subject to normal RACF and application security at z/OS and CS for OS/390 IP. Static keys are maintained throughout the life of the tunnel. The tunnel itself must be activated at both ends by operator command or explicit program invocation.

Figure 175 illustrates the objects that must be created to set up a such a manual VPN.

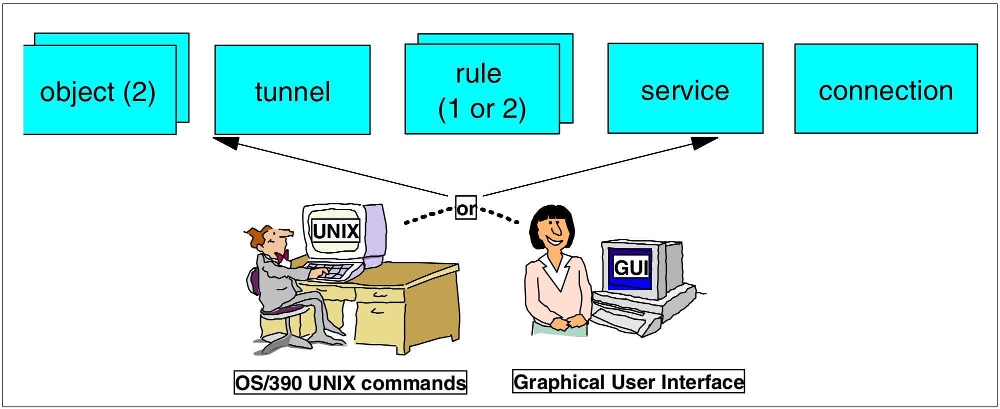
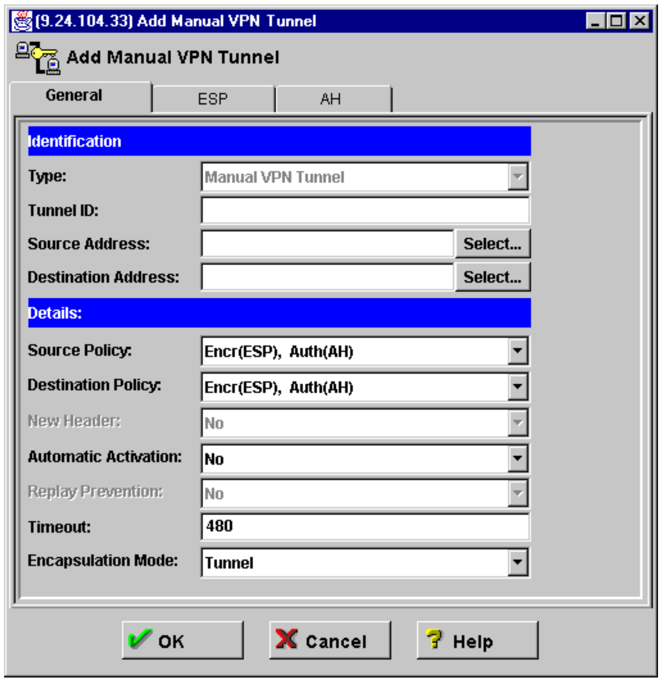


Figure 175. Basic firewall technology manual IPSec tunnel definition

Object A network object defines a subnet or single IP address. There is usually an object for the OS/390 (local) side of the connection and another object for the remote end of the connection. The UNIX command to create an object is: `fwnwobj`.

Tunnel	A tunnel policy determines the type of encryption and authentication schemes that will be applied to the data carried across the tunnel. The tunnel policy must be the same at both ends. The UNIX command that may be used in place of the GUI is `fwtunnl`. The systems at the end of the tunnel must share a secret key.
Rule	A rule defines a permit or deny for some type of higher lever IP protocol. You should define a rule for inbound traffic and a rule for outbound traffic or a single rule that is valid for both directions. The UNIX command `fwfrule` defines this.
Service	A service defines a set of rules. In fact a service is a set of allowed protocols between some source and destination IP address. The UNIX command `fwservice` creates this.
Connection	A connection defines a set of services. A connection defines which services are allowed between two IP addresses/subnets. The UNIX command `fwconns` defines this function.

You have already seen the windows to define four of the object types necessary to establish a manual tunnel. In Figure 176 you see the window that may be used to define the tunnel endpoints:

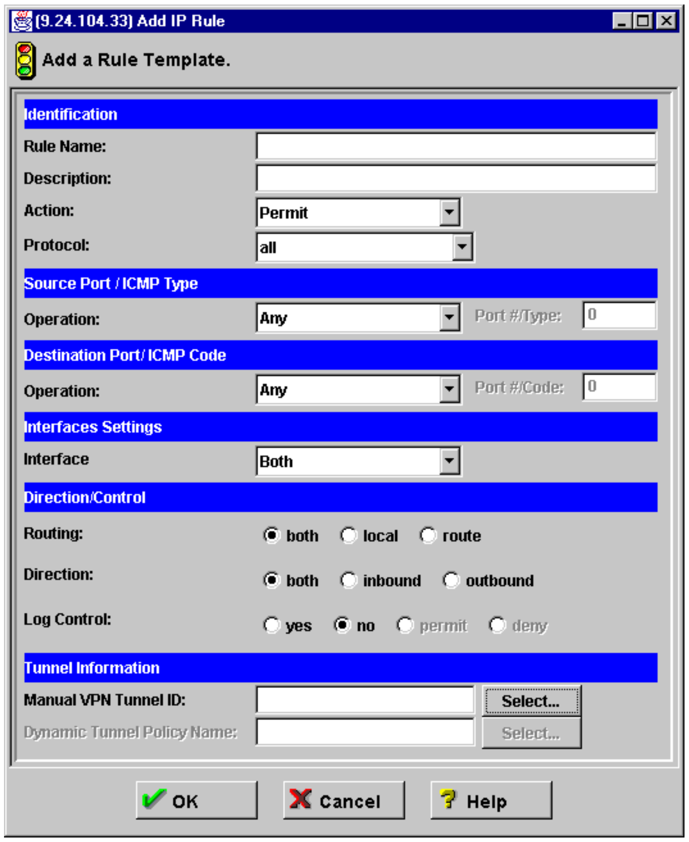


Type:	Manual VPN Tunnel
Tunnel ID:	
Source Address:	
Destination Address:	
Source Policy:	Encr(ESP), Auth(AH)
Destination Policy:	Encr(ESP), Auth(AH)
New Header:	No
Automatic Activation:	No
Replay Prevention:	No
Timeout:	480
Encapsulation Mode:	Tunnel

Figure 176. Manual tunnel definition

The source address field refers to the non-secure side of the tunnel. The destination address represents the secure side of the tunnel. Note the tabs to select ESP or AH protocols.

Figure 177 shows the window that defines a rule that is to be associated with a manual tunnel.


(9.24.104.33) Add IP Rule

Add a Rule Template.

Identification

Rule Name:	
Description:	
Action:	Permit
Protocol:	all

Source Port / ICMP Type

Operation:	Any	Port #/Type:	0

Destination Port/ ICMP Code

Operation:	Any	Port #/Code:	0

Interfaces Settings

Interface	Both

Direction/Control

Routing:	[✓ both] local route
Direction:	[✓ both] inbound outbound
Log Control:	yes [✓ no] permit deny

Tunnel Information

Manual VPN Tunnel ID:		Select...
Dynamic Tunnel Policy Name:		Select...

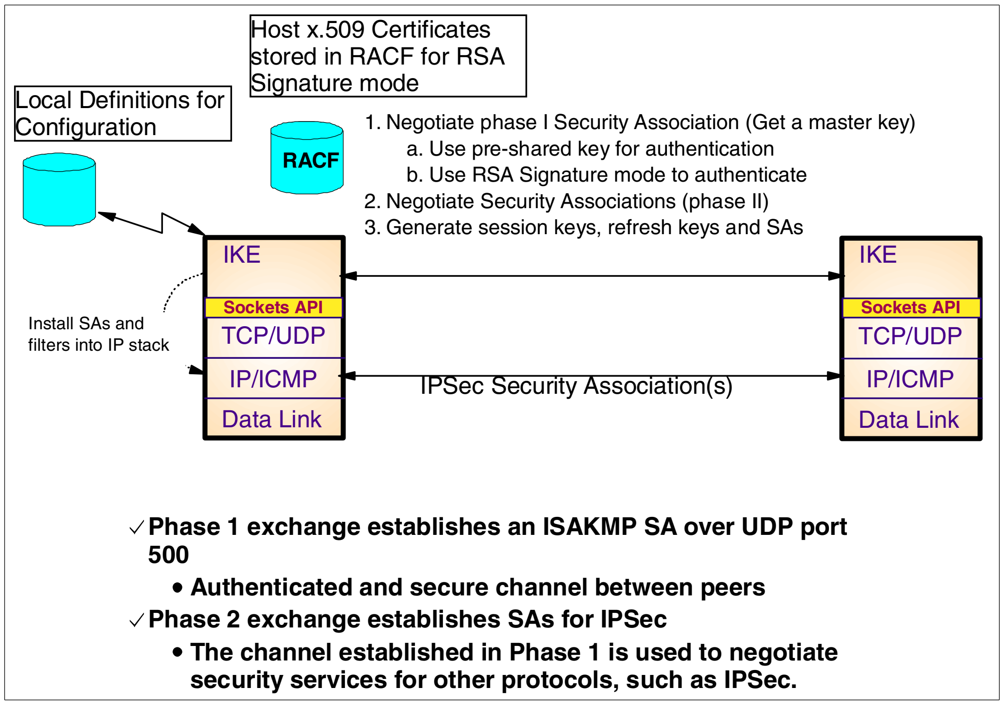
Figure 177. Rule for manual VPN

This rule specifies that any matching data requires the existence of an active manual tunnel that it may use. If there is no active tunnel, the packets will be rejected.

6.5.2.2 Dynamic tunnel definition for VPN

The Internet Key Exchange (IKE) protocol introduced in CS for OS/390 V2R8 IP has been implemented in the ISAKMP daemon of OS/390 Firewall Technologies. This daemon communicates with the Firewall Kernel and supports IKE peer authentication via pre-shared keys or RSA digital signatures. Despite the fact that we are dealing with dynamic tunnels, we must nevertheless manually define the IKE peers.

Recall that there are detailed descriptions of defining dynamic VPNs in Appendix E, “Firewall Technologies configuration for VPN support” on page 347.


Host x.509 Certificates stored in RACF for RSA Signature mode
Local Definitions for Configuration
RACF
1. Negotiate phase I Security Association (Get a master key)
a. Use pre-shared key for authentication
b. Use RSA Signature mode to authenticate
2. Negotiate Security Associations (phase II)
3. Generate session keys, refresh keys and SAs
IKE
Sockets API
TCP/UDP
IP/ICMP
Data Link
Install SAs and filters into IP stack
IPSec Security Association(s)
IKE
Sockets API
TCP/UDP
IP/ICMP
Data Link

- ✓ **Phase 1 exchange establishes an ISAKMP SA over UDP port 500**
  - **Authenticated and secure channel between peers**
- ✓ **Phase 2 exchange establishes SAs for IPSec**
  - **The channel established in Phase 1 is used to negotiate security services for other protocols, such as IPSec.**

Figure 178. IKE exchanges: two phases

Figure 178 shows the results of an IKE exchange. The phase 1 exchange protects the IKE flows. It authenticates both IKE peers and creates identical keying material with a Diffie-Hellman exchange to encrypt further IKE exchanges, thus establishing an ISAKMP Security Association (SA). All keying material and additional information is stored in this SA. Both peers verify the generation of their keying material by generating a hash over the keying material and sending this hash to each other. For authentication with signatures the hash is also signed.

There are two ways IKE peers authenticate each other:

1. By employing a pre-shared key to use for authentication

   For shared-key authentication the key generation process uses the shared key as an input. Therefore if IKE peers were to have different shared keys, they would generate different keying materials. As a result the IKE phase one exchange would fail.

2. By employing a host X.509 certificate with an RSA signature.

   When an RSA signature in an X.509 certificates is used, the signature is verified by the receiving entity with the public key derived from the certificate. The certificate is sent together with the signature of the hash.

The phase 2 exchange, protected by the encryption negotiated in phase 1, creates the SA for the IP layer 3 encryption (IPSec) that will be used in the VPN over which data is exchanged. The Perfect Forward Secrecy (PFS) attribute of IKE ensures that each key refreshed has no relationship to any previously used key, and that the compromise of a single key does not increase the vulnerability of other keys to compromise. Refresh frequency is based upon configuration values that you select for maximum byte count or time interval.

We can use nearly the same illustration we used to describe manual VPNs to depict a dynamic VPN. See Figure 179.

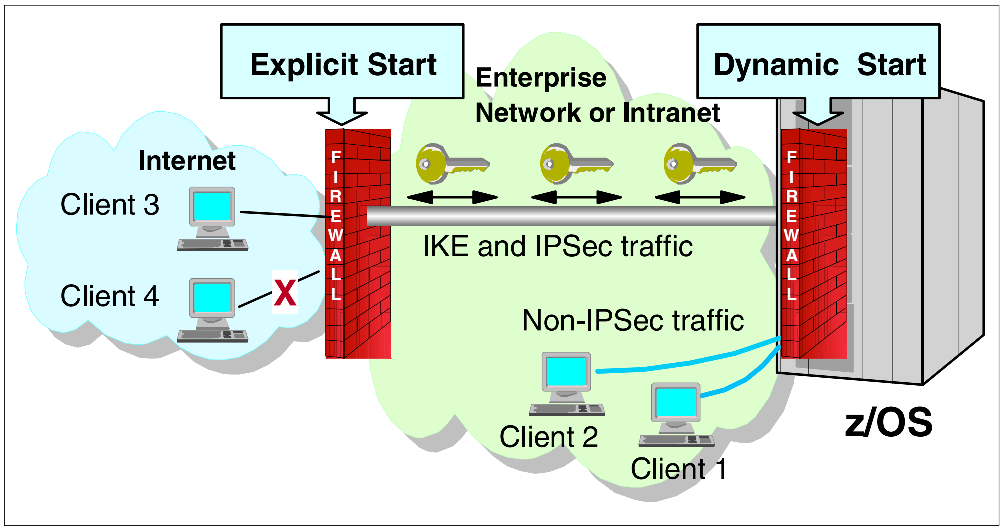


Figure 179. A dynamic tunnel example

Although the illustration shows that the external firewall explicitly executes commands to activate its end of the tunnel, while the z/OS end of the tunnel at the interior firewall activates dynamically, in fact the roles could be reversed. The real point of the picture is to indicate that the keys for the data exchange are dynamically refreshed according to parameters that you specify.

Policies

When defining a VPN, we need to define policies. The policies define a combination of security services and acceptable security service attributes. They are generic in that they can be reused by many VPN configurations. However, we recommend that you define three different security levels of policy to be used in subsequent VPN configurations: high security, medium security and low security.

VPN configurations need three types of policies:

- Key management
- Data management
- Dynamic tunnel

Phase 1 IKE exchange

The phase 1 IKE exchange to establish a dynamic VPN requires the definition of various objects for key management depicted in Figure 180:

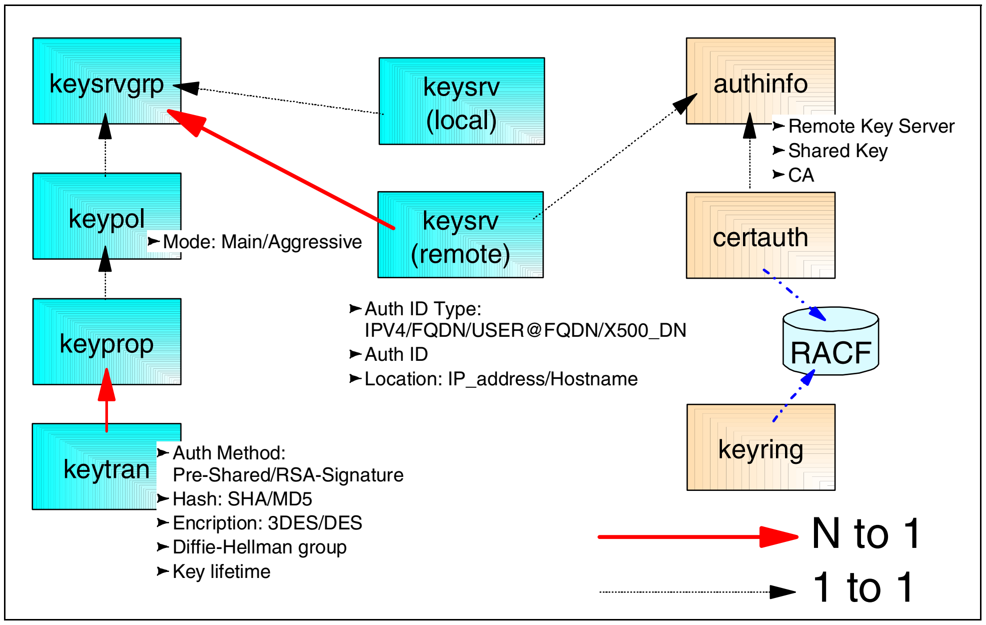


Figure 180. IKE phase 1 object relationships

In the next paragraphs we explain the key management and authorization objects that must be created in order for the phase 1 IKE exchange to succeed.

Key management policy

The key management policy is defined using a key policy template. It defines a generic strategy for how key negotiations between IKE servers are to be protected. Later, it will be specified on a key server group, and could be specified on multiple key server groups. Figure 181 highlights three objects necessary to define the key management policy for the IKE Security Association: keypol, keyprop, keytrans. It also highlights how the keyserver or keyserver groups must be defined which will negotiate key management policy with each other during the IKE exchange.

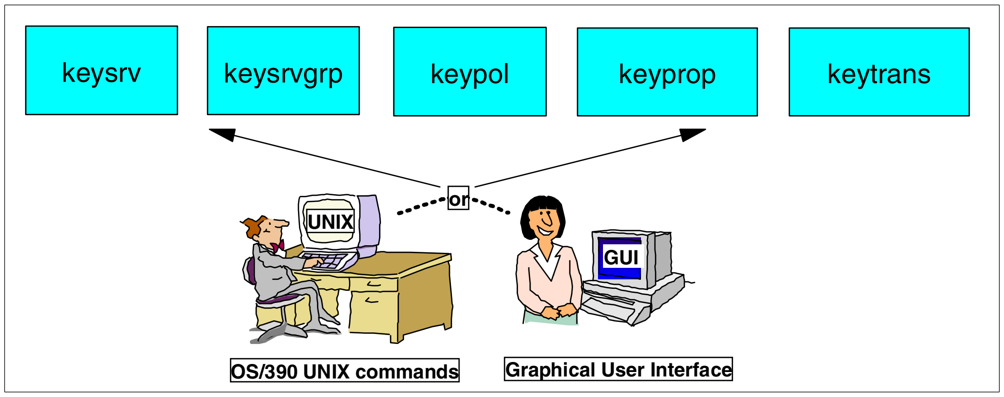


Figure 181. Defining IKE SA for phase 1 IKE protocol: IKE peers and key management policy

Using these definitions of the IKE peers and the key management policy, the firewall negotiates an IKE SA with its IKE peer.

keysrv Key servers are all local and remote IKE peers in your network. A key server object defines the authentication ID used during the

IKE phase 1 exchange. Several ID types are possible. The types are: IPv4 address, Fully Qualified Domain Name (FQDN), the user@FQDN, and the X.500 Distinguished name (DN). If certificates are used, the authentication ID must also reside in the certificate. See Appendix D, "OS/390 certificate management using RACF" on page 321 for further information. Use `fwkeysrv` in UNIX to define this.

keysrvgrp A keyserver group object defines a relation between one local key server and N remote key servers. By grouping them you can control how you identify yourself to remote IKE peers. In addition you can control the IKE encryption policy used by the group. Use `fwkeysrvgrp` in UNIX to define this.

keypol A key policy object defines the IKE negotiation types as defined in RFC 2409. In addition it contains a reference to a key proposal object. Use `fwkeypol` in UNIX to define this.

keyprop A key proposal object defines a set of IKE transforms. The proposal is sent to the IKE peers. A transform set is chosen that is acceptable by both peers. Use `fwkeyprop` in UNIX to define this.

keytran A key transform object defines an IKE transform. An IKE transform contains an authentication method (for example, pre-shared or signature), a HASH algorithm (MD5 or SHA) for authentication and an encryption algorithm (DES or 3DES) for confidentiality.

Figure 182 shows a key policy where it would be acceptable to protect key negotiations in one of the following ways:

- IKE with MD5
- IKE with SHA

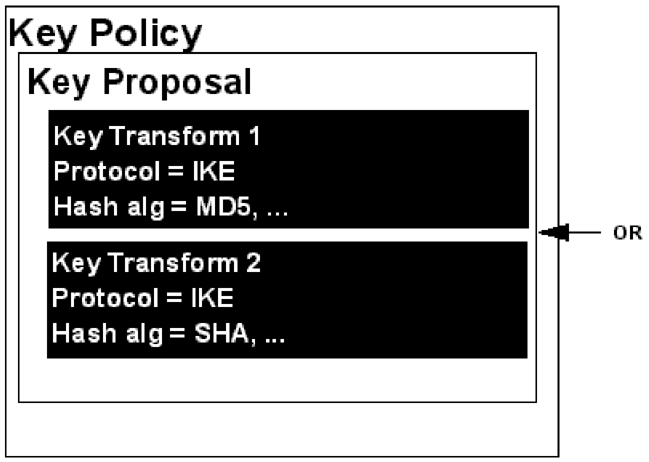


Figure 182. Key policy sample

Key policy is sent by the initiating IKE server to its peer; only one proposal may be sent. If the proposal contains multiple key transforms for IKE, the peer picks one, then the peer sends back the choice made.

The following table shows examples of key management policies with three security levels (low, medium, high):

Table 25. Sample key management policies

Key management policy	Low	Medium	High
Policy Name	IBM_low_key	IBM_med_key	IBM_high_key
Mode	Aggressive	Main	Main
Auth Method	Pre-shred keys	RSA	RSA
Encryption Algorithm	DES	Triple DES	Triple DES
Hash	MD5	SHA	SHA
SA Lifetime	20 Hours	20 Hours	1 Hour
Diffie-Hellman Group	1	1	2

In the process of key management definition, we have to create Key Transform objects, then associate them with the Key Proposal as illustrated in Figure 180 on page 288 and Figure 181 on page 288 above.

Figure 180 on page 288 showed how three additional objects must be created in order to authenticate the IKE partners:

authinfo The authentication information object defines a pre-shared key belonging to a remote key server and an indication which certificate should be used. In addition you are able to specify a CA name in the authentication object. This name is only used to help a peer select a certificate. When a peer wants to use signature authentication, a certificate must be sent containing the public key of the peer. This certificate must be signed by a CA we trust, to help the peer in selecting a certificate signed by a trusted CA we send the name of the CA to the peer. In addition the certificate we send also depends on hints from the remote peer. Normally the auth ID of the local key server is used to select a our certificate. If you use the shared key to authenticate the IKE peer, neither the Certificate Authority nor Key Ring objects are required. It is recommended that you use the different shared key for each peer. Use `fwauthinfo` in UNIX to define this.

certauth The Certificate Authority object identifies the RACDCERT label for Certificate Authority. This certificate will be used to validate certificates from peers. The UNIX command may be used to define this: `fwcertauth`.

keyring This is a RACF-controlled certificate ring. ISAKMP daemon will search this ring for certificates and Certificate Authorities when it is started. Use `fwkeyring` in UNIX to define this.

The management of the certificate and its keyring for VPN that you have just pointed to in the above definitions is further explained in Appendix D, “OS/390 certificate management using RACF” on page 321. RACF definitions must be made to support the X.509 certificates: the local key server's own certificate and the CA certificate that signed the local/remote key server's certificate. This is best explained with the help of Figure 183.

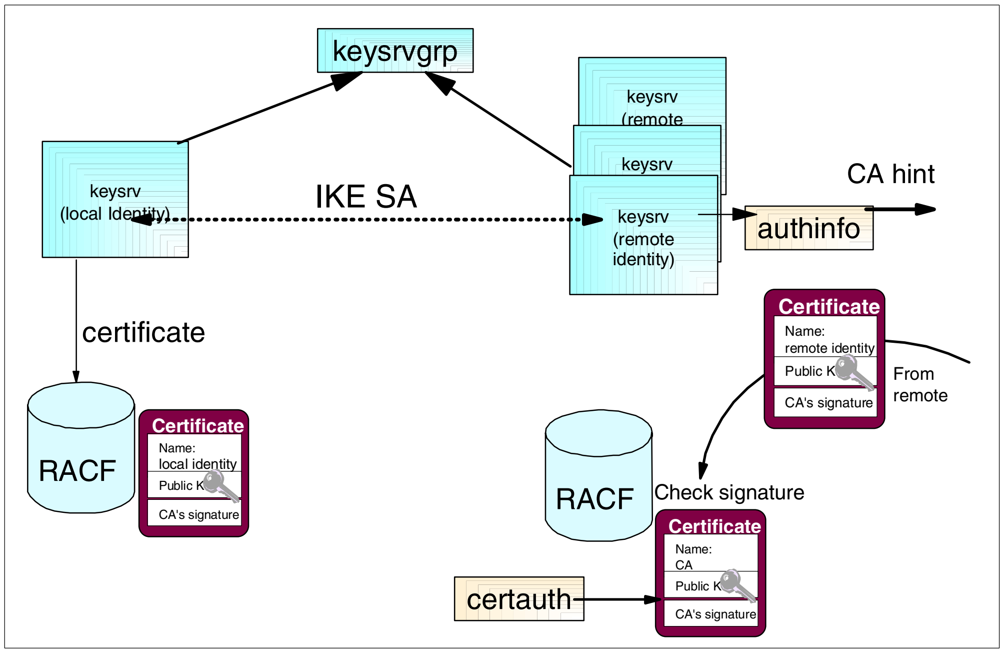


Figure 183. IKE object relationship for phase 1 authentication

Although the picture concentrates on the use of certificates, bear in mind that pre-shared keys may be used to authenticate the IKE partners. In this case the pre-shared key is defined in the Authinfo object and neither the Certificate Authority nor Key Ring objects are required. It is recommended that you use a different shared key for each peer.

When using RSA Signature mode of authentication, the association between a CertAuth object and a remote key server is intended to be a hint to your peer, and the IKE peer is supposed to use this hint or fail the phase 1 negotiation. This hint tells the IKE peer which certificate authority should have signed the certificate that it uses when creating his signature. The hint is sent to the IKE peer when OS/390 is playing the role of the initiator for the phase 1 in both aggressive and main mode, and the responder in aggressive mode.

When OS/390 is the responder in main mode, it will look for the key server entry based on the IP address of the IKE peer. It will search a key server entry that has a matching IP address specification. If it does not exist, the search will be done using the host name associated with the IP address. If an entry is found in either way, it look to see if a CertAuth object has been associated with this key server entry.

The hint is sent in the form of a certificate request payload that contains the name of a CA. OS/390 picks a certificate to create its signature based on a hint given from the IKE peer.

Phase 2 IKE exchange

Recall from Figure 178 on page 286 that there are two phases for the IKE exchange. We have already seen that the key management policy and its objects must be created for this exchange to complete successfully. Now we must look at the objects required for the IKE phase 2 exchange. See Figure 184.

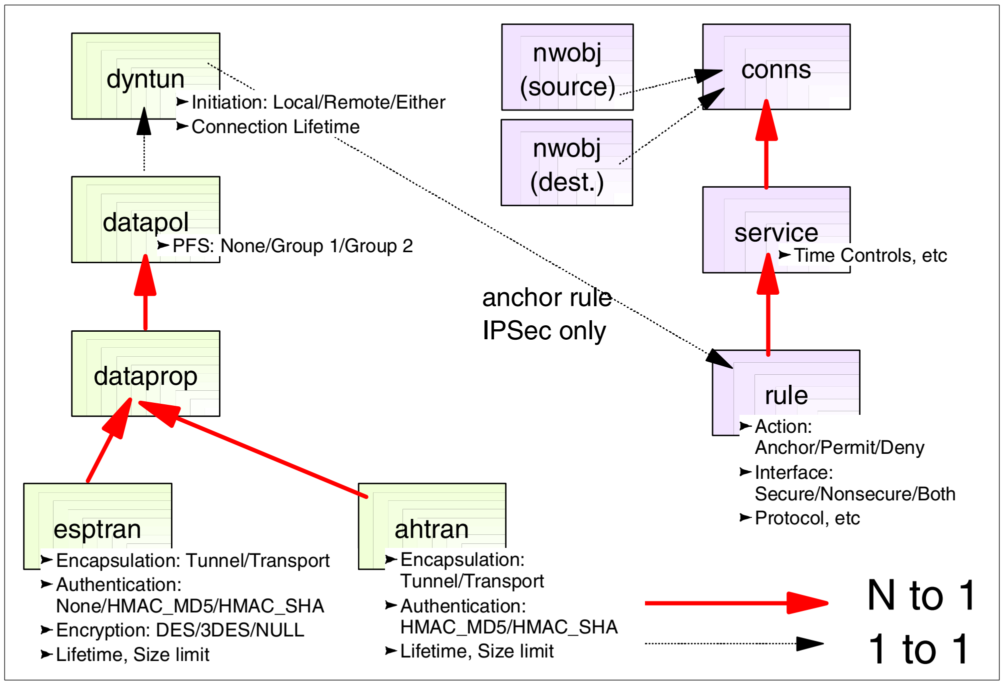

dyntun
►Initiation: Local/Remote/Either
►Connection Lifetime
datapol
►PFS: None/Group 1/Group 2
dataprop
esptran
►Encapsulation: Tunnel/Transport
►Authentication: None/HMAC_MD5/HMAC_SHA
►Encryption: DES/3DES/NULL
►Lifetime, Size limit
ahtran
►Encapsulation: Tunnel/Transport
►Authentication: HMAC_MD5/HMAC_SHA
►Lifetime, Size limit
nwobj (source)
nwobj (dest.)
conns
service
►Time Controls, etc
anchor rule
IPSec only
rule
►Action: Anchor/Permit/Deny
►Interface: Secure/Nonsecure/Both
►Protocol, etc
N to 1
1 to 1

Figure 184. Object relationship for IKE phase 2 exchange

These objects define the data management policy and the dynamic tunnel and its policy (dynamic tunnel policy).

Data management policy

The data management policy is defined using a data policy template. It defines a generic strategy for how IP packets that flow through a VPN are to be protected. Later it will be specified on a dynamic tunnel policy template and could be specified on multiple dynamic tunnel policy templates. Figure 185 highlights the objects to be defined when creating a data management policy and defining a dynamic tunnel policy. These objects allow the firewall to negotiate an IPSec Security Association with its IKE peer.

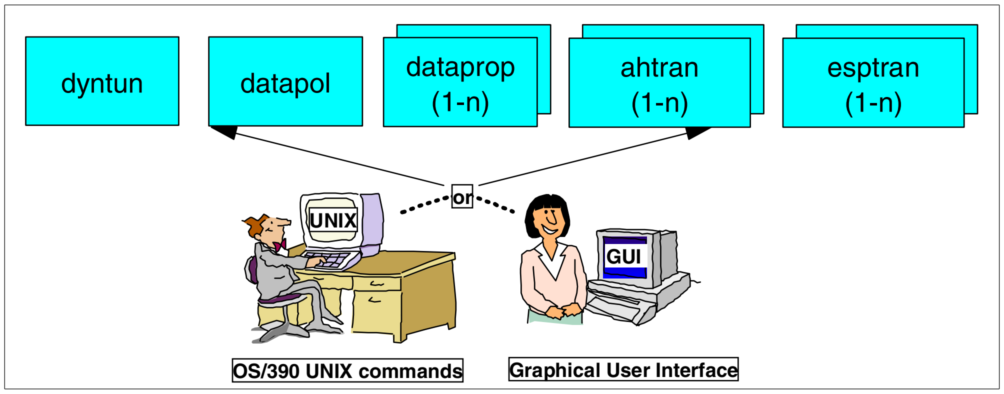


Figure 185. Defining IPSec SA for phase 2 IKE protocol: data management and the dynamic tunnel

dyntun A dynamic tunnel policy object defines an IPSec data policy acceptable for this tunnel. Once IKE has created the IPSec SA it must update the Firewall definitions on the fly. Therefore Firewall definitions are needed to successfully accept an IPSec SA. A dynamic tunnel policy definition forms the start of the object chain needed for the IPSec SA acceptance. The dynamic tunnel policy also defines which side can initiate the IPSec exchange and for how long this tunnel will be active. This object will be specified in a filter rule object. The UNIX command for this function is `fwdyntun`.

datapol A data policy object defines a set of data proposal objects which define information required when negotiating keys for data exchanges: the perfect forward secrecy (PFS) selection and list of data proposals. These proposals are exchanged with a peer during the IPSec exchange. The UNIX command for this function is `fwdatapol`.

dataprop A data proposal object defines a set of AH transforms and a set of ESP transforms. There could be multiple data proposal objects. The UNIX command for this function is `fwdataprop`.

ahtran An AH transform object defines the authentication algorithm (for example, AH or SHA) encapsulation mode (transport or tunnel) and lifetime for an AH transform. There could be multiple AH transform objects. The UNIX command for this function is `fwahtran`.

esptran An ESP transform object defines the authentication algorithm (for example, AH or SHA), encryption algorithm (DES or 3DES), encapsulation mode (transport or tunnel) and lifetime for an ESP transform. There could be multiple ESP transform objects. The UNIX command for this function is `fwesptran`.

The type of IPSec encryption we accept is defined with the above mentioned objects as are the tunnel characteristics.

The diagram in Figure 186 shows a data policy where it would be acceptable to protect IP traffic in one of the following ways:

- AH with HMAC_MD5 and ESP with DES
- AH with HMAC_SHA and ESP with DES
- ESP with Triple DES

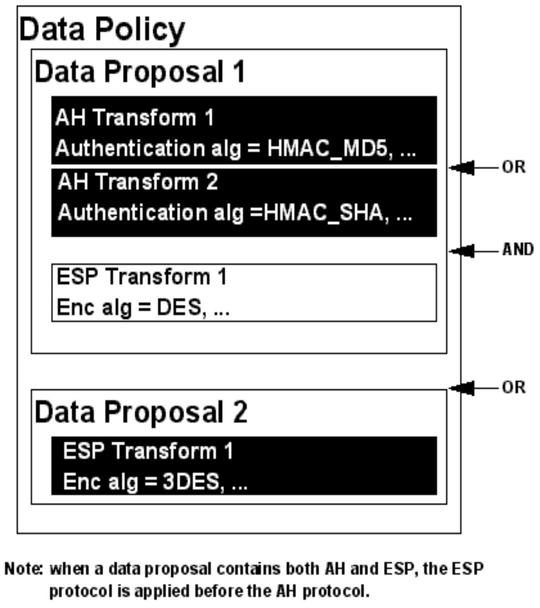

Data Policy
Data Proposal 1
AH Transform 1
Authentication alg = HMAC_MD5, ...
AH Transform 2
Authentication alg =HMAC_SHA, ...
OR
AND
ESP Transform 1
Enc alg = DES, ...
OR
Data Proposal 2
ESP Transform 1
Enc alg = 3DES, ...
Note: when a data proposal contains both AH and ESP, the ESP protocol is applied before the AH protocol.

Figure 186. Data policy sample

A data policy is sent by the initiating device to its peer; if multiple proposals are sent the peer picks one proposal. If the proposal contains multiple transforms for the same security service (for example, multiple AH transforms), the peer picks one. After this, the peer sends back the choices made.

The following tables show examples of data management policies that would be useful in the scenarios we describe in this redbook. Four possible combinations exist; AH and ESP, transport mode, and tunnel mode.

Table 26. Sample data management policies (ESP, tunnel or transport mode)

Data management policy	**Low**	**Medium**	**High**
Transform	ESP	ESP	ESP
Encryption Algorithm	DES	Triple DES	Triple DES
Auth Algorithm	MD5	SHA	SHA
PFS	No	No	Group 2
SA Lifetime	30 min	20 min	10 min

Table 27. Sample data management policies (AH, tunnel or transport mode)

Data management policy	**Low**	**Medium**	**High**
Transform	AH	AH	AH
Auth Algorithm	MD5	SHA	SHA
PFS	No	No	Group 2
SA Lifetime	30 min	20 min	10 min

In the process of data management policy definition, we have to define (based on our company policies) AH transform objects or ESP transform objects or both together, then associate them to a data proposal object, and finally create a data policy using the data proposal object.

Dynamic tunnel policy

Dynamic tunnel policies are defined using a dynamic tunnel template. Such a policy defines a generic strategy for how a dynamic tunnel is to be managed. We have already seen a definition of the dynamic tunnel policy in Figure 185 on page 292. Recall that the dynamic tunnel policy must indicate which IKE peer may initiate the tunnel and what the duration of the tunnel will be. See Table 28.

Table 28. Dynamic tunnel policy

Dynamic tunnel policy	
Initiation	Either
Connection Lifetime	1440

We need to revisit that definition here because dynamic tunnel policy is closely related to the data management policy and has to be specified on an anchor rule. A dynamic tunnel policy could be specified on multiple anchor rules.

It is important in this context to understand the difference between a dynamic *tunnel* and a dynamic *connection*.

A dynamic connection is the communication circuit between the source and destination of data sent through a VPN.

A dynamic tunnel is the portion of a dynamic connection that is protected by IPSec. It is established between two IPSec devices running IKE. Please refer to Figure 187:

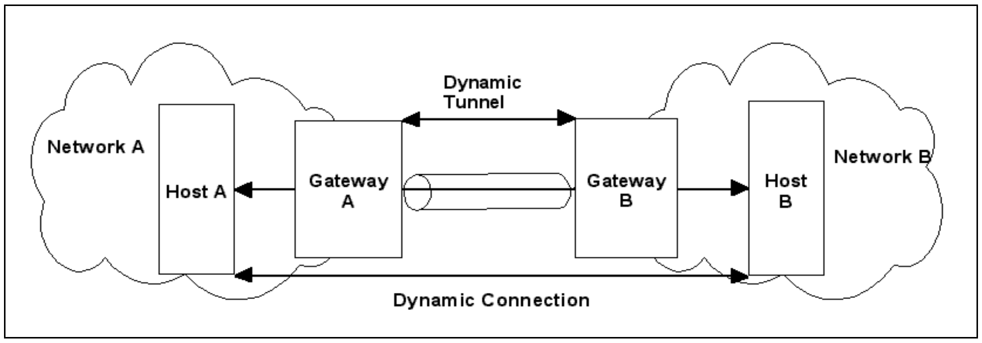


Figure 187. Dynamic tunnel versus dynamic connection

The dynamic tunnel policy object will be associated with an Anchor Filter Rule object.

In Figure 188 we define objects that are related to traffic that is allowed to flow over the dynamic tunnel. You have seen a similar figure before when we discussed setting up simple firewall filtering in 6.4.1, “Defining IP filters” on page 264.

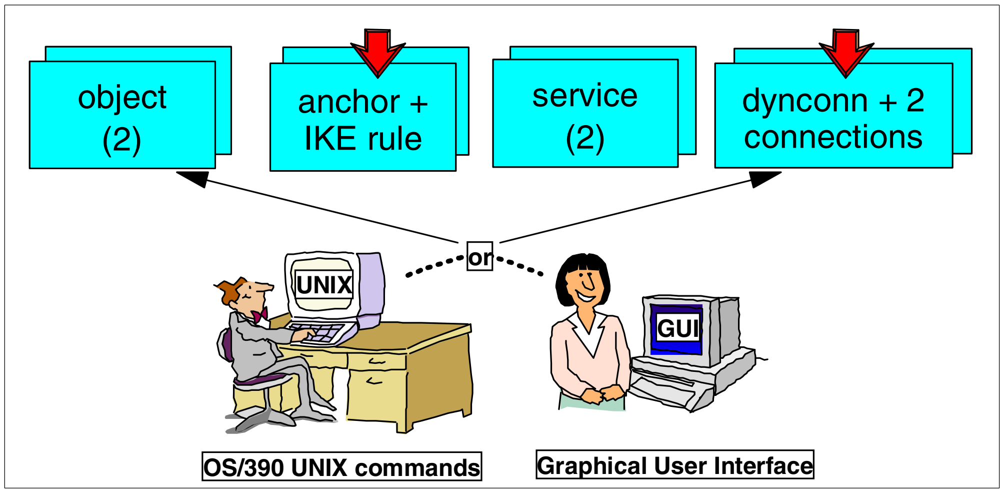


Figure 188. Filter rules for the dynamic tunnel

Note the two differences with regard to the objects you created for simple filters: the rule for a dynamic tunnel is a special kind of rule called an *anchor rule* and the connection that may need to be defined is a special *dynamic connection*. A successful IPSec negotiation results in an IPSec tunnel and a firewall filter.

Object	A network object defines a subnet or single IP address. There is usually an object for the OS/390 (local) side of the connection and another object for the remote end of the connection. The UNIX command to create an object is: `fwnwobj`.
Rule	A rule defines a permit or deny for some type of higher lever IP protocol. For dynamic VPN connections you must define an *anchor rule*. The anchor rule is a template that indicates which traffic must be protected by a dynamic tunnel, and identifies a dynamic tunnel policy. The tunnel defines which encryption types are acceptable. The anchor rule allows dynamic connection objects to be created. When a dynamic VPN connection is activated, the anchor rule is used to determine the placement of the dynamically generated filter rules among the static permit and deny rules. The UNIX command to create an anchor rule is: `fwfrule`.
Service	A service defines a set of rules. In fact a service is a set of allowed protocols between some source and destination IP address. With dynamic tunnels it is recommended that you define two services: one establishes the Anchor Filter Rule between the connection endpoints and one establishes the permit filter rules for the Key Server traffic, that is, the traffic for the ISAKMPD and IPSec protocols, including the IPSec and AH headers. The UNIX command `fwservice` creates this.
Connection	A connection defines a set of services. A connection defines which services are allowed between two IP addresses/subnets. Filters are derived from the connection objects. Therefore you must define connection objects to allow the creation of dynamic filters by IKE. This means that we may need to create two connection objects: one that allows the

ISAKMP and IPSec (AH and ESP) protocol to flow between peers and one that allows encrypted IP traffic to flow over an IPSEC tunnel. The command `fwconns` creates the definition for the IKE connection partners (source/local key server and destination/remote key server) who are to establish the dynamic tunnel. Create two of these. The connection between key servers defines the type of authentication and encryption that will be used for the IKE protocol in this dynamic VPN. The anchor connection defines the type of application traffic that is permitted to flow over the IPSec tunnel.

Dynamic VPN Connection The dynamic VPN connection object defines information that is used to activate a specific VPN tunnel. The object references the remote key server and key server groups for which the connection should be activated. The local (source) and the destination (remote) network objects define which network entities are allowed to send data through the tunnel. The dynamic connection also defines which protocols and which type of port traffic will be protected by IPSec. The UNIX command `fwdynconns` creates the dynamic connection definition for the ISAKMP protocol; however, this definition needs to be in place only if OS/390 is to initiate the IKE exchange.

The use of a dynamic VPN connection object is optional; it is required only if CS for OS/390 IP is the initiator of the dynamic VPN connection. See Figure 189.

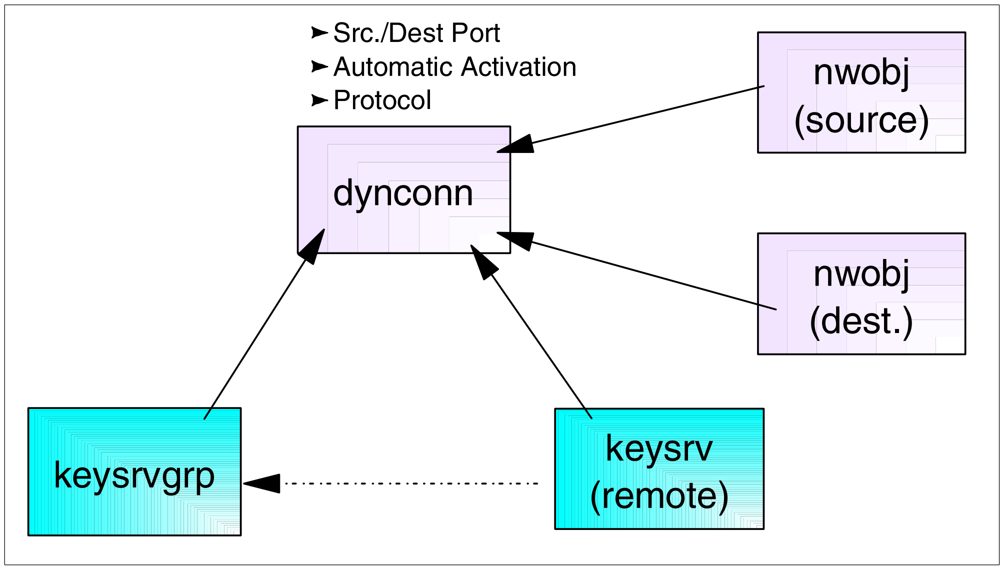
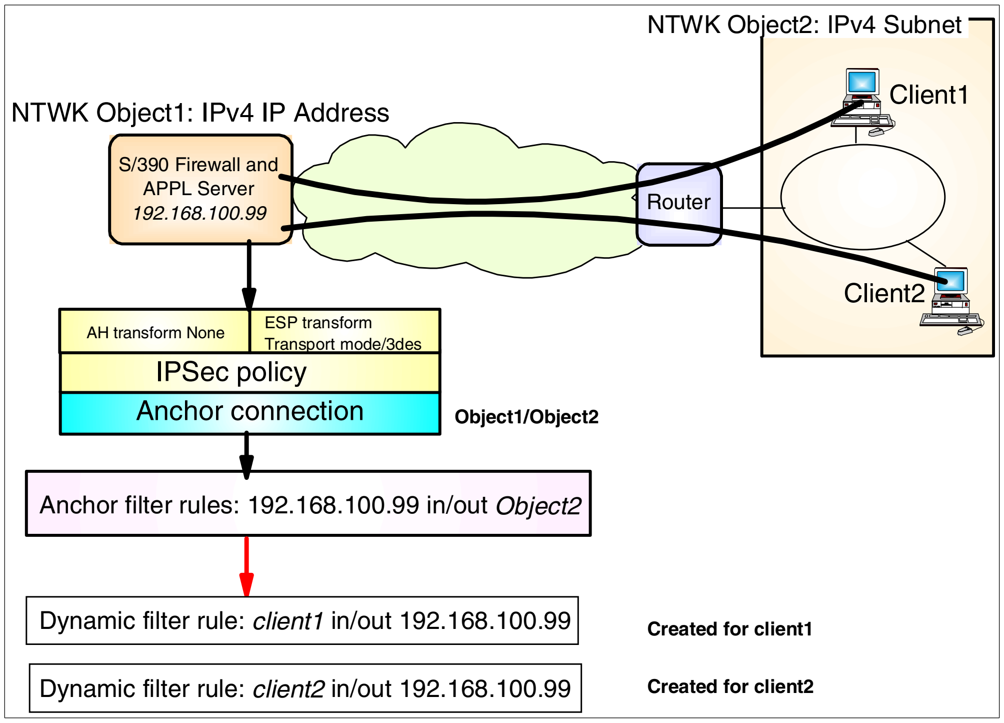


Figure 189. Starting an IKE exchange from z/OS

To illustrate how filter rules are associated with an anchor rule and how a connection object is associated with an anchor rule, we provide Figure 190.



NTWK Object2: IPv4 Subnet
Client1
NTWK Object1: IPv4 IP Address
S/390 Firewall and APPL Server 192.168.100.99
Router
Client2
AH transform None
ESP transform Transport mode/3des
IPSec policy
Anchor connection
Object1/Object2
Anchor filter rules: 192.168.100.99 in/out Object2
Dynamic filter rule: client1 in/out 192.168.100.99
Created for client1
Dynamic filter rule: client2 in/out 192.168.100.99
Created for client2

Figure 190. Anchor connection with subnet object

When a connection object is activated, the following filter(s) is/are generated in the firewall IP kernel:

- A filter rule definition, if the connection is not based on anchor rules
- Two filter rules, that is an inbound filter and an outbound filter, from one connection definition based on anchor rules

Filter rules from an anchor connection contain:

- The Rule Type indicating that the rule is s*tatic/anchor*
- The IPSec data encryption policy referenced in the tunnel statement of the filter
- The IP addresses or address ranges taken from the network objects

After changing any of the objects related to a connection object you must regenerate and activate the filters.

You see in Figure 190 that an anchor connection object has been defined that references an anchor filter rule. The network Ojbect1 defines an IPv4 IP address of OS/390 Firewall, whereas Object2 identifies an IPv4 subnetwork address where clients hosts are located. When the connection object is activated, two dynamic filter rules are generated for each client that initiates a connection to 192.168.100.99. Any clients on the IPv4 subnet that is defined as Object2 are allowed to create dynamic VPN connections. In summary, the anchor filters are the result of our definitions, while the dynamic filters are the result of the exchange of IKE messages between both peers.

6.5.2.3 On-demand dynamic tunnel definition for VPN

On-demand dynamic tunnel is an enhancement to dynamic VPNs that allow an outbound SA to be set up automatically when the designated network traffic requires that it be transmitted securely through a VPN. See Figure 191.

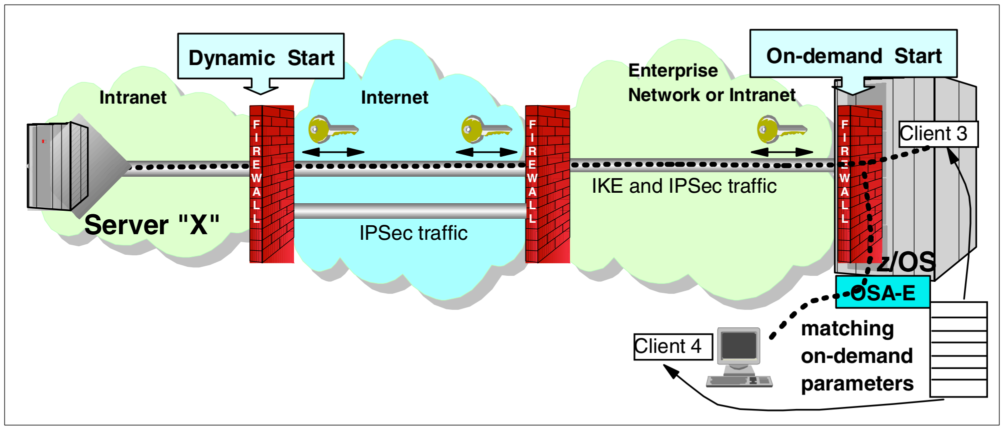
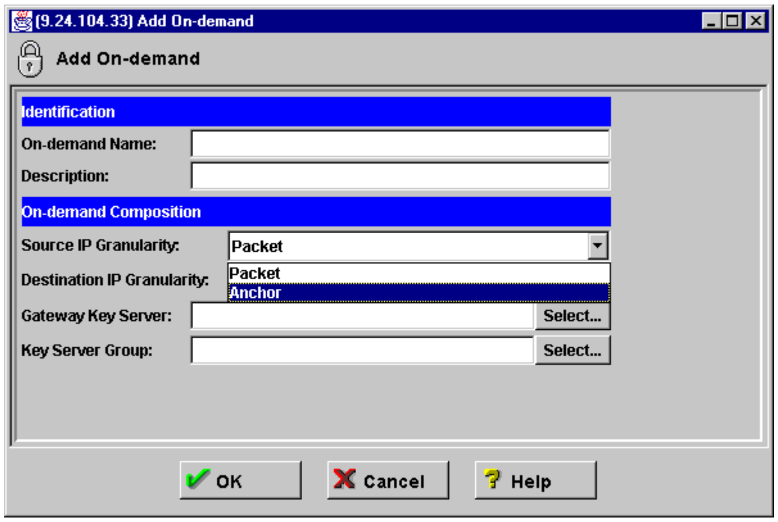


Figure 191. An on-demand VPN example

If Client 3's or Client 4's request for a connection to Server "X" matches the parameters that have been defined for an on-demand activation of the IPSec tunnel, the VPN will be automatically activated. Note that either an internal z/OS client, Client 3, or an outboard client like Client 4, that is using the CS for OS/390 IP stack as a router, can take advantage of on-demand activation if the parameters are set up accordingly.

A new UNIX command, `fwondemand`, is introduced to define the on-demand object. The window in Figure 192 shows how to define this.



On-demand Name:	
Description:	
Source IP Granularity:	Packet
Destination IP Granularity:	Packet / Anchor
Gateway Key Server:	
Key Server Group:	

Figure 192. Viewing an existing on-demand object

The connection definition points to this on-demand object in Figure 193.

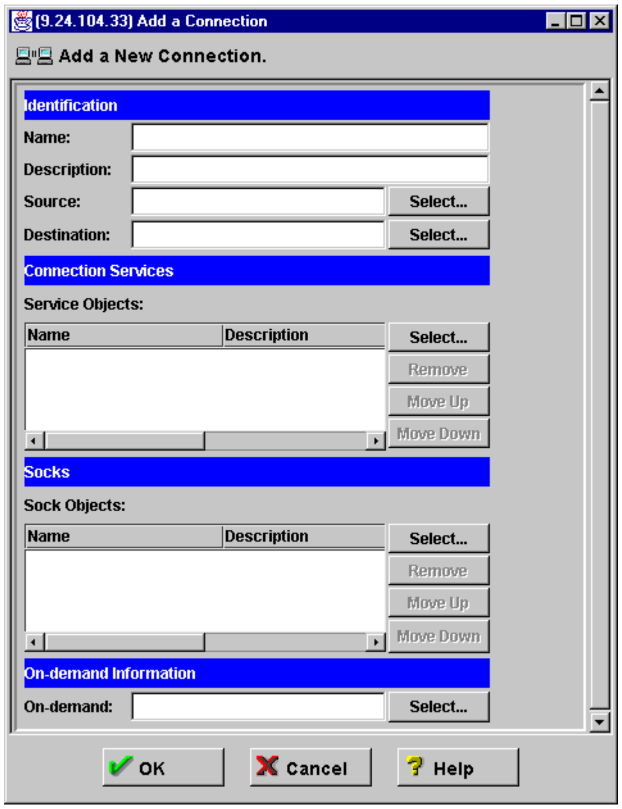


Figure 193. A connection definition: making it eligible for on-demand services

6.6 Detail on defining VPNs

Please refer to Appendix E, “Firewall Technologies configuration for VPN support” on page 347 for step-by-step details of setting up a VPN for the three scenarios we have described in this chapter.

Chapter 7. Security scenarios simplified

Thinking of connecting your z/OS host to the Internet? Or, perhaps expanding your S/390 environment to support more IP-based applications? This chapter provides you with a series of high-level example networks. Each example presents some reasonable security issues that should be considered. The examples, obviously, are not an exhaustive list; neither is the list of security considerations. The intention of these examples it to give system administrators an idea of the range of scenarios and protection levels available.

This book is entirely devoted to security on the network, just as there are entire books devoted to protecting your own home. However, to keep things in perspective, unless you are a really big target, the odds are more likely that your home will be broken into than your network will be hit by a malicious hacker. The problem is, what if you leave your front door unlocked or a window open? What if someone creeps into your home and helps him or herself? The answer is -- with a little bit of effort, you might as well lock the doors.

7.1 Batten down the hatches

All of the following examples provide a list of what type of features are recommended. Each scenario tries to provide different combinations with generally increasing complexity as we move through the chapter. It is highly unlikely that any of these scenarios will meet your needs exactly. One of the most important questions to consider when planning security is:

What services should be allowed?

Allow only those connections and applications that are required. If your users don't need telnet through your firewall, then don't allow it. A good approach is to disallow everything and then incrementally add each service as it is required. The primary vehicle for this is port access control, whether via firewall filtering or on the target z/OS host itself. It is probably better to have a disgruntled user ask your administrator for access through your firewall than to have a hacker let you know that some unexpected loophole exists.

7.1.1 Keeping it simple

This isn't often talked about, but what about no firewall at all? What if we just hook up the z/OS host to the Internet? It might not be as disastrous as you think. Like habitually leaving your house unlocked, you could have a lifetime of trouble-free Internet presence.

Let's take a closer look. Imagine that the only service you want to use is TN3270 (enterprise users, we'll call them). If you wanted more services than just TN3270, you'd probably be willing to do more security too! We see this scenario in Figure 194 on page 302. This isn't quite as absurd as it might seem -- you do similar things each time you dial your home workstation directly to the Internet. The only difference from a technical perspective is that the z/OS host will be listening on port 23 for incoming connections. Normally your home workstation doesn't act as a server.

Advantages:

- Simple. Really simple.
- RACF is already configured.
- Your TN3270 clients probably already support SSL for the TN3270 server (and if you're willing to take the time to create client certificates, you could authenticate all incoming clients). See 2.4, "Secure Sockets Layer" on page 36 for more information.
- One of the quickest ways to implement traffic filtering on z/OS is via the policy agent. See 5.10.3, "Policy agent and security" on page 210 for details. However, traffic filtering is all that the policy agent can do. There are good reasons to use the firewall for filtering. Most notably, as network security needs grow, IPSec will likely become a desirable tool.
- TRMD -- this server is just too good not to implement in these heady days of denial-of-service attacks. TRMD builds upon existing DoS prevention already present in z/OS by dynamically controlling potential resource overallocation (connection flooding), whether intentional or accident. You'll find more information on TRMD in 5.11, "Traffic Regulation Management (TRM) security" on page 224. Policy agent is required to support TRMD.

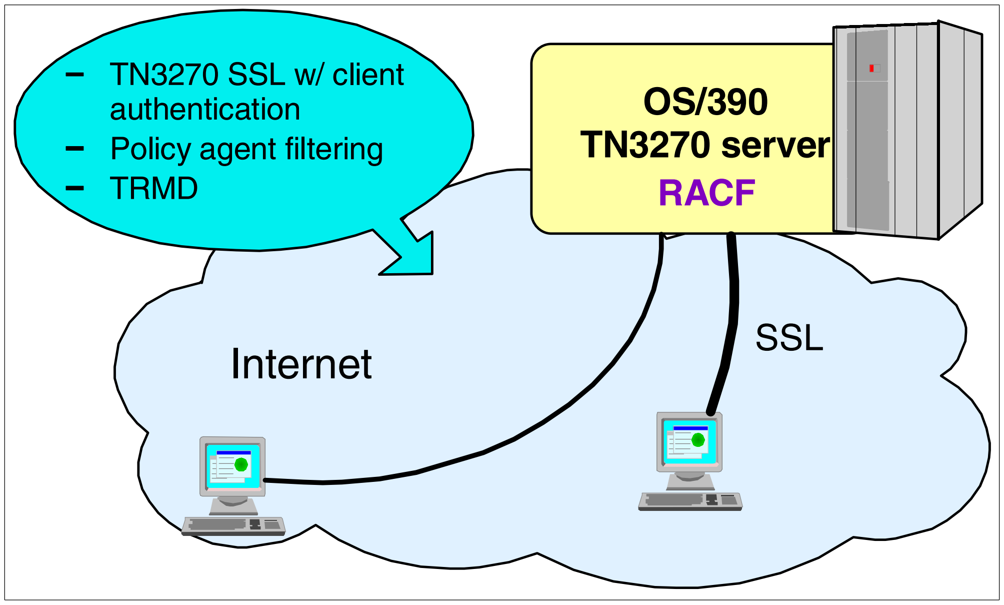


Figure 194. Giving it all away (almost) on the Internet

This configuration isn't as unreasonable as it at first appears. With client authentication in place, there is a considerable amount of control. And, if you don't already have a firewall configured, this is the fastest way to be reasonably safe and accessible to the Internet.

Important

If you are sending a RACF password across the Internet, the password must be secured (SSL or IPSec). There are no exceptions.

A word needs to be said about what may appear to be unusual usage of the z/OS UNIX System Services policy agent. To put things simply, policy agent can be used to control the amount of a certain type of traffic -- for example, it can control the amount of telnet (TCP) traffic over port 23. By setting Maxconnections to a value of zero, the policy agent effectively can block (filter) any type of traffic.

Note, however, that the policy agent can only function as a filter on a destination node -- it cannot function in this capacity as an intermediate node (router).

7.1.2 But we already have a firewall

Let's look at something a little more advanced. Most likely, you already have an active configured firewall serving to protect your internal network from the Internet. All you want to do is allow those faithful workers (enterprise users) to do a little overtime on the weekends without making them commute to the office. Let's say some of these users might want to pull a file off your z/OS host so they can work locally (disconnected). This means we'll need FTP access as well. However, an assumption for this scenario is that no public (that is, non-employee) access is necessary.

A reasonable solution to such a situation might be to use SSL for both TN3270 and FTP. The FTP client and server available in z/OS V1R2 and later support TLS/SSL. A possible network layout is illustrated in Figure 195.

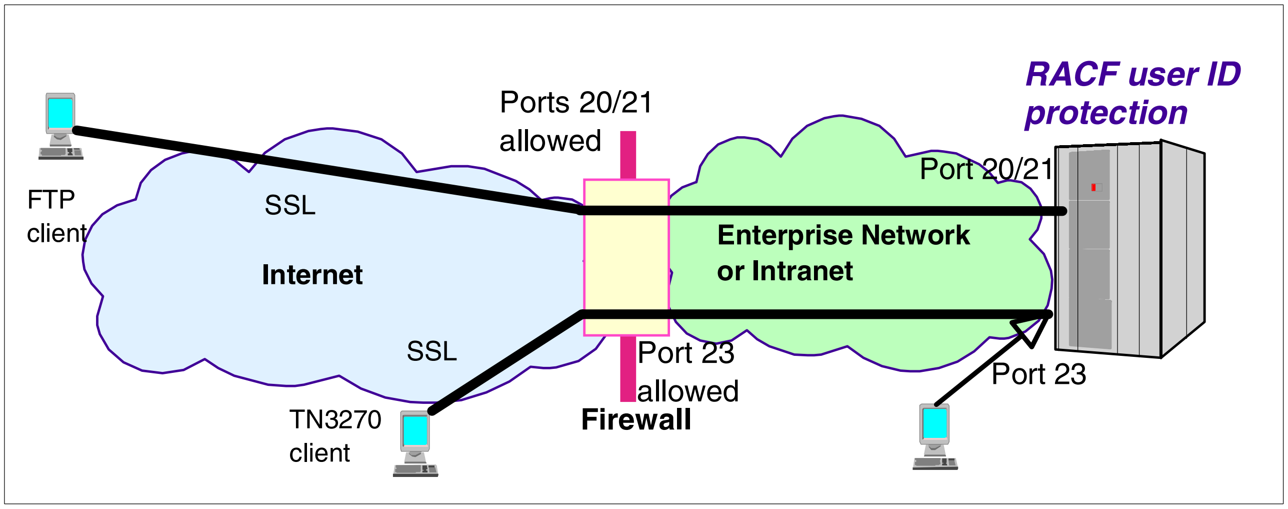


Figure 195. A reasonable and simple secure setup for Internet access

The basic pieces to be implemented to support this configuration are:

- Implement all controls that we did for our fledgling network in Figure 194 on page 302.
- If TLS/SSL cannot be negotiated on the connection, the z/OS host will fail the connection request. You don't want your users connecting without encryption!
- TN3270 and FTP clients must be SSL capable (since the clients are all employees, this is conceivable).
- The firewall allows inbound connections on ports 20, 21,and 23.
- Ephemeral port outbound requests must be permitted for the data portion of the FTP transfers requested by clients on the Internet.

- For FTP sessions initiated from within the enterprise network, the FWFRIENDLY option would be needed to allow data connection establishment via ephemeral port outbound requests. See 5.4.7, “FTP firewall-friendly” on page 157.

This scenario is surprisingly simple, and not too bad in terms of security. But, there is always more that can be done to make life more frustrating for your not-so-friendly hacker.

7.1.3 I want more security

Building on our latest simple secure setup, let’s see what simple changes could be made to improve security. We’ve illustrated these in Figure 196. Here are the relatively simple changes an administrator could make to enhance the strength of security.

- Sometimes, you not only have to protect the system from malicious hackers, you also have to prevent inadvertent compromise by authorized users. Using NETACCESS, you can control what networks any user ID on your z/OS host can access. If, for example, a TN3270 user wanted to use FTP (or any other IP based protocol) to access the Internet, NETACCESS could be used to restrict this capability. For more information, see 4.1, “Network access control” on page 91.

Note

NETACCESS applies only to traffic originating on the z/OS host. It will not have any effect on inbound connections (that’s your firewall’s job anyway). If your z/OS host is functioning as a router, this feature will have no effect on routed traffic.

- If you have IP applications that you don’t want accessed from the Internet, you might be able to use the RACF APPL class to limit which users can access an application (assuming you have particular user IDs coming from the Internet). Further information can be found in 5.4.4, “APPL class and FTP” on page 150.
- Assuming your network design is simple enough, you can avoid router sabotage by using static routes between the routing firewall and your internal routers. If static routing isn’t feasible, see 5.14, “Routing protocol security” on page 246 for a secure alternative.
- Client authentication is still a good idea for any enterprise users that you want to have access to your system. Obviously, this can be handled via SSL. However, this places control of authorized users deep inside your network at the enterprise server. The possibility exists to do authentication via IPSec AH (Authentication Header). IPSec authentication can provide authentication of Internet-based clients. Note that SSL could still be used for encryption -- IPSec merely confirms identity. The advantage of this method is that authorization occurs at the entrance to your network instead of at the destination enterprise server.

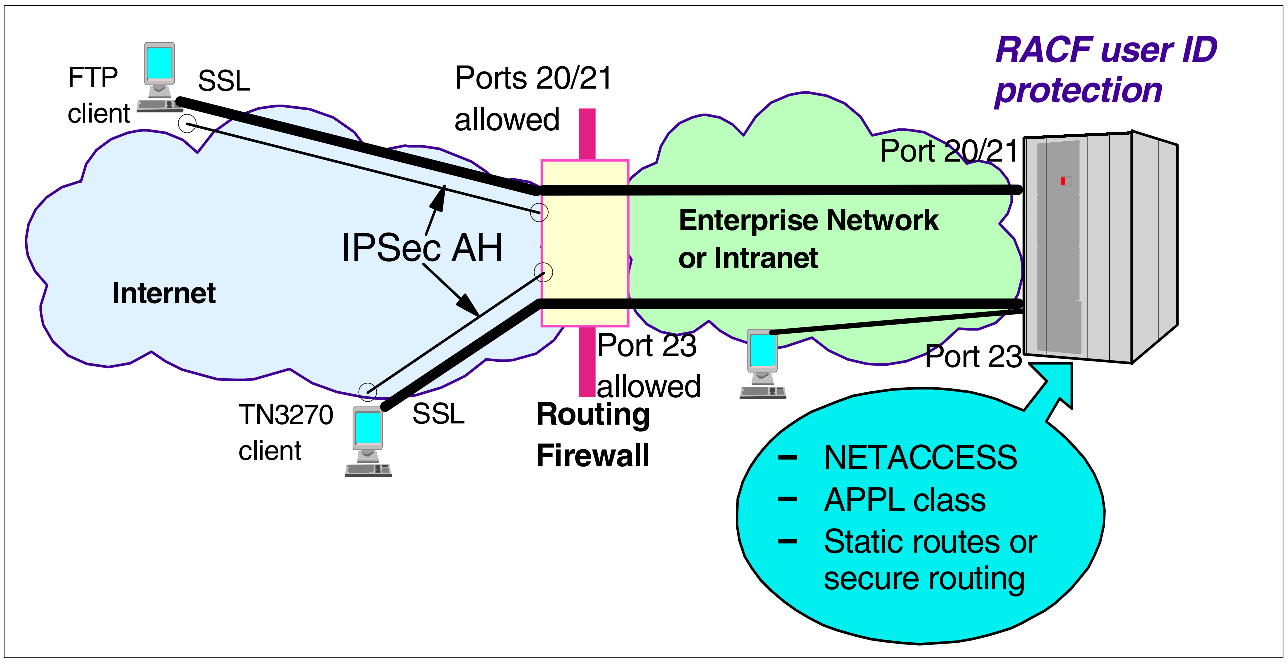


Figure 196. Tightening things up using z/OS security features

There are some other things to consider as we move in the direction of a more secure layout: don't give any user IDs, particularly those being used from the Internet, any more privilege than they absolutely require. Use RACF to control as much as you can. Sure, RACF is big and a little intimidating, but that is all the more reason to have it on your side.

Overall this layout is relatively easy to implement and something that many shops would feel comfortable with -- not unlike having installed good locks on the doors and windows of your home.

7.1.4 Open for business

If you notice, none of the earlier configurations were providing public access. What if your company wants a Web site? How about anonymous FTP service? Such requirements change things considerably in terms of both the security required and the network topology that should be used.

Our goal now is to maximize all of the network security options that are available. To do so, we'll begin by organizing the network topology to match what is considered to be a strongly secure layout. Of course, each topology presented is one instance of many possible configurations. Not all possibilities can be shown, nor can these topologies lay claim to being the best. Take a long look at Figure 197 on page 306. Our simple network days are rapidly fading.

Note that this network diagram is drawn from a network topology perspective. At this time, we'll assume that the two z/OS images are in separate sysplexes. However, there is no reason why the couldn't be in the same CEC (Central Electronics Complex). In fact, as you'll see shortly, it is ideal to have these to LPARs in the same machine. We'll go into more detail on this in 7.2, "Security and the S/390 platform" on page 307.

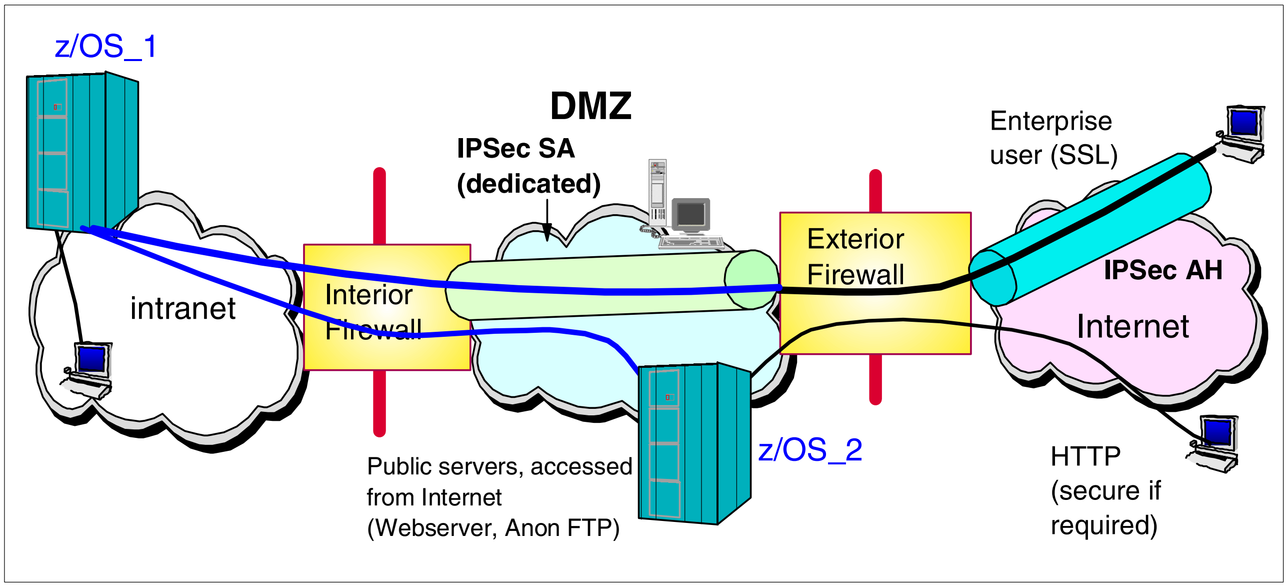


Figure 197. A secure topology with public and enterprise access

Let's examine the features and improvements of this new topology:

- Let's assume we've placed all of the controls on the internal enterprise networks that we did in our previous two sections, with one exception: policy agent is probably no longer necessary as a filter control. The firewalls can accomplish this for us. There are obviously other reasons why you may want to keep the policy agent active.
- Notice that we have two distinct firewalls. The exterior firewall is the one your company has always depended upon. The interior firewall is to provide protection for your internal enterprise network.
- The DMZ (demilitarized zone) is a self-contained LAN segment containing any servers you want to have public access. It is important that the DMZ be self-contained -- if a hacker makes it through your external firewall, you don't want him or her to have access to your enterprise LAN. See 2.2.2.4, "The demilitarized zone" on page 28.
- Internet clients could use TLS/SSL for secure transactions with your public servers, depending upon the sensitivity of your transactions.
- LPAR2 could be used to function as an anonymous FTP server (an excellent application of the z/OS host from a security perspective -- see 5.4.6, "FTP's anonymous levels" on page 151).
- TRMD should be running on your "destination" servers -- LPAR1 and LPAR2.
- In order for enterprise users to access the intranet from the Internet, the following is recommended (there are alternatives, of course):
  - TLS/SSL should be required between the Internet client and the destination server and IPSec AH should be used for client authentication. Alternatively, IPSec ESP could also be used. See 2.3.1, "IPSec" on page 30 for more information on IPSec.
  - Connections from a server in the DMZ zone to the interior firewall (and all the way to the target application, if desired) could use IPSec -- see 6.5, "IPSec in Firewall Technologies" on page 272.

- For accessing the enterprise server, a dedicated link between the exterior firewall and the interior firewall should exist. Authentication of this link is required; only the exterior firewall can make this link. Depending upon security requirements, encryption on this link is optional. For overall details on what sort of capabilities and configuration options are available, see Chapter 6, “SecureWay Security Server Firewall Technologies” on page 253.

7.2 Security and the S/390 platform

Let’s take a closer look at our two S/390 LPARs in Figure 197. Let’s assume that these z/OS images are part of the same CEC. Further, let’s take advantage of z/OS firewall technologies. In our last scenario, we left our S/390 server sitting behind the interior firewall. However, with z/OS firewall technologies running, why not bring the z/OS host forward onto the DMZ? Don’t be alarmed -- using firewall technologies, we will still have our enterprise system safely behind a firewall. As we shall see, this allows us to take advantage of some of the unique S/390 and z/OS features.

Take a look at Figure 198 on page 308. Note that this is simply a different way of viewing the same network that we’re already working with. The only change is that we now have two interior firewalls -- one running on z/OS protecting your enterprise server, and the other protecting your intranet as before.

Here are some important points to consider in this layout:

- One IP stack should be devoted to functioning as a firewall, while the other would function as your normal internal IP services stack, which would also have the firewall functionality, IPSec, and IP filtering. Additionally network access can be used. See 4.1, “Network access control” on page 91 for more details.
- This topology could be achieved with only one shared OSA-Express card.
- Note the communication “cookie” at the heart of Figure 198. These are the preferred methods (not the only methods) of inter-LPAR IP based communication. If the LPARs are within the same CEC, iQDIO can be used as a lightning-fast link between LPARs. iQDIO stands for Internal Queued Direct I/O (sometimes called Hipersockets). The link appears as an XCF link, but iQDIO is used when possible (that is, if the LPARs share memory, iQDIO will be used instead of XCF). iQDIO is only available on zSeries hardware using z/OS V1R2 and newer.
- If an OSA-Express card is present, QDIO may be used to similar effect using an MPCIPA link.
- A word of caution: obviously, these two LPARs should exist in separate sysplexes. The firewall on the DMZ LPAR allows public access. LPAR1, the enterprise LPAR with firewall functionality, should only allow IPSec authenticated connections from the exterior firewall.
- Again, we use a dedicated link between the exterior firewall and LPAR1’s firewall. As in Figure 197, IPSec AH should be used to authenticate that only the exterior firewall has access.
- The only applications (ports) allowed through the enterprise firewall (LPAR1) are TN3270 and FTP that are authenticated using IPSec AH. All other

applications should be disabled for access originating on the DMZ side of this firewall.

- Data flowing across the iQDIO connection cannot be sniffed.

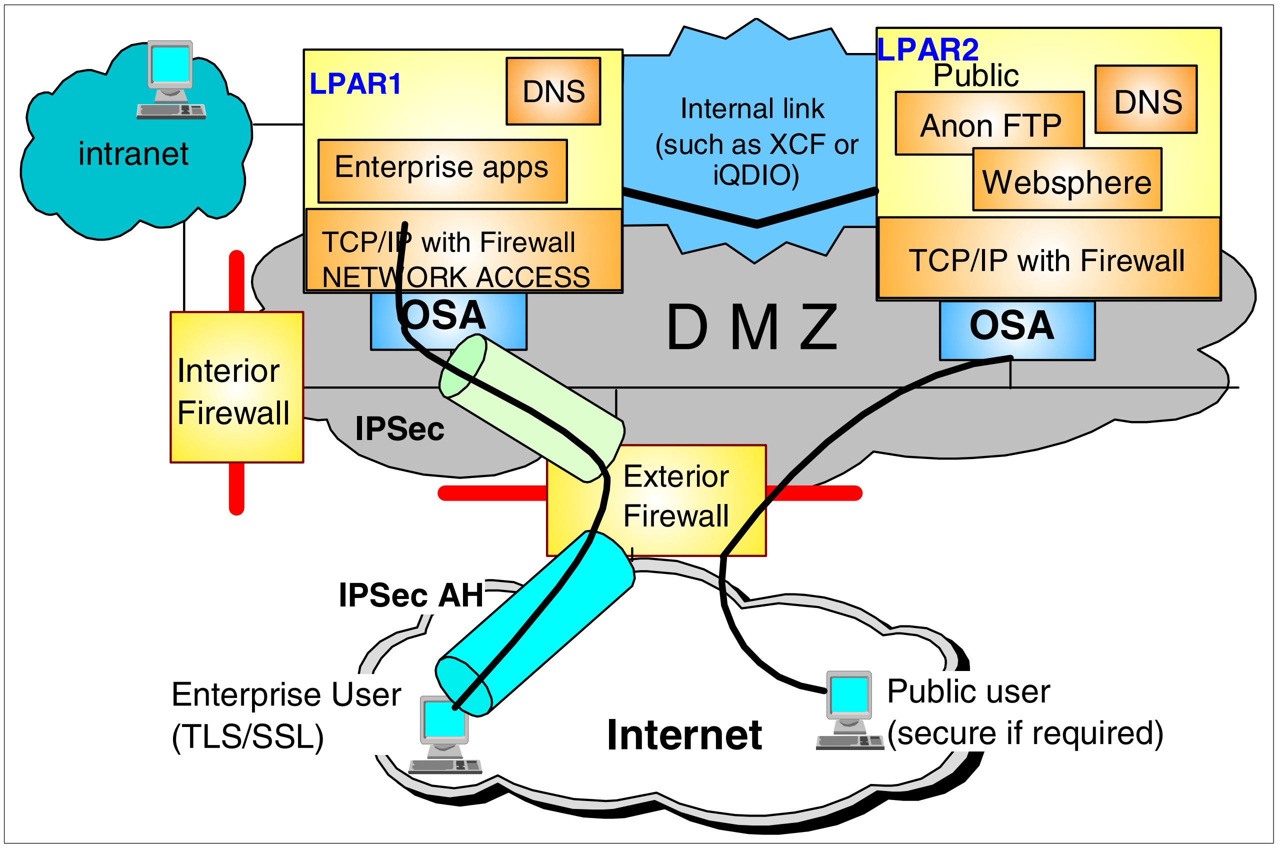


Figure 198. The S/390 view of our secure topology

The topology of Figure 198 is about as far as you might be able to go with most hardware platforms today. However, there is an aspect of network security we haven't directly addressed. That is, we need redundancy in order to provide a fail-safe set of services. The S/390 itself is an environment that is highly unlikely to fail, but highly unlikely is not always good enough. In addition, even with all this network security, the possibility of a hacker damaging or crippling any of these LPARs is there. Why not exploit the S/390 and z/OS capabilities further?

7.2.1 Adding backup capability

Since we already have a sysplex environment, we know that sysplex messaging can be used to dynamically establish links between all hosts in a sysplex. TCP/IP on z/OS is capable of dynamically establishing IP connections over these XCF links. In addition, TCP/IP is capable of using sysplex messaging to determine whether a VIPA (Virtual IP Address) is active and available in the sysplex. Putting it all together, TCP/IP in a sysplex can dynamically determine when a sysplex host or stack goes down and can dynamically regenerate the lost VIPA address on another backup host in the sysplex. When the primary host is revived, the VIPA address automatically returns to the original TCP/IP stack in a non-disruptive fashion. The result is seamless (almost) fail-safe availability of S/390 servers and z/OS applications.

For complete details on dynamic VIPA, see the chapter on VIPA in *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 1: Configuration and Routing*, SG24-5227.

For an example of how this network might appear, see Figure 199.

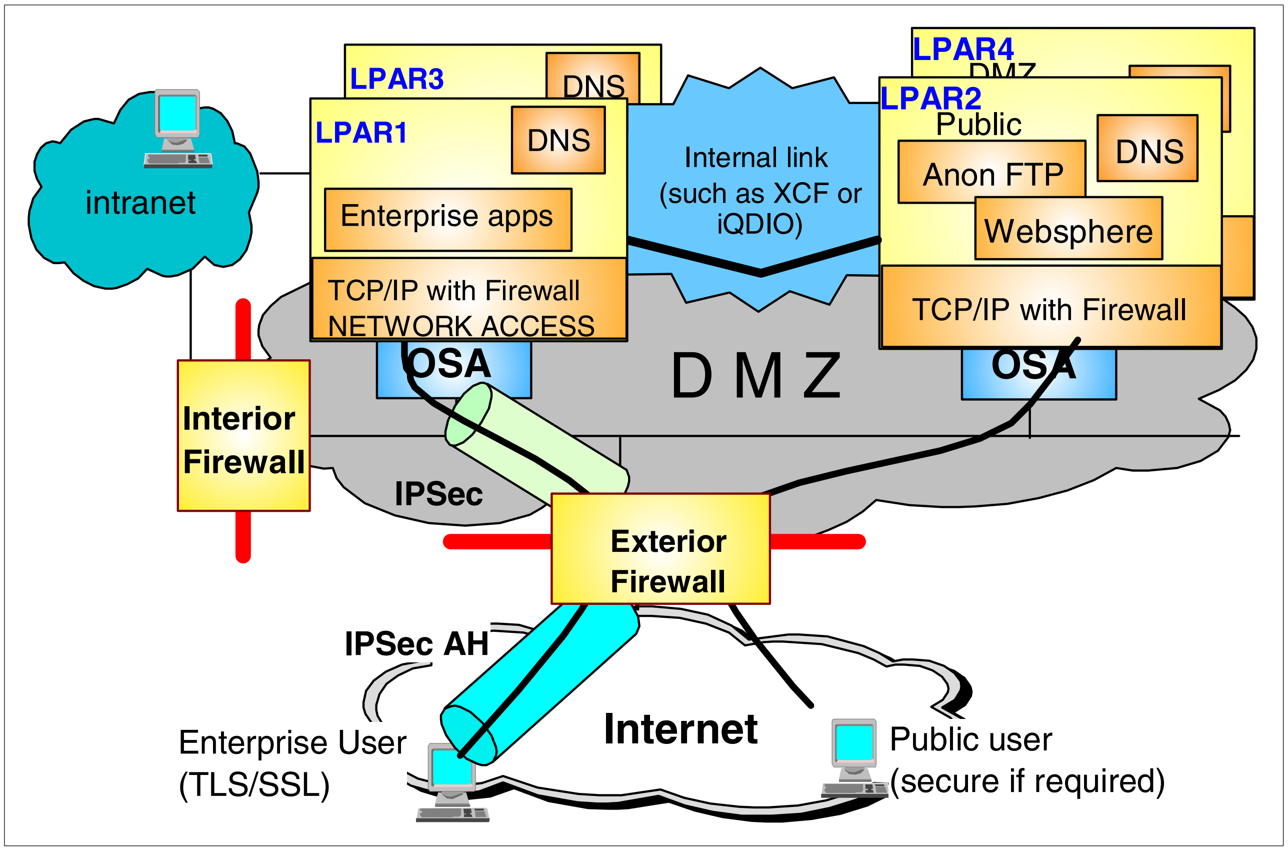


Figure 199. Using automatic VIPA backup capabilities

In Figure 199, we've indicated only one backup image for each S/390 server in our network. This need not be the case -- multiple backup images can be configured. In addition, the VIPABACKUP statement is hierarchical. Backup LPARS can be configured in a cascading fashion such that your most desirable backup LPAR would be selected first. If that LPAR fails also, a third LPAR (perhaps even a test LPAR) can be configured to take over the workload.

As mentioned before, each of these LPAR pairs must exist in their own sysplex.

Any server that can function as a backup for another server must have the appropriate applications active on these backup stacks. Of course, these backup applications would normally be invisible and unreachable to your Internet customers because firewall filter rules would allow access only to the VIPA address you expect these applications to be associated with.

Some applications, such as IBM HTTP Server, can be configured to bind to a specific address. VIPA addresses can be configured to automatically become active when a request is made by an application to bind to them. This may be another method that customers want to use to maintain application availability.

7.2.2 Adding sysplex distributor

Why stop with simple stand-by backup? It is quite possible that your z/OS DMZ application server is going to see a considerable amount of traffic. Another possible scenario is illustrated in Figure 200.

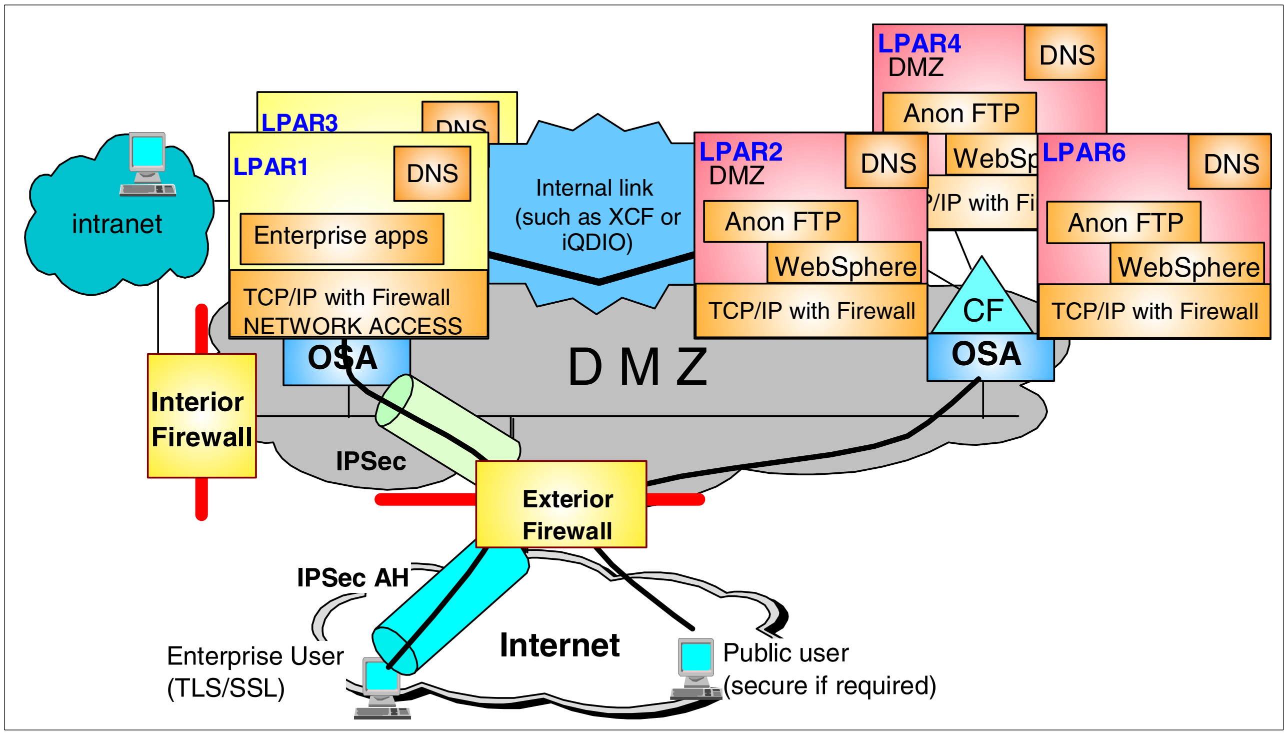


Figure 200. Adding some sysplex distributor capability

With this scenario, we take advantage of sysplex distributor. This is the first scenario where dynamic XCF must be used. In order to utilize sysplex distributor, DYNAMICXCF must be coded for TCP/IP on LPAR2, LPAR4 and LPAR6. For more information about sysplex distributor, see the appropriate chapter in *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 1: Configuration and Routing*, SG24-5227.

Using sysplex distributor as the public server has some fantastic advantages. Load balancing will occur across all three (or more) LPARs. Availability is maintained, since each server in the cluster can act as a backup to the others.

Finally, using sysplex distributor as a public server gives scalability that most shops can only dream of -- virtually unlimited horsepower can be added, even on the fly (via non-disruptive addition of LPARs to the sysplex).

7.3 Intranet security

An intranet is a collection of computers connected through a network, LAN and/or WAN, that belongs to a certain organization and uses Internet technologies for communication inside it.

Some of the defining qualities of an intranet are as follows:

- Uses IP for all platforms. Other protocols may exist (NetBIOS, for example).

- Uses the same applications and services as on the Internet.
- Uses Web technology.
- Is integrated with current systems such as enterprise databases.

Outwardly, an intranet can be likened to a miniature private Internet. However, this is an inaccurate simplification. Intranet sites tend to offer a different type of information from Internet sites (such as the servers sitting in your DMZ). The Internet is aiming at information that is to be shared with the public; the intranet often has information that is of interest only to people within a corporation. Instead of the glossy “brochure-ware” of an Internet site, an intranet site tends toward Human Resources information, internal project tracking, employee expenses, and insurance forms. Obviously, such information not only needs to be secure from outside intruders, but it most likely should be secured from unauthorized internal access. Blocking outsiders, although not easy, has been discussed in the preceding sections. The focus of an intranet scenario becomes how and when to restrict access.

7.3.1 Why intranet security?

The requirements for intranet security depend entirely upon the information present on the intranet and the heterogeneity of the user community. The issue of whether or not the intranet user community (that is, your employees) can be trusted will not be considered in this discussion. Such a matter is primarily an issue of corporate culture.

Let’s take an example of a homogeneous group of intranet users. If your intranet consists of a LAN containing a group of code writers and nothing else, there likely isn’t much need for security. The programmers are likely accessing a centralized server containing various build levels of their projects. Access control, if any is required, would be by programming group. Although the information present on such an intranet might need extreme protection from external sources, the actual intranet security requirements are relatively limited. If programmer Mick wants to look at programmer Keith’s code, it is likely that such information sharing wouldn’t require intranet hacking. Presumably, Mick could tap Keith on the shoulder and ask for a peek. Security requirements in this situation are minimal.

Now, let’s take a look at another homogeneous group; this time, the group is a financial services team. It is conceivable that every user on such a network requires complete confidentiality of his files. If Charlie taps Ron on the shoulder and asks to look at one of Ron’s portfolios, it is possible the request would be denied, or deferred to a higher authority. Client information would be shared on a need-to-know basis only. On such a network, encryption (privacy) of data is possibly a significant concern.

Next, we have a legal department intranet. Here, privacy of data would obviously be a high priority. However, it is likely that access to client files would be governed by the legal requirement of counsel-client confidentiality. With so much at stake, it is likely that such an intranet would want encryption and user authentication.

It is likely that an intranet would have a combination of the above three scenarios. When you mix such user groups, the confidentiality requirements obviously increase.

7.3.2 Intranet security is a good idea, but how?

Intranet security is essentially a subset of Internet security. Firewalls and bastions, although potentially useful, are unlikely to be critical. The focus narrows to the four pillars of Internet security: privacy, authentication, integrity, and non-repudiation. TLS/SSL and IPSec can be used in the intranet for network security. See Figure 201 for an example.

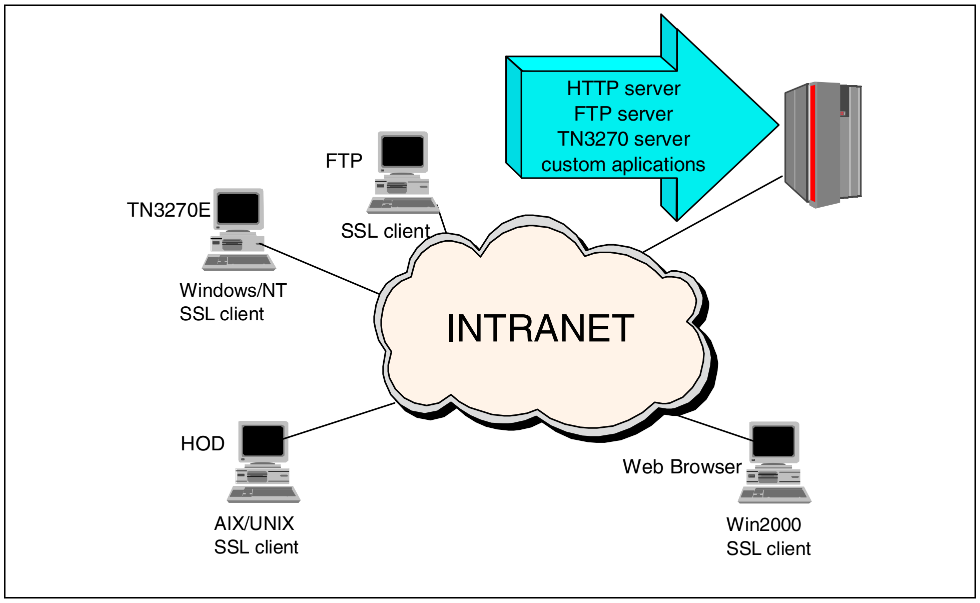


Figure 201. Intranet scenario

Most IP security literature stresses the importance of privacy, hence the need for encryption. In actual fact, there are some significant barriers to being able to sniff (that is, trap or trace) data on the Internet at large. Many of these limitations are physical -- how can a hacker get access to the machines that I'm using on the Internet? However, in the intranet situation, the physical proximity of data makes sniffing of data a simple task to even a novice hacker. Privacy of data on an intranet could therefore be considered a more significant threat than on the Internet.

Next, the intranet is an area that is completely within your company's control (or should be). As such, client authentication again becomes feasible. Certain groups of user (your legal department, for example) might be required to use client authentication as proof of identity.

7.3.3 Custom applications

If a requirement for a custom-written IP application exists, then TLS/SSL might be the first choice for privacy. To meet security requirements, the application could be upgraded to support TLS/SSL. Of course, the z/OS environment contains a complete SSL application programming interface. See the C/C++*OS/390 System Secure Sockets Layer Programming Guide and Reference*, SC24-5877 for complete details.

Alternatively, an installation may choose to implement VPN with applications that don't support TLS/SSL.

Appendix A. Sample RACF definitions

Here you will find samples of the RACF definitions commonly used to run a secure TCP/IP environment. We assume that you have set up the OS/390 environment as described in *OS/390 UNIX System Services Planning*, SC28-1890. You should have a RACF group OMVSGRP and a user OMVSKERN with UID(0). These default names can be changed, if you wish, via the /etc/options and /etc/profile files. We would recommend that you keep the names suggested in the planning guide for user IDs and groups; setting up an OS/390 UNIX System Services environment is non-trivial and using the same names just makes it easier without compromising security.

For the complete description of each RACF command, consult *OS/390 Security Server (RACF) Command Language Reference*, SC28-1919.

A.1 RACF settings for UNIX System Services

- Define a superuser with a user ID of BPXROOT.

  ```
  ADDUSER BPXROOT DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/') PROGRAM('/bin/sh'))
          NOPASSWORD
  ```

  If in BPXPRMxx PARMLIB member SUPERUSER(BPXROOT). It is also the default.

- Activate the FACILITY class (if it is not defined).

  ```
  SETROPTS CLASSACT(FACILITY) GENERIC(FACILITY) AUDIT(FACILITY)
  SETROPTS RACLIST(FACILITY)
  ```

- Create a BPX.SUPERUSER FACILITY class profile.

  ```
  RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)
  SETROPTS RACLIST(FACILITY) REFRESH
  ```

- Alter/add users to have OE superuser permission.

  ```
  ALTUSER user1 OMVS(UID(7) HOME('/u/user1') PROGRAM('/bin/sh'))
  PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(user1) ACCESS(READ)
  SETROPTS RACLIST(FACILITY) REFRESH
  ```

- Set up for a OE default user and group.

  ```
  ADDGROUP OEGROUP OMVS(GID(17))
  ADDUSER OEDFLT NAME('USS DFLTUSER') OMVS(UID(88) HOME(/u/OEDFLT)
  RDEFINE FACILITY BPX.DEFAULT.USER APPLDATA('OEDFLT/OEGROUP')
  SETROPTS RACLIST(FACILITY) REFRESH
  ```

A.2 RACF settings for TCP/IP applications

- Setbup the STARTED class (if it is not defined).

  ```
  SETROPTS GENERIC(STARTED)
  SETROPTS CLASSACT(STARTED) RACLIST(STARTED)
  ```

- Define the TCP/IP system address space started task user ID with a UID=0.

  ```
  ADDUSER tcpip_user DFLTGRP(OMVSGRP) OMVS(UID(0) HOME(/) PGM(/bin/sh))
  ```

- Assign an OMVS segment to started task user IDs specified as UID(0): (ADD or ALTUSER).

```
ADDUSER webserver_user_id DFLTGRP(OMVSGRP) OMVS(UID(0) HOME(/)
        PGM(/bin/sh))
ADDUSER ftpd_user_id DFLTGRP(OMVSGRP) OMVS(UID(0) HOME(/) PGM(/bin/sh))
ADDUSER SYSLOGD DFLTGRP(OMVSGRP) OMVS(UID(0) HOME(/) PGM(/bin/sh))
```

- Define started resource profiles.

```
RDEFINE STARTED TCPIP.* STDATA(USER(tcpip_user) PRIVILEGED(NO) TRUSTED(NO)
        TRACE(NO))
RDEFINE STARTED FTPD.* STDATA(USER(tcpip_user) PRIVILEGED(NO) TRUSTED(NO)
        TRACE(NO))
RDEFINE STARTED IMWEBSRV.* STDATA(USER(webserver_user_id) PRIVILEGED(NO)
        TRUSTED(NO) TRACE(NO))
```

Change `TRACE(NO)` to `TRACE(YES)` if you want to see a message indicating what STARTED profile was used to start the started task.

- Enable OROUTED and OMPROUTE to get started.

```
RDEFINE OPERCMDS (MVS.ROUTEMGR.OMPROUTE) UACC(NONE)
PERMIT MVS.ROUTEMGR.OMPROUTE ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
RDEFINE OPERCMDS (MVS.ROUTEMGR.OROUTED) UACC(NONE)
PERMIT MVS.ROUTEMGR.OROUTED ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

A.2.1 RACF configuration for OS/390 UNIX level security

- Define the BPX.DAEMON facility class profile in RACF.

```
RDEFINE FACILITY BPX.DAEMON UACC(NONE)
```

- Allow user IDs access to the BPX.DAEMON facility class resource.

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(OMVSKERN) ACCESS(READ)
PERMIT BPX.DAEMON CLASS(FACILITY) ID(tcpip_user) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

- Define the BPX.SERVER facility class profile and give the Web server the UPDATE permission.

```
RDEFINE FACILITY BPX.SERVER UACC(NONE)
PERMIT BPX.SERVER CLASS(FACILITY) ID(webserver_user_id) ACCESS(UPDATE)
SETROPTS RACLIST(FACILITY) REFRESH
```

- Set up for SURROGAT class for the PUBLIC user and allow the Web server to access it.

```
SETROPTS CLASSACT(SURROGAT)
SETROPTS RACLIST(SURROGAT)
RDEFINE SURROGAT BPX.SRV.PUBLIC UACC(NONE)
PERMIT BPX.SRV.PUBLIC CLASS(SURROGAT) ID(webserver_user_id) ACCESS(READ)
SETROPTS RACLIST(SURROGAT) REFRESH
```

- Define all other BPX.* facility class profiles and permit, for example, the superuser to set APF and PROGCTL.

```
RDEFINE FACILITY BPX.DEBUG UACC(NONE)
RDEFINE FACILITY BPX.DEFAULT.USER UACC(NONE)
RDEFINE FACILITY BPX.FILEATTR.APF UACC(NONE)
RDEFINE FACILITY BPX.FILEATTR.PROGCTL UACC(NONE)
RDEFINE FACILITY BPX.SMF UACC(NONE)
RDEFINE FACILITY BPX.STOR.SWAP UACC(NONE)
RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)
RDEFINE FACILITY BPX.WLMSERVER UACC(NONE)
```

```
PERMIT BPX.FILEATTR.APF CLASS(FACILITY) ID(superuser) ACCESS(READ)
PERMIT BPX.FILEATTR.PROGCTL CLASS(FACILITY) ID(superuser) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

- Activate program control (if not already active) to identify programs as controlled. Add DATASET profiles only if necessary.

```
SETROPTS WHEN(PROGRAM)
ADDSD 'cee.SCEERUN' UACC(READ) GENERIC
ADDSD 'websrv_hlq.SIMWMOD1' UACC(READ) GENERIC
ADDSD 'SYS1.LINKLIB' UACC(READ) GENERIC
ADDSD 'TCPIP_hlq.SEZALINK' UACC(READ) GENERIC

RDEFINE PROGRAM * ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('cee.SCEERUN'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('websrv_hlq.SIMWMOD1'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('TCPIP_hlq.SEZALINK'//NOPADCHK) UACC(READ)
```

If you need to define a firewall, issue the following commands as well:

```
RALTER PROGRAM * ADDMEM('ICA.SICALMOD'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

We omitted the VOLSER parameter from the ADDMEM clauses; it is no longer required.

If you code six asterisks within single quotation marks (`'******'`), this refers to the current SYSRES volume, which might be useful in a sysplex environment.

A.2.2 RACF definitions to control the use of the TCP/IP operator commands

- Set up for the VARY.TCPIP command.

```
SETROPTS CLASSACT(OPERCMDS) GENERIC(OPERCMDS)
SETROPTS RACLIST(OPERCMDS)
RDEFINE OPERCMDS (MVS.VARY.TCPIP.**) UACC(NONE)
PERMIT MVS.VARY.TCPIP.**) ACCESS(CONTROL) CLASS(OPERCMDS) ID(uid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

A.3 Required RACF definitions to get Firewall Technologies started

- Define user and group and create the FWKERN.START.REQUEST resource profile:

```
ADDGROUP FWGRP SUPGROUP(SYS1) OMVS(GID(nnn))
ADDUSER FWKERN OMVS(HOME('/u/fwkern/') UID(0))
        DFLTGRP(FWGRP) AUTHORITY(CREATE) UACC(ALTER) NOPASSWORD
RDEFINE FACILITY FWKERN.START.REQUEST UACC(NONE)
PERMIT FWKERN.START.REQUEST CLASS(FACILITY) ID(FWKERN) ACCESS(UPDATE)
RDEFINE STARTED FWKERN.* STDATA(USER(FWKERN))
SETROPTS RACLIST(STARTED) REFRESH
SETROPTS RACLIST(FACILITY) REFRESH
```

The HOME directory has to be created with:

```
MKDIR '/u/fwkern' MODE(7,5,5)
```

- Permit FWKERN access to the BPX.SERVER to allow ISAKMP server.

```
PERMIT BPX.SERVER CLASS(FACILITY) ID(FWKERN) ACC(READ)
```

- Grant permission to use OCSF services to FWKERN user ID:

```
PERMIT CDS.CSSM CLASS(FACILITY) ID(FWKERN) ACC(READ)
PERMIT CDS.CSSM.CRYPTO CLASS(FACILITY) ID(FWKERN) ACC(READ)
PERMIT CDS.CSSM.DATALIB CLASS(FACILITY) ID(FWKERN) ACC(READ)
SETROPTS CLASS(FACILITY) REFRESH
```

- Permit FWKERN access to start the servers and access the TCPIP data sets.

```
RDEFINE STARTED ICAPSLOG.**  STDATA(USER(FWKERN) GROUP(FWGRP))
RDEFINE STARTED ICAPSOCK.**  STDATA(USER(FWKERN) GROUP(FWGRP))
RDEFINE STARTED ICAPPFTP.**  STDATA(USER(FWKERN) GROUP(FWGRP))
RDEFINE STARTED ICAPCFGS.**  STDATA(USER(FWKERN) GROUP(FWGRP))
RDEFINE STARTED ICAPSTAK.**  STDATA(USER(FWKERN) GROUP(FWGRP))
RDEFINE STARTED ICAPIKED.**  STDATA(USER(FWKERN) GROUP(FWGRP))
PERMIT 'TCPIP.**' ID(FWKERN) ACCESS(READ)
SETROPTS RACLIST(STARTED) REFRESH
```

- Permit FWKERN to READ to BPX.SMF (for logging) and BPX.DAEMON facility class profiles.

```
PERMIT BPX.SMF CLASS(FACILITY) ID(FWKERN) ACCESS(READ)
PERMIT BPX.DAEMON CLASS(FACILITY) ID(FWKERN) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

- All user IDs that will use the firewall configuration GUI need access to ICA.CFGSRV.

```
RDEFINE FACILITY ICA.CFGSRV UACC(NONE)
PERMIT ICA.CFGSRV CLASS(FACILITY) ID(userid) ACCESS(UPDATE)
SETROPTS RACLIST(FACILITY) REFRESH
```

A.4 RACF definition to manage certificate in RACF common keyring

- Some required authorization to perform the basic RACDCERT actions:

```
RDEFINE FACILITY IRR.DIGTCERT.ADD UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.ADDRING UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.CONNECT UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.GENCERT UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.GENREQ UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.ADD CLASS(FACILITY) ID(userid) ACC(CONTROL)
PERMIT IRR.DIGTCERT.ADDRING CLASS(FACILITY) ID(userid) ACC(UPDATE)
PERMIT IRR.DIGTCERT.CONNECT CLASS(FACILITY) ID(userid) ACC(CONTROL)
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(userid) ACC(CONTROL)
PERMIT IRR.DIGTCERT.GENREQ CLASS(FACILITY) ID(userid) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(userid) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(FWKERN) ACC(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(userid) ACC(UPDATE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(FWKERN) ACC(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Appendix B. Default permissions for HFS files in OS/390 UNIX

The following table shows the default permissions set by the system:

Use	To Create a	Default Permissions

`mkdir shell command`	Directory	`owner=rwx`
		`group=rwx`
		`other=rwx In octal form: 777`

`MKDIR TSO command`	Directory	`owner=rwx`
		`group=r-x`
		`other=r-x In octal form: 755`

`JCL with no PATHDISP specified`	Directory or file	`owner=---`
		`group=---`
		`other=--- In octal form: 000`

`ISPF editor, OEDIT command, oedit command`	File	`owner=rwx`
		`group=---`
		`other=--- In octal form: 700`

`vi editor`	File	`owner=rw-`
		`group=rw-`
		`other=rw- In octal form: 666`

`ed editor`	File	`owner=rw-`
		`group=rw-`
		`other=rw- In octal form: 666`

`Redirection (>)`	File	`owner=rw-`
		`group=rw-`
		`other=rw- In octal form: 666`

`cp command`	File	Sets the output file permissions to the input file permissions.

`OCOPY command`	File	Permission bits for a new file are specified with the ALLOCATE command, using the PATHMODE keyword, prior to entering the OCOPY command. If the PATHMODE keyword is omitted, the default is:
		`owner=---`
		`group=---`
		`other=--- In octal form: 000`

`OPUT or OPUTX command`	File	For a text file:
		`owner=rw-`
		`group=---`
		`other=--- In octal form: 600`
		
		For a binary file:
		`owner=rwx`
		`group=---`
		`other=--- In octal form: 700`

Appendix C. Digital certificate formats supported by RACDCERT

The following information is based on OS/390 Security Server V2R10. Be sure to check if there is any updated information available.

The digital certificate must be in one of the following formats to be managed by the RACF `RACDCERT` command:

1. A single BER-encoded X.509 certificate.
2. A Privacy Enhanced Mail (PEM)-encoded X.509 certificate. If the input is in this format, only the Originator Certificate will be used.
3. One or more X.509 certificates contained within a PKCS #7 BER encoding. If the input is in this format, only the first certificate in the PKCS #7 encoding will be used.
4. One or more X.509 certificates contained within a PKCS #12 BER encoding.
5. A Base64-encoded certificate. The data must include this string immediately prior to the Base64 encoding:

   ```
   -----BEGIN CERTIFICATE-----
   ```

   The data must also include this string immediately following the Base64 encoding:

   ```
   -----END CERTIFICATE-----
   ```

Note the following additional details regarding RACDCERT's certificate processing:

1. All fields as defined for X.509 Version 1 certificates must be present and must have a length greater than zero (non-null).
2. X.509 certificates with version numbers greater than 3 are not supported.
3. Version 3 certificates with critical extensions are not supported. Noncritical extensions are ignored. Critical extensions that are supported include:
   - keyUsage - { 2 5 29 37 }
   - basicConstraints - { 2 5 29 19 }
   - subjectAltname - { 2 5 29 17 }
   - issuerAltName - { 2 5 29 18 }
   - certificatePolicies - { 2 5 29 32 }
   - policyMappings - { 2 5 29 33 }
   - policyConstraints - {2 5 29 36 }
   - nameConstraints - { 2 5 29 30 }
   - extKeyUsage - { 2 5 29 37 }
   - hostIdMapping - { 1 3 18 0 2 18 1 }
4. Subject and issuer names can contain only the following string types:
   - TF8 - TAG 12 (7-bit ASCII only)
   - PRINTABLESTRING - TAG 19
   - T61STRING - TAG 20
   - IA5STRING - TAG 22

- VISIBLESTRING - TAG 26
- GENERALSTRING - TAG 27
- BMPString - TAG 30 (ASCII Unicode only)

5. For a self-signed certificate, the total length of the subject's distinguished name must be 229 characters or less. For non-self-signed certificates, the length of the subject's distinguished name is limited to 255 characters. These lengths include the X.509 identifiers (such as C= and CN=) and the dot qualifiers.
6. If the certificate's signature is incorrect, the certificate is not added.
7. If the certificate that is being added has the same subject's distinguished name, issuer's distinguished name, and public key as an existing certificate, and the certificate being added is not a duplicate, `RACDCERT ADD` processing checks the validity dates and times of the certificate. If the certificate that is being added passes the signature and date validity checks, if the end date and time on the certificate being added is later than that of the existing certificate, and if the existing certificate is not expired, it is replaced.

Care must be taken when transporting the different certificate encodings to and from an OS/390 system. Both the BER-encoded X.509 and PKCS formats are binary formats. As such, they must be transported in their exact binary format.

Note: Do not perform any ASCII-to-EBCDIC translations on these formats.

PEM and Base64, however, are text-based protocols and should be transported as text. If transporting from an ASCII system, the ASCII-to-EBCDIC translation must be performed for the PEM format and Base64 format certificates.

The following information is displayed for each digital certificate defined from the `RACDCERT LIST` command:

- Serial number
- Issuer's distinguished name
- Label
- Certificate ID
- Status (trusted, not trusted, or highly trusted)
- Validity dates
- Private key size
- Type of private key (ICSF or DER-encoded key), or NONE if there is no private key
- Rings
- Up to 256 bytes of the subject's name, as found in the certificate itself
- Extensions, if present (specifically, keyUsage and subjectAltName)

For more information about the certificate supported by the RACF `RACDCERT` command, refer to *OS/390 Security Server (RACF) Command Language Reference*, SC28-1919.

Appendix D. OS/390 certificate management using RACF

Most SSL-enabled applications in OS/390 V2R10 are making use of the System SSL toolkit. For certificate management, two alternatives exist:

- The GSKKYMAN utility, which creates and maintains a key database as a file in the HFS. It can also create self-signed certificates and certificate requests for other CAs.
- The RACF command `RACDCERT`, which creates and maintains keyrings that are stored in the RACF database. This command can also be used to create self-signed certificates and certificate requests for other CAs.

Using RACF keyrings is the preferred method because it provides better security for the certificates and their private keys. With RACF keyrings, stash files containing key database passwords are not used and access to keyrings and certificates is controlled by RACF.

In this appendix, we show both methods of creating and managing certificates.

Table 29 summarizes all applications that make use of the certificate management tools in OS/390 V2R10.

Table 29. Applications that use digital certificates in OS/390 V2R10

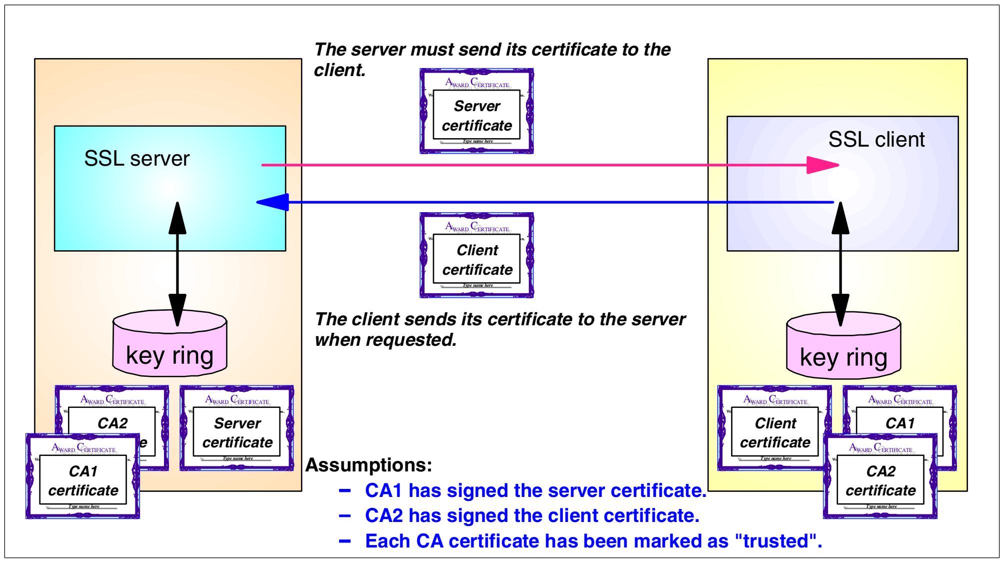
		RACDCERT	GSKKYMAN
SSL	TN3270 server	X	X
	HTTP Server	X	X
	LDAP server	X	X
	Policy agent		X
	DCAS server	X	
	Firewall configuration client		X
IKE	IKE server	X	

D.1 How certificates are used to establish secure environment

The use of certificates is required to establish an SSL connection, because the SSL procedure requires that an SSL server send its certificate to an SSL client, its communication peer. However, the use of the server certificate is up to the client, meaning the client may authenticate the server certificate and reject the connection if it is not trusted, or he may accept the certificate without performing any verification.

Whether or not client certificate is performed is depending on server's configuration. When client authentication is selected, a client is requested to send its certificate to the server. The server verifies if the client certificate has been signed by a Certificate Authority (CA) that is trusted for him. An SSL connection is established if the client certificate is validated.

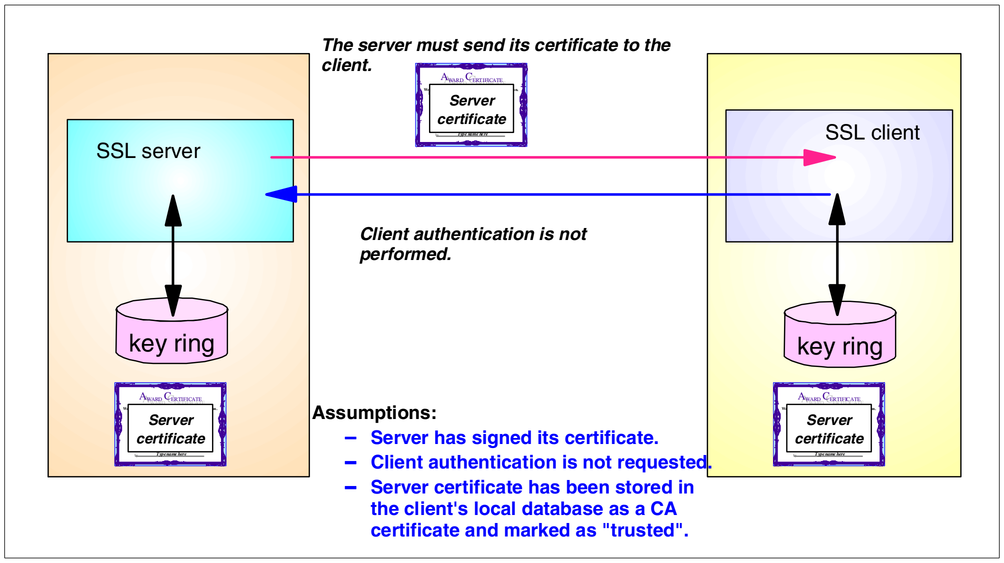
To perform server or client authentication, the client or server has to recognize that a CA that has signed its communication peer's certificate is trusted, respectively. (Of course, each side has to trust the CA that has issued its own digital certificate.) Thus the CA's certificates must be located in the client's and server's local database (or *keyring*) marked as trusted. See Figure 202.



The server must send its certificate to the client.
Server certificate
SSL server
SSL client
Client certificate
The client sends its certificate to the server when requested.
key ring
key ring
CA2
Server certificate
CA1 certificate
Client certificate
CA1
CA2 certificate
Assumptions:
– CA1 has signed the server certificate.
– CA2 has signed the client certificate.
– Each CA certificate has been marked as "trusted".

Figure 202. Certificate management for SSL: CA-signed certificates

If you choose to use self-signed certificate instead of CA-signed, the CA certificate that should be trusted is identical to the server/client certificate itself. Suppose that the server itself has signed its certificate and client authentication is not required, the server certificate must be exported and stored in the client's local database as a trusted CA certificate, because the server certificate is the issuer's certificate for itself.



The server must send its certificate to the client.
Server certificate
SSL server
SSL client
Client authentication is not performed.
key ring
key ring
Server certificate
Server certificate
Assumptions:
– Server has signed its certificate.
– Client authentication is not requested.
– Server certificate has been stored in the client's local database as a CA certificate and marked as "trusted".

Figure 203. Certificate management for SSL: self-signed certificate without client authentication

When the client does not perform server authentication, it is not necessary to store the server certificate in the client's keyring.

D.1.1 Certificate usage for the IKE connection

Digital signatures may be exchanged between IKE communication peers to authenticate and identify each other. The authentication method using digital certificates is usually called RSA signature mode authentication. In this case, the CA that has signed its certificate has to be trusted by its communication peers.

D.2 Certificate management with RACF keyrings

RACF can be used to create, register, store, and administer digital certificates and the private keys associated with the certificates. RACF can also be used to create and manage keyrings of stored digital certificates. Certificates are stored in the RACF database, while private keys may be stored in the ICSF Public Key Data Set (PKDS), encrypted under a 168-bit Triple-DES key.

RACF distinguishes three types of digital certificates:

- Certificate-authority certificates: these certificates are associated with certificate authorities (CAs) and are used to verify signatures in other certificates.
- Site certificates: these certificates are associated with servers or network entities in other locations than the local system.
- User certificates: these certificates are associated with a RACF user ID and are used to authenticate a user's identity.

A user certificate or a certificate that has been connected to a keyring with `USAGE(PERSONAL)` is the only type of certificate whose private key can be used to create signatures. Therefore, all server certificates for local servers need to be user certificates or the need to be connected to an appropriate keyring with `USAGE(PERSONAL)`.

The following step-by-step example is generic in nature. It can be used to create a RACF keyring for the IBM HTTP Server for OS/390, the TN3270 server, or other servers that are SSL enabled.

For the dynamic VPN implementation using the IKE protocol, when RSA Signature mode authentication is chosen, each of the endpoints, IKE servers, must have a certificate signed by a CA to authenticate each other. The RACF `RACDCERT` command is the only certificate management tool supported by the IKE server in Firewall Technologies.

In both the SSL and IKE environment, a server can use a self-signed certificate or a certificate obtained from an external CA. We will show both alternatives here.

D.2.1 Using a CA-signed certificate for an SSL environment

This section presents the steps required to implement the SSL environment for IBM HTTP Server, although a similar procedure can be used for other SSL-enabled application servers. In this scenario, we use a server certificate signed by a public CA.

1. Generate a self-signed certificate

   We will use this certificate as a base for the certificate request we will be creating.

   ```
   RACDCERT  ID(WEBSRV)  GENCERT
   ```

```
       SUBJECTDSN(CN('itso.raleigh.ibm.com')
               O('IBM Corporation')
               OU('ITSO Raleigh Webserver')
               C('US'))
       WITHLABEL('Web Server Certificate')
```

Make sure the common name (CN) is the same as the host or domain name of the server.

2. Create a certificate request for the CA

   The certificate request will be stored in an MVS data set with a name like 'BOCHE.WEBSERV.GENREQ'.

   ```
   RACDCERT  ID(WEBSRV)  GENCERT
          GENREQ(LABEL('Web Server Certificate'))
          DSN('BOCHE.WEBSERV.GENREQ')
   ```

   This certificate request needs to be sent to the Certificate Authority. The format of the request is Base64-encoded text. The data set can be transmitted to a PC with FTP and pasted into the appropriate field in the certificate request. Alternatively, cutting and pasting between a host emulator window and the Web browser can be used.

3. Store the returned certificate into a data set

   The CA usually returns the certificate using e-mail or similar means. The certificate is in Base64-encoded text format. Again, use the same technique as before to copy the certificate into a data set named, for instance, 'BOCHE.WEBSERV.CERT'.

4. Replace the self-signed certificate with the certificate received from and signed by the CA

   ```
   RACDCERT  ID(WEBSRV)
          ADD('BOCHE.WEBSERV.CERT')
          WITHLABEL('Web Server Certificate')
   ```

5. Create a keyring for the server

   This keyring must not already exist for this user. Keyring names becomes names of RACF profiles in the DIGTRING class, and can contain only characters that are allowed in RACF profile names. Although asterisks are allowed in keyring names, a single asterisk is not allowed.

   ```
   RACDCERT ID(WEBSRV) ADDRING(WEBSERVER)
   ```

6. Connect the certificate to the keyring

   Now we can create the connection between the digital certificate and the keyring with the `RACDCERT CONNECT` command and associate it with the HTTP started task user ID.

   ```
   RACDCERT ID(WEBSRV) CONNECT(ID(WEBSRV) LABEL('Web Server Certificate')
            RING(WEBSERVER) DEFAULT USAGE(PERSONAL))
   ```

For the information about how IBM HTTP Server handles the client certificate when client authentication is performed, refer to 5.1.7, “Associating a client certificate with a RACF user ID” on page 121.

When client authentication is required by other application servers, such as TN3270, you might have to store the CA certificate that has signed client certificates into the RACF database. The similar procedure described in D.2.3.2, “Step 2: Receiving CA certificates” on page 326 may be used for this purpose.

D.2.2 Using a self-signed certificate for an SSL environment

When you choose to use a self-signed certificate for the server, follow the steps below:

1. Create a self-signed certificate for the local CA

```
RACDCERT  CERTAUTH  GENCERT
       SUBJECTDSN(O('IBM Corporation')
               OU('ITSO Raleigh Certificate Authority') C('US'))
       WITHLABEL('ITSO  Certificate Authority')
```

2. Generate a self-signed certificate for the server

   This certificate is signed with the CA certificate created in the first step.

```
RACDCERT  ID(WEBSRV)  GENCERT
       SUBJECTDSN(CN('itso.raleigh.ibm.com')
               O('IBM Corporation')
               OU('ITSO Raleigh Webserver')
               C('US'))
       WITHLABEL('Web Server Certificate')
       SIGNWITH(CERTAUTH)
       LABEL('ITSO  Certificate Authority')
```

   Make sure the common name (CN) is the same as the host or domain name of the server.

3. Create a keyring for the server

   To create a keyring and to connect the self-signed certificate with the keyring, continue with step 5 in D.2.1, “Using a CA-signed certificate for an SSL environment” on page 323.

D.2.3 Using a CA-signed certificate for IKE connections

The steps similar to the following need to be done when CA-signed digital certificates are used:

1. Define the firewall IKE keyring, by default, ikekeyring.
2. Receive the CA certificate and store it in the RACF database, if necessary.
3. Obtain the CA-signed certificate and add it to the RACF database.
4. Connect both certificates to the IKE keyring.

D.2.3.1 Step 1: Defining the IKE keyring

Verify that the user ID of FWKERN, by default the same name FWKERN, has not already defined the IKE keyring, by default, ikekeyring:

```
RACDCERT ID(fwkern) LISTRING(*)
```

If it is already created, you will see the following:

```
Digital ring information for user FWKERN:
Ring:
>ikekeyring<
 *** No certificates connected ***
```

If you do not see a message such as this one, you have to add the keyring with the following command:

```
RACDCERT ID(fwkern) ADDRING(ikekeyring)
```

D.2.3.2 Step 2: Receiving CA certificates

The next step is to receive the CA certificate from a known CA that supports *VPN certificates.*

We have used the following Web addresses to request test certificates that can be used temporarily for projects such as our own:

Entrust

```
http://freecerts.entrust.com/vpncerts/index.htm
```

VeriSign

```
https://onsite-test-fe.bbtest.net/bakeoff/
```

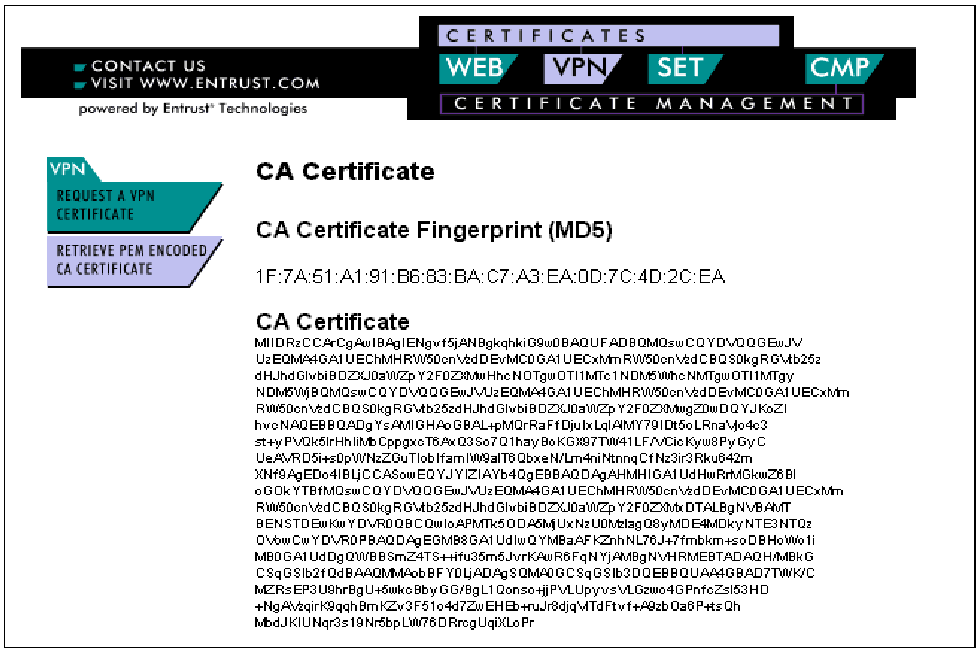
SSH

```
http://isakmp-test.ssh.fi/cgi-bin/nph-cert-test
```

Baltimore Technologies

```
http://www.baltimore.com/vpn/index.html
```

If you connect (for example) to the Entrust site, you will be able to fill in a request form, and eventually you will see a window as shown in Figure 204:


CERTIFICATES
CONTACT US
VISIT WWW.ENTRUST.COM
powered by Entrust® Technologies
WEB
VPN
SET
CMP
CERTIFICATE MANAGEMENT
VPN
REQUEST A VPN CERTIFICATE
RETRIEVE PEM ENCODED CA CERTIFICATE

CA Certificate

CA Certificate Fingerprint (MD5)

1F:7A:51:A1:91:B6:83:BA:C7:A3:EA:0D:7C:4D:2C:EA

CA Certificate

```
MIIDRzCCArCgAwIBAgIENgvf5jANBgkqhkiG9w0BAQUFADBQMQswCQYDVQQGEwJV
UzEQMA4GA1UEChMHRW50cnVzdDEvMC0GA1UECxMmRW50cnVzdCBQS0kgRGVtb25z
dHJhdGlvbiBDZXJ0aWZpY2F0ZXMwHhcNOTgwOTI1MTc1NDM5WhcNMTgwOTI1MTgy
NDM5WjBQMQswCQYDVQQGEwJVUzEQMA4GA1UEChMHRW50cnVzdDEvMC0GA1UECxMm
RW50cnVzdCBQS0kgRGVtb25zdHJhdGlvbiBDZXJ0aWZpY2F0ZXMwgZ0wDQYJKoZI
hvcNAQEBBQADgYsAMIGHAoGBAL+pMQrRaFfDjuIxLqlAlMY79IDt5oLRnaVjo4c3
st+yPVQk5lrHhliMbCppgxcT6AxQ3So7Q1hayBoKGX97TW41LF/VCicKyw8PyGyC
UeAVRD5i+s0pWNzZGuTloblfamlW9alT6QbxeN/Lm4niNtnnqCfNz3ir3Rku642m
XNf9AgEDo4IBLjCCASowEQYJYIZIAYb4QgEBBAQDAgAHMHIGA1UdHwRrMGkwZ6Bl
oGOkYTBfMQswCQYDVQQGEwJVUzEQMA4GA1UEChMHRW50cnVzdDEvMC0GA1UECxMm
RW50cnVzdCBQS0kgRGVtb25zdHJhdGlvbiBDZXJ0aWZpY2F0ZXMxDTALBgNVBAMT
BENSTDEwKwYDVR0QBCQwIoAPMTk5ODA5MjUxNzU0MzlagQ8yMDE4MDkyNTE3NTQz
OVowCwYDVR0PBAQDAgEGMB8GA1UdIwQYMBaAFKZnhNL76J+7fmbkm+soDBHoWo1i
MB0GA1UdDgQWBBSmZ4TS++ifu35m5JvrKAwR6FqNYjAMBgNVHRMEBTADAQH/MBkG
CSqGSIb2fQdBAAQMMAobBFY0LjADAgSQMA0GCSqGSIb3DQEBBQUAA4GBAD7TWK/C
MZRsEP3U9hrBgU+6wkcBbyGG/BgL1Qonso+jjPVLUpyvsVLGzwo4GPnfcZsI53HD
+NgAVzqirK9qqhBmKZv3F51o4d7ZwEHEb+ruJr8djqVTdFtvf+A9zbOa6P+tsQh
MbdJKIUNqr3s19Nr5bpLW76DRrcgUqiXLoPr
```

Figure 204. Requesting a CA certificate

In order to receive this CA certificate, the next step will be to send this encoded text to our OS/390 system.

Be careful with the use of copy and paste; remember that you have to consider the ASCII/EBCDIC translation. To avoid problems, a better way is to FTP the text file into a preallocated data set with the following attributes:

Record format	VB
Record length	4096
Block size	27998

Next, you must add this CA certificate with the RACDCERT command:

```
RACDCERT CERTAUTH ADD('JRIVERA.RECEIVE.CAENT') TRUST
         WITHLABEL('CA Entrust')
```

In this example the preallocated data set is JRIVERA.RECEIVE.CAENT and we assign our new certificate the label "CA Entrust".

To verify that it is correctly defined, use the following command:

```
RACDCERT CERTAUTH LIST
```

You should expect to see results as shown below:

```
Label: CA Entrust
Status: TRUST
Start Date: 1998/09/25 13:54:39
End Date:   2018/09/25 14:24:39
Serial Number:
     >360BDFE6<
Issuer's Name:
     >OU=Entrust PKI Demonstration Certificates.O=Entrust.C=US<
Subject's Name:
     >OU=Entrust PKI Demonstration Certificates.O=Entrust.C=US<
Private Key Type: None
Ring Associations:
  Ring Owner: FWKERN
  Ring:
     >ikekeyring<
```

D.2.3.3 Step 3: Obtain a CA-signed certificate for the IKE server

The next step is the creation of the Key Server certificate. Perform the following steps:

1. Generate a self-signed certificate for the server. This certificate is associated with the user ID that is associated with the IKE server.

   ```
   RACDCERT ID(FWKERN) GENCERT SUBJECTSDN( CN('itso.ral.ibm.com')
            OU('ITSO') O('HOST RA03') C('US'))
            WITHLABEL('HOST RA03')
   ```

2. Create a certificate request to send to our chosen Certificate Authority.

   The certificate request being created is based on the self-signed certificate generated in the step above. Place this certificate into the preallocated data set (JRIVERA.REQHOST.RA03, in our example) with the following attributes:

Record format	VB
Record length	4096
Block size	27998

   The command to put the certificate into the MVS data set is:

   ```
   RACDCERT ID(FWKERN) GENREQ (LABEL('HOST RA03')) DSN('JRIVERA.REQHOST.RA03')
   ```

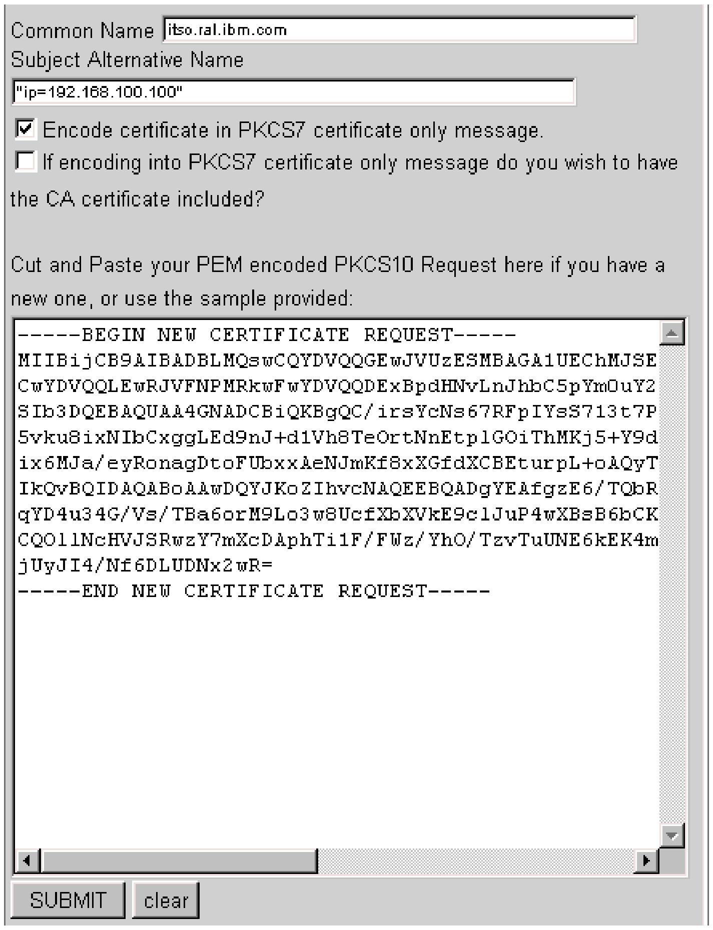
3. Send the certificate request to the Certificate Authority. The certificate request is in Base64-encoded text.

   Typically, the request is sent to the Certificate Authority by using cut and paste to place the certificate request into a Web form that is then sent to the Certificate Authority.

Depending on the CA, you may have to define the contents of particular fields as follows:

- CN=your.domain.name
- Subject Alternate Name="ip=192.168.100.100" (your IP address)

Note: RACF is not involved in this step.



Common Name itso.ral.ibm.com

Subject Alternative Name

"ip=192.168.100.100"

[✓] Encode certificate in PKCS7 certificate only message.

[ ] If encoding into PKCS7 certificate only message do you wish to have the CA certificate included?

Cut and Paste your PEM encoded PKCS10 Request here if you have a new one, or use the sample provided:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBijCB9AIBADBLMQswCQYDVQQGEwJVUzESMBAGA1UEChMJSE
CwYDVQQLEwRJVFNPMRkwFwYDVQQDExBpdHNvLnJhbC5pYm0uY2
SIb3DQEBAQUAA4GNADCBiQKBgQC/irsYcNs67RFpIYsS713t7P
5vku8ixNIbCxggLEd9nJ+d1Vh8TeOrtNnEtplGOiThMKj5+Y9d
ix6MJa/eyRonagDtoFUbxxAeNJmKf8xXGfdXCBEturpL+oAQyT
IkQvBQIDAQABoAAwDQYJKoZIhvcNAQEEBQADgYEAfgzE6/TQbR
qYD4u34G/Vs/TBa6orM9Lo3w8UcfXbXVkE9clJuP4wXBsB6bCK
CQOllNcHVJSRwzY7mXcDAphTi1F/FWz/YhO/TzvTuUNE6kEK4m
jUyJI4/Nf6DLUDNx2wR=
-----END NEW CERTIFICATE REQUEST-----
```

Figure 205. Defining the certificate attributes

In Firewall Technologies configuration, you have the following options to choose from for your AuthID type (see Figure 206 on page 329):

- IPv4 (IP Version 4 address)
- FQDN (fully qualified domain name)
- USER@FQDN (user within fully qualified domain, as in an e-mail address)
- X500_DN (X.500 distinguished name)

Depending on the AuthID type chosen, specific information may be required for the Subject Alternate Name field. For example, when IPv4 is chosen, this field must contain the IP address over which the IKE connection is established.

In the scenarios in Chapter 6, “SecureWay Security Server Firewall Technologies” on page 253 we chose IPv4 or USER@FQDN as our AuthID type.

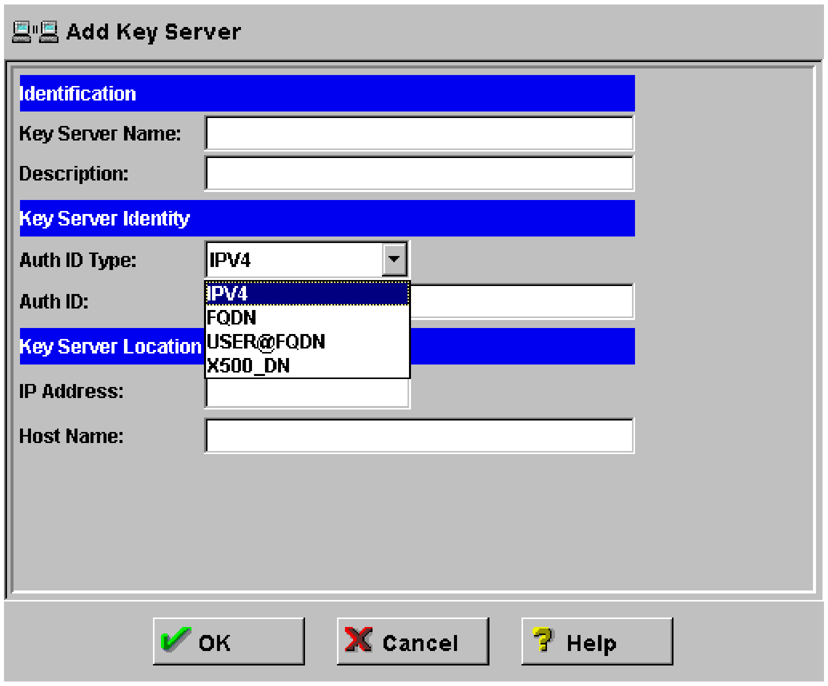


Figure 206. Selecting the Auth ID Type

4. The Certificate Authority validates the certificate. If the certificate is approved by the Certificate Authority, it is signed by the Certificate Authority, and returned to the requestor.

   Note: RACF is not involved in this step.

5. Receive the returned certificate in a data set with the previously described attributes (per "JRIVERA.SIGNED.RA03'). The returned certificate is in Base64-encoded text. The transfer may be done with FTP or a similar technique.

   Note: RACF is not involved in this step.

6. Replace the self-signed certificate with the certificate signed by the Certificate Authority.

   Note that the certificate is replaced only if the user ID that is specified as the ID value on the RACDCERT ADD command is the same user ID that was specified when the certificate was created. If the ID is not the same, then the certificate is added anew.

   ```
   RACDCERT ID(FWKERN) ADD('JRIVERA.SIGNED.RA03') WITHLABEL('HOST RA03')
   ```

D.2.3.4 Step 4: Connect both certificates to the Firewall keyring

Finally, connect both the CA certificate and the IKE server certificate to the default firewall keyring, ikekeyring:

```
RACDCERT ID(FWKERN) CONNECT (CERTAUTH LABEL('CA ENTRUST') RING(ikekeyring))
RACDCERT ID(FWKERN) CONNECT(ID(FWKERN) LABEL('HOST RA03') RING(ikekeyring)
         DEFAULT
```

For the IKE certificate, do not forget to use the value DEFAULT.

In order to check the status of your new certificate, the following command may be used:

```
RACDCERT ID(fwkern) LIST
```

You will see a screen like this:

```
Label: HOST RA03
Status: TRUST
Start Date: 1999/08/26 10:09:55
End Date:   1999/10/26 10:39:55
Serial Number:
     >360CC9A7<
Issuer's Name:
     >OU=Entrust PKI Demonstration Certificates.O=Entrust.C=US<
Subject's Name:
     >CN=itso.ral.ibm.com.OU=Entrust/VPN Connector.OU=No Liability as per h<
     >ttp://freecerts.entrust.com/license.htm.OU=Entrust PKI Demonstration <
     >Certificates.O=Entrust.C=US<
Private Key Type: Non-ICSF
Private Key Size: 1024
Ring Associations:
  Ring Owner: FWKERN
```

D.2.4 Using a self-signed certificate for IKE connections

Sometimes you may believe that your own intranet is not secure, and you need to be able to construct a VPN between any two internal machines in your company.

In that case, you may not wish to pay an external Certificate Authority for a certificate because the whole process is under your control. You may prefer to define your own installation as a CA, and use self-signed certificates. To do this, perform the following steps:

1. Generate a self-signed certificate to represent the local Certificate Authority. This certificate is used as the CA certificate.

```
RACDCERT CERTAUTH GENCERT
       SUBJECTSDN(CN('itso.ral.ibm.com')
               O('IBM ITSO')
               OU('RA03 Self CertAuth')
               C('US'))
       WITHLABEL('SELFSIGNED RA03')
```

2. Create the Server Certificate signed by the already created self-signed Certificate Authority:

```
RACDCERT ID(FWKERN) GENCERT
       SUBJECTSDN(CN('itso.ral.ibm.com')
               O('IBM ITSO') OU('Host RA03') C('US'))
       WITHLABEL('Server-RA03') SIGNWITH(CERTAUTH LABEL('SELFSIGNED RA03'))
```

3. Connect both definitions to your (already created) keyring:

```
RACDCERT ID(FWKERN) CONNECT(CERTAUTH LABEL('SELFSIGNED RA03')
       RING(IKEKEYRING))
RACDCERT ID(FWKERN) CONNECT(ID(FWKERN) LABEL('SERVER-RA03')
       RING(IKEKEYRING) DEFAULT)
```

D.3 Certificate management using the GSKKYMAN utility

Although the use of RACF keyring is preferred for security reasons, it is still possible to create and maintain key databases in the HFS.

For the complete description about the GSKKYMAN utility, please refer to *OS/390 System Secure Sockets Layer Programming Guide and Reference*, SC24-5877.

D.3.1 Using a CA-signed certificate for SSL connections

The following step-by-step example shows how to create a key database in the HFS and how to create certificate requests for external CAs for the IBM HTTP Server for OS/390.

However, a similar procedure may be used for other server applications, including the TN3270 server and the policy agent. You can find further information about particular server applications in *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 1: Configuration and Routing*, SG24-5227 and *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228.

D.3.1.1 Create a new key database

You use the GSKKYMAN utility provided by OS/390 System SSL for this task. First of all, to run the GSKKYMAN utility, you must have access to the OS/390 Cryptographic Services message catalogs and DLLs. For example, if the OS/390 Cryptographic Service DLL library is not part of the LINKLST concatenation, an `export STEPLIB=hlq.SGSKLOAD` command might be needed. For additional information, refer to *OS/390 Cryptographic Services System Secure Sockets Layer Programming Guide and Reference,* SC24-5877.

There are two ways for you to create a server certificate. One is to ask an external Certificate Authority (CA) to create your server's certificate. If you plan to publish your information to the public or your business partners, you should get your certificate from a trusted CA such as VeriSign, Inc. or any other certification authority whose root CA certificate is contained in the key database of the Web browsers used by your clients.

The second way that you create your server's certificate is by yourself. This type of certificate is called a *self-signed* certificate. This might be useful when you use this for testing purposes or you establish the SSL connections only within your intranet. For detailed information on how to create a self-signed certificate, see *Enterprise Web Serving with the Lotus Domino Go Webserver for OS/390*, SG24-2074.

In our test environment, we elected to have our certificate issued by VeriSign. In the following discussion, we will show how to create our server certificate.

Figure 207 shows the flow of obtaining a server certificate and how to make IBM HTTP Server enable SSL server authentication:

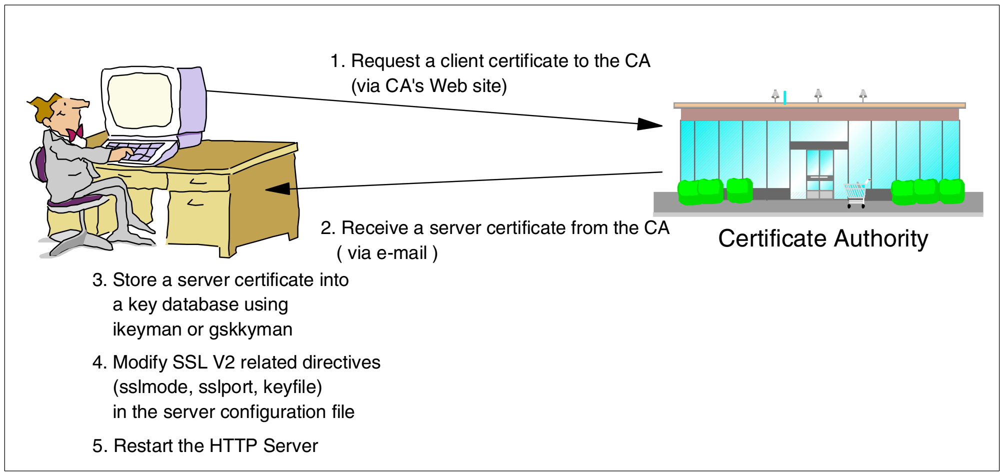

1. Request a client certificate to the CA (via CA's Web site)
2. Receive a server certificate from the CA (via e-mail)
Certificate Authority
3. Store a server certificate into a key database using ikeyman or gskkyman
4. Modify SSL V2 related directives (sslmode, sslport, keyfile) in the server configuration file
5. Restart the HTTP Server

Figure 207. Flow to implement SSL server authentication

First of all, you need to create a key database file (which must be specified in the httpd.conf file later) by using the GSSKYMAN utility.

```
TAKADA @ RA03:/u/takada/sslkey>gskkyman 1

            IBM Key Management Utility

Choose one of the following options to proceed.

   1  - Create new key database
   2  - Open key database
   3  - Change database password

   0  - Exit program

Enter your option number: 1 2
Enter key database name or press ENTER for "key.kdb": server.kdb 3
Enter password for the key database.......>****** 4
Enter password again for verification.....>******
Should the password expire? (1 = yes, 0 = no) [1]: 1
Enter password expiration time (number of days) or press ENTER for 60 days:

The database has been successfully created, do you want to continue to work with
 the database now? (1 = yes, 0 = no) [1]: 1
```

Figure 208. Create a key database using the GSKKYMAN utility

1 Run the GSKKYMAN utility from the OS/390 UNIX shell environment.

2 You can create the new key database by selecting option **1 - Create new key database**.

3 Enter the key database name. We defined a new key database as server.kdb.

4 Specify a password for this key database. In the future you will have to enter this password to open this key database.

7.3.3.1 Create a public-private key pair and certificate request

Figure 209 shows the menu screen of the GSKKYMAN utility. To create a public-private key pair and a certificate request, you select **3 - Create new key pair and certificate request** on this screen.

```
Key database menu

Current key database is /u/takada/sslkey/server.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 3 1
Enter certificate request file name or press ENTER for "certreq.arm": server.arm 2
Enter a label for this key................> ITSO Raleigh Webserver Cert 3
Select desired key size from the following options (512):
    1:    512
    2:    1024
Enter the number corresponding to the key size you want: 1 4
Enter certificate subject name fields in the following. 5
    Common Name (required)................> mvs03c.itso.ral.ibm.com
    Organization (required)...............> IBM Corp.
    Organization Unit (optional)..........> ITSO Raleigh
    City/Locality (optional)..............> Research Triangle Park
    State/Province (optional).............> North Carolina
    Country Name (required 2 characters)..> US

Please wait while key pair is created...

Your request has completed successfully, exit gskkyman? (1 = yes, 0 = no) [0]: 0
```

Figure 209. Create a new key pair and certificate request

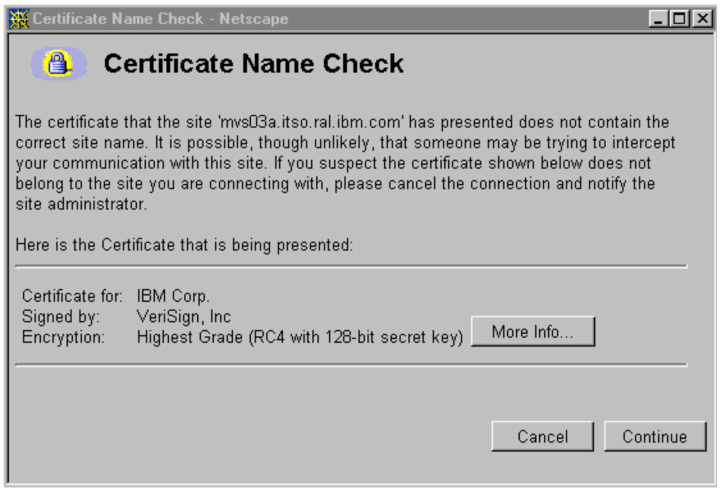
1 Select option **3 - Create new key pair and certificate request** to create a new key pair and certificate request. If you want to create a self-signed certificate, select option **5 - Create a self-signed certificate.**

2 Specify a file name for the certificate request. Later you have to send the contents of this file to the CA you selected.

3 Enter a label related to this key and certificate.

4 Select the key size you desire. However, the key size depends on the location. In our test environment (ITSO Raleigh), we installed the strong crypto version of the GSKKYMAN utility. In almost all cases, you would want to install the strong crypto version and use a key size of 1024 bits.

5 Enter the certificate subject name fields. Common Name should be your server's host name. If you specify another name, a user will receive the window shown in Figure 210 when accessing this server:


Certificate Name Check - Netscape

Certificate Name Check

The certificate that the site 'mvs03a.itso.ral.ibm.com' 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:	IBM Corp.
Signed by:	VeriSign, Inc
Encryption:	Highest Grade (RC4 with 128-bit secret key)

Figure 210. Netscape Navigator's window checking certificate name

The only way a Web browser can check the server's identity is to compare the host name in the URL with the host name in the Common Name attribute in the certificate. If they don't match, Netscape Navigator will display the warning window shown in Figure 210. Whether Internet Explorer (IE) will display a warning window or simply terminate the connection depends on the release level of IE and the security level chosen.

```
Key database menu

Current key database is /u/takada/sslkey/server.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 11 6

The encrypted password has been stored in file /u/takada/sslkey/server.sth

Your request has completed successfully, exit gskkyman? (1 = yes, 0 = no) [0]: 1
```

Figure 211. Create a stash file containing an encrypted password for a key database

6 Create an encrypted key database password. This file is used by the IBM HTTP Server when the server tries to open this key database file. This password file is created in the same directory as the key database file.

Note

For most server applications, you need to store the encrypted key database password in a stash file. To create a stash file, you specify option 11 in the GSKKYMAN menu screen. The extension of a stash file is .sth. If you do not create a stash file, some servers may start but may not be able to use SSL. For some applications, the stash file name has to be known to servers. Check the configuration information for each server.

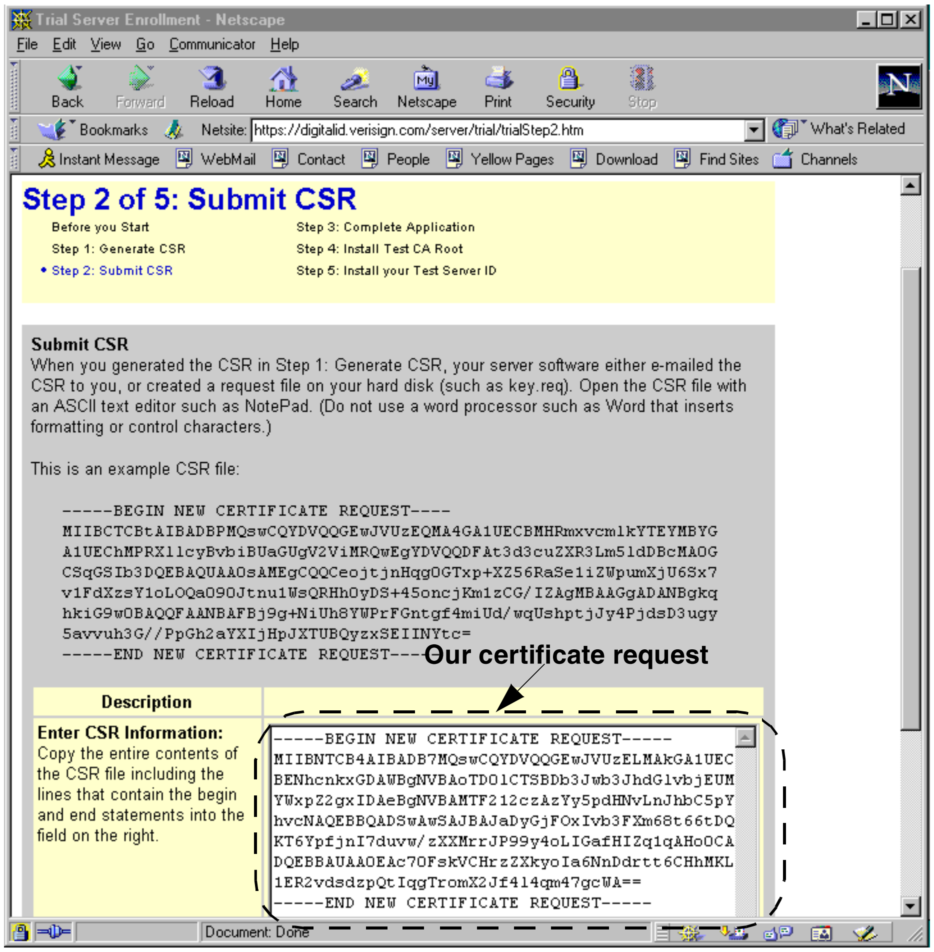
After the work shown in Figure 209 on page 333 is done, you have three files in addition to the key database:

- A certificate request file (*.arm)
- A stash file (*.sth)
- A key pair file (*.rdb)

Figure 212 shows the contents of the certificate request file. You send this request to the Certificate Authority to be signed. We sent this certificate request to VeriSign; as shown in Figure 213 on page 336, you can copy and paste the contents of the request file into the VeriSign form.

```
TAKADA @ RA03:/u/takada/sslkey>ls -l
total 192
rw-r--r--   1 TAKADA   OMVSGRP      513 Aug  6 18:56 server.arm
-rw-r--r--   1 TAKADA   OMVSGRP    65080 Aug  6 18:54 server.kdb
-rw-r--r--   1 TAKADA   OMVSGRP     5080 Aug  6 18:56 server.rdb
-rw-------   1 TAKADA   OMVSGRP      129 Aug  6 18:56 server.sth
TAKADA @ RA03:/u/takada/sslkey>cat server.arm
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBNTCB4AIBADB7MQswCQYDVQQGEwJVUzELMAkGA1UECBMCTkMxDTALBgNVBAcT
BENhcnkxGDAWBgNVBAoTD0lCTSBDb3Jwb3JhdGlvbjEUMBIGA1UECxMLSVRTTyBS
YWxpZ2gxIDAeBgNVBAMTF212czAzYy5pdHNvLnJhbC5pYm0uY29tMFwwDQYJKoZI
hvcNAQEBBQADSwAwSAJBAJaDyGjFOxIvb3FXm68t66tDQ+dn9B/zLthCS7dc7nor
KT6YpfjnI7duvw/zXXMrrJP99y4oLIGafHIZq1qAHo0CAwEAAaAAMA0GCSqGSIb3
DQEBBAUAA0EAc70FskVCHrzZXkyoIa6NnDdrtt6CHhMKLJKtIitStFPXZVIMQxPK
1ER2vdsdzpQtIqgTromX2Jf4l4qm47gcWA==
-----END NEW CERTIFICATE REQUEST----
```

Figure 212. The content of the certificate request file


Trial Server Enrollment - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Netsite: https://digitalid.verisign.com/server/trial/trialStep2.htm What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
Document: Done
Our certificate request

Step 2 of 5: Submit CSR

Before you Start	Step 3: Complete Application
Step 1: Generate CSR	Step 4: Install Test CA Root
• Step 2: Submit CSR	Step 5: Install your Test Server ID

Submit CSR
When you generated the CSR in Step 1: Generate CSR, your server software either e-mailed the CSR to you, or created a request file on your hard disk (such as key.req). Open the CSR file with an ASCII text editor such as NotePad. (Do not use a word processor such as Word that inserts formatting or control characters.)

This is an example CSR file:

```
-----BEGIN NEW CERTIFICATE REQUEST----
MIIBCTCBtAIBADBPMQswCQYDVQQGEwJVUzEQMA4GA1UECBMHRmxvcmlkYTEYMBYG
A1UEChMPRXllcyBvbiBUaGUgV2ViMRQwEgYDVQQDFAt3d3cuZXR3Lm5ldDBcMA0G
CSqGSIb3DQEBAQUAA0sAMEgCQQCeojtjnHqg0GTxp+XZ56RaSe1iZWpumXjU6Sx7
v1FdXzsY1oLOQa09OJtnu1WsQRHh0yDS+45oncjKm1zCG/IZAgMBAAGgADANBgkq
hkiG9w0BAQQFAANBAFBj9g+NiUh8YWPrFGntgf4miUd/wqUshptjJy4PjdsD3ugy
5avvuh3G//PpGh2aYXIjHpJXTUBQyzxSEIINYtc=
-----END NEW CERTIFICATE REQUEST----
```

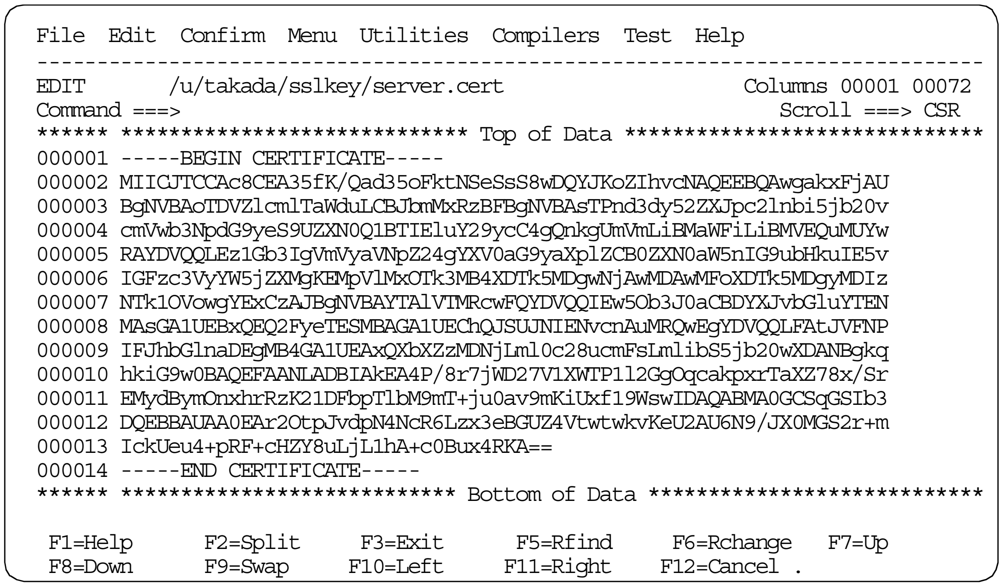
Description	
Enter CSR Information: Copy the entire contents of the CSR file including the lines that contain the begin and end statements into the field on the right.	-----BEGIN NEW CERTIFICATE REQUEST----- MIIBNTCB4AIBADB7MQswCQYDVQQGEwJVUzELMAkGA1UEC BENhcnkxGDAWBgNVBAoTDO1CTSBDb3Jwb3JhdGlvbjEUM YWxpZ2gxIDAeBgNVBAMTF212czAzYy5pdHNvLnJhbC5pY hvcNAQEBBQADSwAwSAJBAJaDyGjFOxIvb3FXm68t66tDQ KT6YpfjnI7duvw/zXXMrrJP99y4oLIGafHIZq1qAHoOCA DQEBBAUAA0EAc7OFskVCHrzZXkyoIa6NnDdrtt6CHhMKL 1ER2vdsdzpQtIqgTromX2Jf414qm47gcWA== -----END NEW CERTIFICATE REQUEST-----

Figure 213. Certificate request submit form at VeriSign Web site

After you send a certificate request to an external CA, it can take several days before the request is processed and the certificate is returned. We used the trial Secure Server ID from VeriSign to test the IBM HTTP Server for OS/390 SSL function in the ITSO Raleigh test environment. This provides a temporary certificate that is valid for two weeks from the date of issuance. Because the certificate is temporary, it does not require the extensive checking that a real certificate would, so we received it almost immediately after submitting the certificate request.

While you are waiting for the CA to process your certificate request, it is a good idea to exploit a trial certificate to test SSL sessions. Alternatively (or in addition), you can use the GSKKYMAN utility to create a self-signed certificate to enable SSL sessions between clients and the server. For detailed information regarding how to make a self-signed certificate, see *IBM HTTP Server for OS/390 Planning, Installing, and Using*, SC31-8690, or *Enterprise Web Serving with the Lotus Domino Go Webserver for OS/390*, SG24-2074.

After receiving a certificate from the CA via e-mail, copy and paste it to an HFS file. In Figure 214, we created a file server.cert and put our certificate into this file.



```
 File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
 -------------------------------------------------------------------------------
 EDIT       /u/takada/sslkey/server.cert                    Columns 00001 00072
 Command ===>                                                  Scroll ===> CSR
 ****** ***************************** Top of Data *****************************
 000001 -----BEGIN CERTIFICATE-----
 000002 MIICJTCCAc8CEA35fK/Qad35oFktNSeSsS8wDQYJKoZIhvcNAQEEBQAwgakxFjAU
 000003 BgNVBAoTDVZlcmlTaWduLCBJbmMxRzBFBgNVBAsTPnd3dy52ZXJpc2lnbi5jb20v
 000004 cmVwb3NpdG9yeS9UZXN0Q1BTIEluY29ycC4gQnkgUmVmLiBMaWFiLiBMVEQuMUYw
 000005 RAYDVQQLEz1Gb3IgVmVyaVNpZ24gYXV0aG9yaXplZCB0ZXN0aW5nIG9ubHkuIE5v
 000006 IGFzc3VyYW5jZXMgKEMpVlMxOTk3MB4XDTk5MDgwNjAwMDAwMFoXDTk5MDgyMDIz
 000007 NTk1OVowgYExCzAJBgNVBAYTAlVTMRcwFQYDVQQIEw5Ob3J0aCBDYXJvbGluYTEN
 000008 MAsGA1UEBxQEQ2FyeTESMBAGA1UEChQJSUJNIENvcnAuMRQwEgYDVQQLFAtJVFNP
 000009 IFJhbGlnaDEgMB4GA1UEAxQXbXZzMDNjLml0c28ucmFsLmlibS5jb20wXDANBgkq
 000010 hkiG9w0BAQEFAANLADBIAkEA4P/8r7jWD27V1XWTP1l2GgOqcakpxrTaXZ78x/Sr
 000011 EMydBymOnxhrRzK21DFbpTlbM9mT+ju0av9mKiUxf19WswIDAQABMA0GCSqGSIb3
 000012 DQEBBAUAA0EAr2OtpJvdpN4NcR6Lzx3eBGUZ4VtwtwkvKeU2AU6N9/JX0MGS2r+m
 000013 IckUeu4+pRF+cHZY8uLjL1hA+c0Bux4RKA==
 000014 -----END CERTIFICATE-----
 ****** **************************** Bottom of Data ****************************

  F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
  F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel .
```

Figure 214. The content of the server certificate issued by the trusted CA

Because the GSKKYMAN utility only accepts HFS files, you have to create an HFS file for your certificate.

4. **Receive your CA-signed server certificate into the key database**

   Please see Figure 215 for details on how to save your certificate into your key database.

```
 IBM Key Management Utility

Choose one of the following options to proceed.

   1  - Create new key database
   2  - Open key database
   3  - Change database password

   0  - Exit program

Enter your option number: 2 1
Enter key database name or press ENTER for "key.kdb": server.kdb
Enter password for the key database.......>******

            Key database menu

Current key database is /u/takada/sslkey/server.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 4 2
Enter certificate file name or press ENTER for "cert.arm": server.cert 3
Do you want to set the key as the default in your key database? (1 = yes, 0 = no)
1 4
Please wait while certificate is received.......
```

Figure 215. Store the server certificate into the key database

1 Open the key database created in Figure 208 on page 332.

2 Select option **4 - Receive a certificate issued for your request** to store your server certificate file.

3 Specify your server certificate file created in Figure 214 on page 337.

4 You have to select 1. If you set another key as default, server authentication will fail.

D.3.2 Using self-signed certificate for SSL connections

The following step-by-step example shows how to create a self-signed certificate to be used for SSL-enabled applications in OS/390.

Before you run the `gskkyman` command, you might need to make this file known to your OMVS environment. Issue the following command prior to issuing `gskkyman`:

```
export STEPLIB=hlq.SGSKLOAD
```

```
KAKKY @ RA03:/u/kakky/security>export STEPLIB=GSK.SGSKLOAD
KAKKY @ RA03:/u/kakky/security>gskkyman


            IBM Key Management Utility

Choose one of the following options to proceed.

   1  - Create new key database
   2  - Open key database
   3  - Change database password

   0  - Exit program

Enter your option number: 1
Enter key database name or press ENTER for "key.kdb": sslkey.kdb       1
Enter password for the key database.......>
Enter password again for verification.....>
Should the password expire? (1 = yes, 0 = no) [1]: 0

The database has been successfully created, do you want to continue to work wit
 the database now? (1 = yes, 0 = no) [1]: 1
```

Figure 216. Creating a key database for the Telnet server

1 The name of a newly created key database. This name must be defined in configuration files for SSL-enabled applications.

```
            Key database menu

Current key database is /u/kakky/security/sslkey.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 5
Enter version number of the certificate to be created (1, 2, or 3) [3]:  3   1
Enter a label for this key................> r2631 Telnet server
Select desired key size from the following options (512):
    1:    512
    2:    1024
Enter the number corresponding to the key size you want: 1
Enter certificate subject name fields in the following.
    Common Name (required)................> mvs03a.itso.ral.ibm.com
    Organization (required)...............> IBM
    Organization Unit (optional)..........> ITSO Raleigh
    City/Locality (optional)..............> Cary
    State/Province (optional).............> NC
    Country Name (required 2 characters)..> US
Enter number of valid days for the certificate [365]:
Do you want to set the key as the default in your key database? (1 = yes, 0 = no
) [1]: 1          2
Do you want to save the certificate to a file? (1 = yes, 0 = no) [1]: 1      3
Should the certificate binary data or Base64 encoded ASCII data be saved? (1 = A
SCII, 2 = binary) [1]: 2        4
Enter certificate file name or press ENTER for "cert.crt": r2631Telnet.crt

Please wait while self-signed certificate is created...

Your request has completed successfully, exit gskkyman? (1 = yes, 0 = no) [0]: 0
```

Figure 217. Creating a key pair and a self-signed certificate for the Telnet server

1 Choose Version 3. The version number refers to the X.509 standard version number.

2 You have to mark this key as default in the key database so that the Telnet server will select this certificate associated with this key in order to send to the client.

3 Save this self-signed certificate in binary format 4. In most implementations, the saved server certificate must be sent to the client and stored there as a trusted CA certificate. In this scenario, since the certificate is extracted in binary format, it should be transferred in FTP binary mode.

After creating a self-signed certificate, you may have to create a stash file, which contains the encrypted password for the key database. Most applications, including the TN3270E server, use this file to open the key database specified in their configuration and extract the server certificate. Thus, if you have not created a stash file or if the file name has not been known to a server, the SSL connection establishment will fail.

```
             Key database menu

Current key database is /u/kakky/security/sslkey.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 11

The encrypted password has been stored in file /u/kakky/security/sslkey.sth

Your request has completed successfully, exit gskkyman? (1 = yes, 0 = no) [0]:  1

KAKKY @ RA03:/u/kakky/security>ls -laF
total 176
drwxr-xr-x   2 DB2USER  SYSPROG     8192 Apr  5 12:45 ./              1
drwx------  13 KAKKY    SYSPROG     8192 Apr  5 12:36 ../
-rw-r--r--   1 OMVSKERN SYSPROG      474 Apr  5 12:42 r2631Telnet.crt
-rw-r--r--   1 OMVSKERN SYSPROG    65080 Apr  5 12:42 sslkey.kdb
-rw-------   1 OMVSKERN SYSPROG      129 Apr  5 12:45 sslkey.sth
```

Figure 218. Storing the key database password in an encrypted file

1 After this we should have these three files in the directory.

D.3.3 Other operations with a key database

Using the GSKKYMAN utility, you can do other operations as well.

D.3.3.1 Import a CA certificate into the key database

After opening a key database, select **6 - Store a CA certificate**, then select the name of the file name that contains the certificate to be added.

Note that if your communicating peer is using a self-signed certificate, the certificate should be added as a trusted CA certificate.

```
           Key database menu

Current key database is /u/kakky/security/sslkey.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 6
Enter certificate file name or press ENTER for "cert.arm": ldapsrv.crt
Enter a label for this key................> LDAP server on MVS03

Please wait while certificate is stored...

Your request has completed successfully, exit gskkyman? (1 = yes, 0 = no) [0]: 0
```

Figure 219. Adding a self-signed certificate as a trusted CA certificate

D.3.3.2 Manage a certificates in the key database

Digital certificates in a key database can be exported with or without a private key. You can also browse contents of a certificate.

```
            Key database menu

Current key database is /u/kakky/security/sslkey.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 1

            Key and certificate list

Key database name is /u/kakky/security/sslkey.kdb

Please choose one of the following keys to work with.

    1  - LDAP server on MVS03
    2  - r2631 Telnet server
    3  - Integrion Certification Authority Root
    4  - IBM World Registry Certification Authority
    5  - Thawte Personal Premium CA
    6  - Thawte Personal Freemail CA
    7  - Thawte Personal Basic CA
    8  - Thawte Premium Server CA
    9  - Thawte Server CA

Enter a key number or press ENTER for more labels: 1
```

Figure 220. Listing certificates in a key database

By selecting **1 - List/Manage keys and certificates** from the main menu, all keys/certificates in the key database will be displayed.

If you select one of them, you can do several operations against the chosen key/certificate.

To see the key information, choose **1 - Show key information** as shown in Figure 221.

```
                   Key Menu

Currently selected key: LDAP server on MVS03

Choose one of the following options to proceed.

    1  - Show key information
    2  - Set the selected key as default
    3  - View certificate of the key
    4  - Remove trust root status
    5  - Copy the certificate of this key to a file
    6  - Delete the key
    7  - Export the key to another database

    0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 1


     Basic information of the currently selected key

                     Unique ID:    14
                         Label:    LDAP server on MVS03
         Chosen as default key:    false
                      Key size:    512
               Set as trusted:     true
         Private key existence:    false
  User defined field existence:    false


     Certificate information for the selected key

                    Version:    3
               Serial number:    a8128046d38d3792
                 Issuer name:
                                 R2615 LDAP Server
                                 ITSO Raleigh
                                 IBM
                                 Cary, NC
                                 US
                Subject name:
                                 R2615 LDAP Server
                                 ITSO Raleigh
                                 IBM
                                 Cary, NC
                                 US
              Effective date:    07/11/00
             Expiration date:    07/12/01
    Signature algorithm OID:    md5WithRSAEncryption
            Issuer unique ID:    false
           Subject unique ID:    false
        Number of extensions:    0

Press ENTER to return to the previous menu
```

Figure 221. Display the key information

On the menu in Figure 221, option **5 - Copy the certificate of this key to a file** allows a certificate to be exported into an HFS file without a private key.

In the meantime, if you want a certificate to be exported with its private key, select **9 - Export keys** from the main menu, then choose the format and the name of the file in which the certificate is saved.

```
            Key database menu

Current key database is /u/kakky/security/sslkey.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 9


            Export Key Menu

Current key database is /u/kakky/security/sslkey.kdb

    1  - Export keys to another key database
    2  - Export keys to a PKCS12 file

    0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 2


            Key and certificate lists of the export key database

Key database name is /u/kakky/security/sslkey.kdb

Please choose one of the following keys to work with.

    1  - LDAP server on MVS03
    2  - r2631 Telnet server
    3  - Integrion Certification Authority Root
    4  - IBM World Registry Certification Authority
    5  - Thawte Personal Premium CA
    6  - Thawte Personal Freemail CA
    7  - Thawte Personal Basic CA
    8  - Thawte Premium Server CA
    9  - Thawte Server CA

Enter the numbers of the keys that you want to export (for example, 2,5)
 or press ENTER for more labels: 2
Enter output PKCS12 file name or press ENTER for "key.p12": TenletCert.p12
Enter a password to protect the output PKCS12 file:
Enter password again for verification.....>

Your request has completed successfully, exit gskkyman? (1 = yes, 0 = no) [0]:
```

Figure 222. Export a certificate in PKCS#12 format (with a private key)

Similarly, to import a certificate with a private key, you can use option **8 - Import keys**.

Appendix E. Firewall Technologies configuration for VPN support

To implement VPN configurations in OS/390, you have to implement the firewall services that are part of OS/390 (not CS for OS/390). In addition, the IKE function in OS/390 is implemented by the firewall services in conjunction with CS for OS/390, Open Cryptographic Services Facility and the Security Server (RACF).

E.1 z/OS Communications Server IP services customization

To configure CS for OS/390 IP to support IKE functions you need to perform the following steps. If you need more information, see *OS/390 V2R10.0 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835. At ITSO Raleigh, we used the TCPIPB stack in system RA03 to implement the firewall function.

E.1.1 Configure the TCP/IP Profile

Add the device statements that you will use to connect OS/390 to the network. Refer to *IBM Communications Server for OS/390 IP Configuration*, GC31-8513, for more information. The example below is for a 2216 in LCS mode, or an Open Systems Adapter, connected to a token-ring.

```
DEVICE TR1B LCS      2020 AUTORESTART
LINK  TR1B  IBMTR    0    TR1B
.........
HOME 9.24.104.28  TR1B
.....
START  TR1B
```

If you want to start FWKERN (the firewall program) automatically, insert the following AUTOLOG statement:

```
AUTOLOG
   FWKERN           ; OS/390 Firewall
ENDAUTOLOG
```

Note: If you are running a system that will be connected to the Internet or to another nonsecure network you should remove any AUTOLOG statements for the standard TCP/IP servers. They should start only after FWKERN.

Add the following PORT statements:

```
 500 UDD OMVS                   ; OS/390 Firewall ISAKMP Server
 514 UDP OMVS                   ; OS/390 Firewall Syslogd Server
1014 TCP OMVS                   ; OS/390 Firewall Configuration Client
```

Add the following definitions in the IPCONFIG block statement:

```
IPCONFIG
     FIREWALL
     DATAGRAMFWD
ENDIPCONFIG
```

The FIREWALL keyword cannot be dynamically activated using the `VARY OBEY` console command. To activate the firewall function, you have to restart the TCP/IP stack.

E.1.2 Configure /etc/services

Create the /etc/services file if it does not exist, then add the definitions for the SYSLOG server and ISAKMP server as follows:

```
syslog   514/udp
isakmp   500/udp
```

E.1.3 Checking the TCP/IP configuration

During the TCP/IP stack initialization, you can check that the following messages appear in the system log:

```
EZZ0641I IP FORWARDING NOFWDMULTIPATH SUPPORT IS ENABLED
EZZ0349I FIREWALL SUPPORT IS ENABLED
```

You can check the stack configuration as below:

```
Display TCPIP,TCPIPB,N,CONFIG 1
EZZ2500I NETSTAT CS V2R8 TCPIPB
TCP CONFIGURATION TABLE:
DEFAULTRCVBUFSIZE:  00065536  DEFAULTSNDBUFSIZE: 00065536
DEFLTMAXRCVBUFSIZE: 00262144
MAXRETRANSMITTIME:  120.000   MINRETRANSMITTIME: 0.500
ROUNDTRIPGAIN:      0.125     VARIANCEGAIN:      0.250
VARIANCEMULTIPLIER: 2.000     MAXSEGLIFETIME:    60.000
DEFAULTKEEPALIVE:   0.120     LOGPROTOERR:       00
TCPFLAGS:           10
UDP CONFIGURATION TABLE:
DEFAULTRCVBUFSIZE: 00016384  DEFAULTSNDBUFSIZE: 00016384
CHECKSUM:          00000001  LOGPROTOERR:       01
UDPFLAGS:          23
IP CONFIGURATION TABLE:
FORWARDING: YES 2   TIMETOLIVE: 00060  RSMTIMEOUT:  00015
FIREWALL:   00001 3 ARPTIMEOUT: 01200  MAXRSMSIZE:  65535
```

Figure 223. Report from NETSTAT CONFIG command

1 This (`netstat`) command shows configuration information about the TCP/IP stack.

2 FORWARDING: YES means that TCP/IP will transfer data between networks. NO means that this stack will not transfer data between networks.

3 FIREWALL: 00001 means that the firewall function is active in this stack. 00000 means that this stack is not activated.

To check if the devices are correctly configured and ready issue the `NETSTAT DEvlinks` command.

```
Display TCPIP,TCPIPB,N,DEvlinks 1
EZZ2500I NETSTAT CS V2R8 TCPIPB 757
DEVNAME: LOOPBACK         DEVTYPE: LOOPBACK  DEVNUM: 0000
  LNKNAME: LOOPBACK         LNKTYPE: LOOPBACK    STATUS: READY
    NETNUM: 0   QUESIZE: 0   BYTEIN: 0000308050   BYTEOUT: 0000308050
  BSD ROUTING PARAMETERS:
    MTU SIZE: 00000            METRIC: 00
    DESTADDR: 0.0.0.0          SUBNETMASK: 0.0.0.0
  MULTICAST SPECIFIC:
    MULTICAST CAPABILITY: NO
DEVNAME: TR1B             DEVTYPE: LCS       DEVNUM: 2020
  LNKNAME: TR1B             LNKTYPE: TR          STATUS: READY 2
    NETNUM: 0   QUESIZE: 0   BYTEIN: 0001000461   BYTEOUT: 0001722471
    ARPMACADDRESS: NON-CANONICAL   SRBRIDGINGCAPABILITY: YES
    BROADCASTCAPABILITY: YES       BROADCASTTYPE: ALL RINGS
  BSD ROUTING PARAMETERS:
    MTU SIZE: 00000            METRIC: 00
    DESTADDR: 0.0.0.0          SUBNETMASK: 255.255.255.0
  MULTICAST SPECIFIC:
    MULTICAST CAPABILITY: YES
```

Figure 224. Report from NETSTAT DEvlinks command

1 This command displays information about devices and links defined in the TCP/IP stack.

2 STATUS: READY means that this device is ready and operational.

To check if the PORT statements are correctly configured issue the `NETSTAT PORTList` command.

```
Display TCPIP,TCPIPB,N,PORTList 1
EZZ2500I NETSTAT CS V2R8 TCPIPB 779
PORT# PROT USER     FLAGS RANGE
00007 TCP  MISCSERV A
00009 TCP  MISCSERV A
00019 TCP  MISCSERV A
00020 TCP  OMVS
00021 TCP  FTPD1    A
00021 TCP  FTPDB1   A
00025 TCP  SMTP     A
00053 TCP  OMVS     A
00080 TCP  OMVS     A
00111 TCP  OMVS     A
00443 TCP  OMVS     A
00500 UDP OMVS    A    2
00514 UDP OMVS    A    2
00515 TCP  T03ALPD  A
00750 TCP  MVSKERB  A
00751 TCP  ADM@SRV  A
00760 TCP  IOASNMP  A
01014 TCP OMVS    A     2
```

Figure 225. Report from NETSTAT PORTList command

1 This command shows information about port reservations in the TCP/IP stack.

2 UDP ports 500 and 514, and TCP port 1014 are reserved for a UNIX System Services application, which is FWKERN in this case.

E.2 UNIX System Services customization

There are some parameters in the BPXPRMxx member in SYS1.PARMLIB that must be checked and changed if necessary. Check the following parameters:

- MAXPROCSYS: The firewall requires 11 processes to start its servers. The default is 200.
- MAXPROCUSER: The firewall requires 11 processes to start its servers. The default is 25.
- MAXFILEPROC: The firewall requires at least 25 open file descriptors for its servers, two for each concurrent connection to the SOCKS server and four for each concurrent connection to the FTP proxy server. The default is 64.
- MAXTHREADTASKS: The firewall requires approximately 10 threads for firewall servers, one thread for each concurrent connection to the SOCKS server, and one thread for each concurrent connection to the proxy FTP server. The default is 50.
- MAXTHREAD: The SOCKS and proxy FTP servers require one thread for each concurrent connection. The default is 200.
- MAXSOCKETS: The firewall requires approximately 25 sockets for firewall servers, two for each concurrent connection to the SOCKS server, and four for each concurrent connection to the FTP proxy server. IKE requires one additional socket per interface defined to a firewall stack. The default is 64.

Check if you have AF_UNIX configured. If not, add the following statements to the BPXPRMxx file:

```
NETWORK DOMAINNAME(AF_UNIX)
        DOMAINNUMBER(1)
        MAXSOCKETS(100)
        TYPE(IBMUDS)
```

For more information about the BPXPRMxx configuration, please see *OS/390 UNIX System Services Planning*, SC28-1890 and *OS/390 Initialization and Tuning Reference*, SC28-1752.

E.3 Customizing Firewall Technologies and cryptographic services

The following procedures assume that you are performing them from a RACF administration ID. Table 30 shows a brief description of the RACF profiles we are using here.

Table 30. RACF profile description

Class	Profile	Description
FACILITY	BPX.SMF	Checks if the caller attempting to cut an SMF record is allowed to write an SMF record or test if an SMF type or subtype is being recorded.
FACILITY	BPX.DAEMON	Restricts access to the following services: seteuid, setuid, setruid, setreuid, and spawn. The caller of this service must be a superuser.

Class	Profile	Description
FACILITY	BPX.SERVER	Restricts the use of the pthread_security_np service. A user with read or write access to the BPX.SERVER FACILITY class profile can use this service. It creates or deletes the security environment for the caller's thread. This profile is also used to restrict the use of the BPX1ACK service, which determines access authority to an OS/390 resource.
FACILITY	BPX.SUPERUSER	Users with access to the BPX.SUPERUSER FACILITY class profile can switch to superuser authority (effective UID of 0).
FACILITY	BPX.FILEATTR.APF	Controls that users are allowed to set the APF authorized attribute in an HFS file. This authority allows the user to create a program that will run APF authorized. This is similar to the authority of allowing a programmer to update SYS1.LINKLIB or SYS1.LPALIB.
FACILITY	BPX.FILEATTR.PROGCTL	Controls that users are allowed to set the program-controlled attribute in an HFS file. Programs marked with this attribute can execute in server address spaces that run with a high level of authority.
FACILITY	ICA.CFGSRV	Permits the access of the firewall configuration GUI.
FACILITY	IRR.DIGTCERT.ADD	Permission to add a certificate.
FACILITY	IRR.DIGTCERT.ADDRING	Permission to create a keyring.
FACILITY	IRR.DIGTCERT.CONNECT	Permission to connect to a keyring.
FACILITY	IRR.DIGTCERT.DELETE	Permission to delete a certificate.
FACILITY	IRR.DIGTCERT.DELRING	Permission to delete a keyring.
FACILITY	IRR.DIGTCERT.GENCERT	Permission to generate a certificate.
FACILITY	IRR.DIGTCERT.GENREQ	Permission to generate a certificate request.
FACILITY	IRR.DIGTCERT.LIST	Permission to list a certificate.
FACILITY	IRR.DIGTCERT.LISTRING	Permission to list a keyring.
FACILITY	CDS.CSSM.CRYPTO	Authorizes the daemon to call a Cryptographic Service Provider.
FACILITY	CDS.CSSM.DATALIB	Authorizes the daemon to call a Data Library (DL) Service Provider.

Class	Profile	Description
FACILITY	FWKERN.START.REQUEST	Permits FWKERN to issue the `start console` command and to start its servers. It also controls firewall administrator IDs that are allowed to issue the `fwdaemon` command to start and stop firewall servers.
STARTED	STC name	The STARTED class allows you to assign RACF identities to started procedures and jobs dynamically, using the `RDEFINE` and `RALTER` commands. Unlike the started procedures table, it does not require you to modify code or re-IPL in order to add or modify RACF identities for started procedures. It provides, in effect, a dynamic started procedures table.
CSFSERV	ICSF Services	Controlling use of Integrated Cryptographic Service Facility (ICSF) cryptographic services.

E.3.1 Planning for group definition

Determine the current group definitions:

```
LISTGRP * OMVS NORACF
```

Note: Bear in mind that two groups, SYS1 (or equivalent) and FWGRP, are required for the OS/390 Firewall Technologies configuration. If these groups already exist, note their group identifiers. If not, note available group identifiers for use in the definition of the required group(s).

Determine if the SYS1 group (or a logically equivalent group to contain UID=0 users) exists:

```
LISTGRP SYS1 OMVS
```

If the SYS1 (or equivalent) group does not exist, create it:

```
ADDGROUP SYS1 OMVS(GID(nnn))
```

Note: The installation-defined group ID for the SYS1 group that was identified in the previous steps is denoted by `nnn`.

E.3.2 Defining users and groups

Define the firewall startup address space to RACF or an equivalent security product:

- Define the FWKERN user.
- Define the firewall startup program as a started task. For example:

```
ADDGROUP FWGRP SUPGROUP(SYS1) OMVS(GID(nnn))
MKDIR '/u/fwkern' MODE(7,5,5)
ADDUSER FWKERN OMVS(HOME('/u/fwkern/') UID(0))
        DFLTGRP(FWGRP) AUTHORITY(CREATE) UACC(ALTER) PASSWORD(pw)
RDEFINE STARTED FWKERN STDATA(USER(FWKERN))
SETROPTS RACLIST(STARTED) REFRESH
```

Note: The installation-defined group ID for the FWGRP group that was identified in the previous steps is denoted by `nnn`; the password for the FWKERN user ID is denoted by `pw`. Choose this password with extreme care to avoid potential security exposures.

E.3.3 Granting users and groups authority to access firewall objects

Create the FWKERN.START.REQUEST resource profile:

```
RDEFINE FACILITY FWKERN.START.REQUEST UACC(NONE)
PERMIT FWKERN.START.REQUEST CLASS(FACILITY) ID(FWKERN) ACCESS(UPDATE)
SETROPTS CLASSACT(FACILITY)
```

Permit FWKERN access to start the servers:

```
RDEFINE STARTED ICAPSLOG.**  STDATA(USER(FWKERN) GROUP(FWGRP))
RDEFINE STARTED ICAPSOCK.**  STDATA(USER(FWKERN) GROUP(FWGRP))
RDEFINE STARTED ICAPPFTP.**  STDATA(USER(FWKERN) GROUP(FWGRP))
RDEFINE STARTED ICAPCFGS.**  STDATA(USER(FWKERN) GROUP(FWGRP))
RDEFINE STARTED ICAPSTAK.**  STDATA(USER(FWKERN) GROUP(FWGRP))
RDEFINE STARTED ICAPIKED.**  STDATA(USER(FWKERN) GROUP(FWGRP))
SETROPTS RACLIST(STARTED) REFRESH
```

Permit FWKERN read access to the TCP/IP data sets if necessary:

```
PERMIT 'TCPIP.**' ID(FWKERN) ACCESS(READ)
```

Permit FWKERN to read the BPX.SMF facility (for logging):

```
RLIST FACILITY BPX.SMF ALL
RDEFINE FACILITY BPX.SMF UACC(NONE)
PERMIT BPX.SMF CLASS(FACILITY) ID(FWKERN) ACCESS(READ)
```

Permit FWKERN to read the BPX.DAEMON facility:

```
RLIST FACILITY BPX.DAEMON
RDEFINE FACILITY BPX.DAEMON UACC(NONE)
PERMIT BPX.DAEMON CLASS(FACILITY) ID(FWKERN) ACCESS(READ)
```

Once you activate the BPX.DAEMON facility, you have to turn on program control using `SETROPTS WHEN(PROGRAM)`.

Program control

This command activates RACF program control, which includes both access control to load modules and program access to data sets. To set up access control to load modules, you must identify your controlled programs by creating a profile for each in the PROGRAM class. To set up program access to data sets, you must add a conditional access list to the profile of each program-accessed data set. Then, when program control is active, RACF ensures that each controlled load module is executed only by callers with the defined authority. RACF also ensures that each program-accessed data set is opened only by users who are listed in the conditional access list with the proper authority and who are executing the program specified in the conditional access list entry.

The firewall server programs must be marked as program controlled. It is also necessary to mark the SSL library SGSKLOAD as program controlled. This can be done by using the following commands:

```
RALTER PROGRAM * ADDMEM('ICA.SICALMOD'/'******'/NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'/'******'/NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

Note: Program control is the concept of having trusted applications. When it is active, processes will be marked dirty if they attempt to load programs from libraries that are not trusted. OS/390 USS also has the concept of trusted applications. In the UNIX file system, executable files may be tagged with the program-controlled extended attribute. If a user issues an OS/390 shell command or runs a program that does not have the program-controlled extended attribute, the process becomes dirty; in either case, the process is never cleaned. That is, the dirty bit remains, which will cause certain services to fail as a result. For more information see *OS/390 UNIX System Services Planning*, SC28-1890.

To use the configuration server, some setup is required. All user IDs that will use the firewall configuration GUI must be given permission explicitly to update the configuration through the configuration server. This includes user IDs that have superuser privileges or are members of the firewall group. For example:

```
RDEFINE FACILITY ICA.CFGSRV UACC(NONE)
PERMIT ICA.CFGSRV CLASS(FACILITY) ID(userid) ACCESS(UPDATE)
SETROPTS CLASSACT(FACILITY)
SETROPTS RACLIST(FACILITY) REFRESH
```

If the user IDs that will administer the firewall are not superusers (UID=0), add them to the FWGRP group as follows:

```
CONNECT userid GROUP(FWGRP)
```

The IKE server supports the ability to perform peer authentication using RSA signature mode. RSA Signature mode requires that digital certificates be stored in RACF and connected to a keyring. RACF provides digital certificate and keyring support using the `RACDCERT` command. The authorizations necessary to perform the basic `RACDCERT` actions are shown below:

```
RDEFINE FACILITY IRR.DIGTCERT.ADD UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.ADDRING UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.CONNECT UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.GENCERT UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.GENREQ UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.ADD CLASS(FACILITY) ID(userid) ACC(CONTROL)
PERMIT IRR.DIGTCERT.ADDRING CLASS(FACILITY) ID(userid) ACC(UPDATE)
PERMIT IRR.DIGTCERT.CONNECT CLASS(FACILITY) ID(userid) ACC(CONTROL)
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(userid) ACC(CONTROL)
PERMIT IRR.DIGTCERT.GENREQ CLASS(FACILITY) ID(userid) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(userid) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(userid) ACC(UPDATE)
SETROPTS RACLIST(FACILITY) REFRESH
```

Note: Here `userid` is the ID of the user who will be executing the `RACDCERT` command to store digital certificates.

Authority to the IRR.DIGTCERT.function resource in the FACILITY class allows a user to issue the `RACDCERT` command that is used to install and maintain digital certificates and keyrings in RACF. To issue the `RACDCERT` command, users must have one of the following authorities:

- The SPECIAL attribute
- Sufficient authority to access the resource IRR.DIGTCERT.function in the FACILITY class.
  - READ access to IRR.DIGTCERT.function to issue the `RACDCERT` command for themselves.
  - UPDATE access to IRR.DIGTCERT.function to issue the `RACDCERT` command for others.
  - CONTROL access to IRR.DIGTCERT.function to issue the `RACDCERT` command for SITE and CERTAUTH certificates. This authority also has other uses.

Refer to the *OS/390 Security Server (RACF) Command Language Reference*, SC28-1919 for a complete description of the facilities and authorizations needed to create and modify digital certificates and keyrings.

E.3.4 Installing the Open Cryptographic Services Facility (OCSF)

To perform the cryptographic functions needed by the IKE function you have to install and configure OCSF. In this section we will show how to perform this configuration. For more information about OCSF. please see *OS/390 Open Cryptographic Services Facility Application Developer's Guide and Reference*, SC24-5875.

To use OCSF services the following administration must be done:

- OCSF-related RACF FACILITY class profiles need to be defined and the FACILITY class made active if it is not already active.
- All of the programs, modules, and DLLs loaded in the OCSF application address space must be defined as program controlled. Programs or modules loaded from the traditional OS/390 search order (that is, STEPLIB, LINKLIST, etc.) need to reside in program-controlled libraries. Programs loaded from the UNIX file system must have the program-controlled extended attribute.
- OCSF application user IDs must be defined to RACF and permitted to the OCSF facility class profiles. Depending on whether your system is operating with OS/390 UNIX security or UNIX security, these user IDs will also need to be permitted to the BPX.SERVER facility class profile (when OS/390 UNIX security is in effect), or the OCSF daemon application must run with an effective UID of 0 (when UNIX security is in effect).

If you use OCSF from APF-authorized applications, you will need to turn on the APF-authorized extended attribute for the OCSF DLLs in the /usr/lpp/ocsf/lib and /usr/lpp/ocsf/addins directories. The SMP/E installation of OCSF does not ordinarily turn on the APF-authorized extended attribute. You can turn on the APF-authorized extended attribute using the `extattr +a` command.

APF authorization

The ISAKMP firewall (IKE function) runs as an APF-authorized application, so you have to turn on the APF-authorized extended attribute for the OCSF and OCEP dynamically loaded libraries.

Mark the OCSF programs in the OCSF UNIX Library as APF authorized and program controlled using the `extattr` command. To issue the `extattr` command the user ID has to have access to a specific RACF class profile:

```
RDEFINE FACILITY BPX.FILEATTR.APF ACC(NONE)
PERMIT BPX.FILEATTR.APF CLASS(FACILITY) USER(GIANCA) ACCESS(UPDATE)
RDEFINE FACILITY BPX.FILEATTR.PROGCTL ACC(NONE)
PERMIT BPX.FILEATTR.PROGCTL CLASS(FACILITY) USER(GIANCA) ACCESS(UPDATE)
SETROPTS CLASS(FACILITY) REFRESH
```

From the command prompt in the USS shell, issue the `extattr shell` command.

```
$ cd /usr/lpp/ocsf/lib
$ extattr +a *.dll  1
$ ls -E *.dll        2
-rwxr-xr-x  aps  2 OMVSKERN SYS1       49152 May 11 21:20 cdserprt.dll
-rwxr-xr-x  aps  2 OMVSKERN SYS1       86016 May 11 21:20 cdsibmut.dll
-rwxr-xr-x  aps  2 OMVSKERN SYS1       86016 May 11 21:20 cdskwtf.dll
-rwxr-xr-x  aps  2 OMVSKERN SYS1     4173824 May 11 21:20 cdsnspsp.dll
-rwxr-xr-x  aps  2 OMVSKERN SYS1      192512 May 11 21:20 cdsport.dll
-rwxr-xr-x  aps  2 OMVSKERN SYS1      188416 May 11 21:21 cdsrandm.dll
-rwxr-xr-x  aps  2 OMVSKERN SYS1      823296 May 11 21:19 cssm32.dll
lrwxrwxrwx       1 OMVSKERN SYS1          16 May 11 21:20 cssmmanp.dll -> cssmma
np_sl3.dll
-rwxr-xr-x  aps  2 OMVSKERN SYS1       36864 May 11 21:20 cssmmanp_sl3.dll
lrwxrwxrwx       1 OMVSKERN SYS1          16 May 11 21:20 cssmusep.dll -> cssmus
ep_sl3.dll
-rwxr-xr-x  aps  2 OMVSKERN SYS1       36864 May 11 21:20 cssmusep_sl3.dll

$ cd /usr/lpp/ocsf/addins
$ extattr +a *.so  1
$ ls -E *.so        2
-rwxr-xr-x  aps  2 OMVSKERN   SYS1      450560 May 11 21:20 ibmcca.so
-rwxr-xr-x  aps  2 OMVSKERN   SYS1      589824 May 11 21:20 ibmcl.so
-rwxr-xr-x  aps  2 OMVSKERN   SYS1     1474560 May 11 21:21 ibmcl2.so
-rwxr-xr-x  aps  2 OMVSKERN   SYS1     5701632 May 11 21:21 ibmdl2.so
-rwxr-xr-x  aps  2 OMVSKERN   SYS1     6856704 May 11 21:20 ibmocepdl.so
-rwxr-xr-x  aps  2 OMVSKERN   SYS1      425984 May 11 21:20 ibmoceptp.so
-rwxr-xr-x  aps  2 OMVSKERN   SYS1     1138688 May 11 21:20 ibmswcsp.so
-rwxr-xr-x  aps  2 OMVSKERN   SYS1       57344 May 11 21:21 ibmtp.so
-rwxr-xr-x  aps  2 OMVSKERN   SYS1     3563520 May 11 21:21 ibmtp2.so
-rwxr-xr-x  aps  2 OMVSKERN   SYS1     1069056 May 11 21:21 ibmwkcsp.so
```

Figure 226. extattr shell command

1 Give the APF authorization attributes to all files with the .dll suffix in the /usr/lpp/ocsf/lib directory and files with the .so suffix in /usr/lpp/ocsf/addins.

2 Using the `ls` command with the -E option, you can see the extended attributes of the HFS files. The `a` and `p` flags in the second column indicate that the files do have the APF-authorized and program-controlled attribute.

Define the following RACF FACILITY class profiles:

```
DEFINE FACILITY CDS.CSSM UACC(NONE)
RDEFINE FACILITY CDS.CSSM.CRYPTO UACC(NONE)
RDEFINE FACILITY CDS.CSSM.DATALIB UACC(NONE)
```

Grant the permission to use OCSF services to the FWKERN user ID:

```
PERMIT CDS.CSSM CLASS(FACILITY) ID(FWKERN) ACC(READ)
PERMIT CDS.CSSM.CRYPTO CLASS(FACILITY) ID(FWKERN) ACC(READ)
PERMIT CDS.CSSM.DATALIB CLASS(FACILITY) ID(FWKERN) ACC(READ)
```

Permit FWKERN access to the BPX.SERVER:

```
PERMIT BPX.SERVER CLASS(FACILITY) ID(FWKERN) ACC(READ)
```

Refresh the FACILITY class and PROGRAM class profile:

```
SETROPTS CLASS(FACILITY) REFRESH
```

Run the installation script from the UNIX System Services shell. You have to use a user ID with a UID of 0 (superuser) or have permission to issue the `su` command (BPX.SUPERUSER profile). This user ID must have access permission to the CDS.CSSM and the BPX.SERVER RACF facility class. Issue the following commands to give a user access permissions:

```
PERMIT CDS.CSSM CLASS(FACILITY) ID(userid) ACC(READ)
PERMIT CDS.CSSM.CRYPTO CLASS(FACILITY) ID(userid) ACC(READ)
PERMIT CDS.CSSM.DATALIB CLASS(FACILITY) ID(userid) ACC(READ)
PERMIT BPX.SERVER CLASS(FACILITY) ID(userid) ACC(READ)
PERMIT BPX.SUPERUSER  CLASS(FACILITY) ID(userid) ACC(READ)
SETROPTS CLASS(FACILITY) REFRESH
```

The installation scripts that you need to run are determined by which of the OCSF features you have installed. When you have the Security Level 3 feature or the Security Level 2 feature installed, you have to issue the installation scripts for the strong cryptographic facility in addition to scripts for the basic level feature.

If you have the Security Level 1, 2, or 3 feature or the French feature perform the following steps:

```
$ su
$ cd /usr/lpp/ocsf/bin
$ ocsf_install_basic_crypto
Installing CSSM...
CSSM Framework successfully installed
Installing IBMTP...
Addin successfully installed.
Installing IBMTP2...
Addin successfully installed.
Installing IBMCL...
Addin successfully installed.
Installing IBMCL2...
Addin successfully installed.
Installing IBMDL2...
Addin successfully installed.
Installing IBMWKCSP...
Addin successfully installed.
Installing IBMCCA...
Addin successfully installed.
```

Figure 227. ocsf_install_basic_crypto shell command

If the user does not have read access permission to the BPX.SERVER RACF FACILITY class, the command will fail with the message shown in Figure 228. You will also see the RACF error message on the MVS console as in Figure 229.

```
JRIVERA @ RA28:/usr/lpp/ocsf/bin>ocsf_install_basic_crypto
Installing CSSM...
CSSM Framework successfully installed
Installing IBMTP...
CSSM_Init failed
Exiting with bad rc = 65
Error: could not install IBMTP.
```

Figure 228. Error message from the ocsf_install_basic_crypto command

```
ICH408I USER(JRIVERA ) GROUP(WTCRES  ) NAME(JORGE RIVERA        )
  BPX.SERVER CL(FACILITY)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

Figure 229. Error message on MVS console

If you have Security Level 2 or 3 installed perform the following step as well:

```
$ ocsf_install_strong_crypto
Installing IBMSWCSP...
Addin successfully installed.Installing CSSM...
```

Figure 230. ocsf_install_strong_crypto shell command

Now, run the installation verification procedures (IVP). This verifies that you have installed and configured the product correctly.

```
$ cd /usr/lpp/ocsf/ivp
$ ocsf_baseivp
Starting OCSF base addins ivp

Initializing CSSM
CSSM Initialized

Attaching ibmwkcsp

 **************************************************************************
 * Portions of the IBM Software Cryptographic Service Provider or IBM    *
 * Weak Software Cryptographic Service Provider contain software         *
 * provided by RSA Data Security, Inc. Before use, see                   *
 * /usr/lpp/ocsf/README.FIRST for required terms.                        *
 **************************************************************************
Attach successful, Detaching ibmwkcsp
Detach of ibmwkcsp successful

Attaching ibmcca
Attach successful, Detaching ibmcca
Detach of ibmcca successful

Attaching ibmcl
Attach successful, Detaching ibmcl
Detach of ibmcl successful

Attaching ibmcl2
Attach successful, Detaching ibmcl2
Detach of ibmcl2 successful

Attaching ibmdl2
Attach successful, Detaching ibmdl2
Detach of ibmdl2 successful

Attaching ibmtp
Attach successful, Detaching ibmtp
Detach of ibmtp successful

Attaching ibmtp2
Attach successful, Detaching ibmtp2
Detach of ibmtp2 successful

Completed OCSF base addins ivp
```

Figure 231. ocsf_baseivp shell command

If you have Security Level 2 or 3 installed run the following script:

```
$ cd /usr/lpp/ocsf/ivp
$ ocsf_scivp
Starting OCSF strong crypto ivp

Initializing CSSM
CSSM initialized

Attaching ibmswcsp

 ***********************************************************************
 * Portions of the IBM Software Cryptographic Service Provider or IBM    *
 * Weak Software Cryptographic Service Provider contain software         *
 * provided by RSA Data Security, Inc. Before use, see                   *
 * /usr/lpp/ocsf/README.FIRST for required terms.                        *
 ***********************************************************************
Attach successful, Detaching ibmswcsp
Detach of ibmswcsp successful

Completed OCSF strong crypto ivp
```

Figure 232. ocsf_scivp shell command

E.3.5 Installing Open Cryptographic Enhanced Plug-Ins (OCEP)

Now we will install the OCSF plug-ins. For more information, please see *OS/390 Security Server Open Cryptographic Enhanced Plug-ins Guide and Reference*, SA22-7429. Ensure that the installation process enables the OCEP code for program control.

If you have not defined the following RACF facility class profiles, issue the commands below to define them:

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
SETROPTS CLASS(FACILITY) REFRESH
```

Give the FWKERN user ID permission to use the class profiles:

```
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(FWKERN) ACC(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(FWKERN) ACC(READ)
SETROPTS CLASS(FACILITY) REFRESH
```

Give permission to use the class profiles to the user ID that will install the code and run the IVP programs:

```
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(userid) ACC(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(userid) ACC(READ)
SETROPTS CLASS(FACILITY) REFRESH
```

Mark the OCSF-enhanced plug-in programs in the OCSF UNIX Library APF authorized and program controlled:

```
$ su
$ cd /usr/lpp/ocsf/addins
$ extattr +ap *.so
$ ls -E *.so
-rwxr-xr-x  aps  2 OMVSKERN SYS1      450560 May 11 21:20 ibmcca.so
-rwxr-xr-x  aps  2 OMVSKERN SYS1      589824 May 11 21:20 ibmcl.so
-rwxr-xr-x  aps  2 OMVSKERN SYS1     1474560 May 11 21:21 ibmcl2.so
-rwxr-xr-x  aps  2 OMVSKERN SYS1     5701632 May 11 21:21 ibmdl2.so
-rwxr-xr-x  aps  2 OMVSKERN SYS1     6856704 May 11 21:20 ibmocepdl.so
-rwxr-xr-x  aps  2 OMVSKERN SYS1      425984 May 11 21:20 ibmoceptp.so
-rwxr-xr-x  aps  2 OMVSKERN SYS1     1138688 May 11 21:20 ibmswcsp.so
-rwxr-xr-x  aps  2 OMVSKERN SYS1       57344 May 11 21:21 ibmtp.so
-rwxr-xr-x  aps  2 OMVSKERN SYS1     3563520 May 11 21:21 ibmtp2.so
-rwxr-xr-x  aps  2 OMVSKERN SYS1     1069056 May 11 21:21 ibmwkcsp.so
```

Figure 233. extattr shell command

Run the installation script from a UNIX System Services shell. To run this command, you must have superuser authority (UID=0).

```
$ cd /usr/lpp/ocsf/bin
$ ocep_install
Installing IBMOCEPTP...
Addin successfully installed.
Installing IBMOCEPDL...
Addin successfully installed.
```

Figure 234. ocep_install shell command

Now, run the Installation Verification Procedures (IVP). This verifies that you have installed and configured the product correctly.

```
$cd /usr/lpp/ocsf/ivp
$ocep_ivp
Starting OCEP IVP

Initializing CSSM
CSSM Initialized

Attaching ibmocepdl
Attach successful, Detaching ibmocepdl
Detach of ibmocepdl successful

Attaching ibmoceptp
Attach successful, Detaching ibmoceptp
Detach of ibmoceptp successful

Completed OCEP IVP
```

Figure 235. ocep_ivp shell command

E.3.6 ICSF/MVS authorization

OS/390 Firewall Technologies can take advantage of the encryption and decryption functions available in the new generation of System/390 processors. The firewall uses several of these functions in two ways:

- Encrypting and decrypting TCP/IP packet data in an IP tunnel.

- Encrypting signature data included as part of the IKE message flows. This encryption is only performed when RSA Signature Mode authentication is requested and the associated certificate was defined using the `RACDCERT` command specifying the ICSF keyword.

This support is provided by the combination of the Integrated Cryptographic Feature on the processor and the Integrated Cryptographic Service Facility/MVS (ICSF/MVS) software product.

To use this support, ICFS/MVS must be started and running. It is preferable to do this prior to starting TCP/IP; however, it can also be done when TCP/IP is active.

Note: The remaining configuration steps are applicable only to the use of hardware cryptography when encrypting and decrypting TCP/IP packet data in an IP tunnel.

If you plan to use this hardware crypto support and issue TCP/IP commands such as `OPING` from a user ID on the system where the firewall is running, this user ID must be permitted to access the ICSF/MVS cryptographic services (CSFSERV). This is because these are RACF controlled. Perform the following steps to set up profiles in the CSFSERV resource class and permit users to access these profiles:

Define the appropriate profiles in the CSFSERV class:

```
RDEFINE CSFSERV service-name UACC(NONE) other-optional-operands
```

The service names that the OS/390 Firewall Technologies use are CSFCKI, CSFDEC1, CSFENC1, CSFRNG, CSFCKM, and CSFOWH1. Note that if Triple-DES hardware crypto support is not available on your S/390 processor, the CSFCKM service is not used.

Permit user access to these profiles:

```
PERMIT profile-name CLASS(CSFSERV) ID(yourid) ACCESS(READ)
```

Activate the CSFSERV class and refresh the in-storage RACF profiles. This is done by the RACF administrator.

```
SETROPTS CLASS(CSFSERV)
SETROPTS RACLIST(CSFSERV)REFRESH
```

The MAXLEN installation option for hardware crypto determines the maximum length that can be used to encrypt and decrypt data using ICSF/MVS. Set the MAXLEN ICSF/MVS installation option to greater than 65527, since this is the maximum TCP/IP packet size. For more information, refer to the *ICSF/MVS Administrator's Guide*, SC23-0097.

E.4 OS/390 UNIX customization and firewall startup

This final section of the appendix shows what is necessary to configure within UNIX System Services to get the firewall function working.

E.4.1 Copying shell scripts

OS/390 Firewall Technologies contains the following executable shell scripts:

```
fwlogmgmt
```

```
   getmsg
```

Running shell scripts from locales that are not generated from code page IBM-1047 requires multiple copies of each shell script, one for each different locale's code page. You can use the `iconv` command to convert a shell script from one code page to another. For example, to convert the fwlogmgmt script to the Da_DK.IBM-277 locale, enter the following command:

```
iconv -f IBM-1047 -t Da_DK.IBM-277 /usr/lpp/fw/bin/fwlogmgmt > /tmp/fwlogmgmt
```

For more information about the `iconv` command, see *OS/390 UNIX System Services Command Reference*, SC28-1892.

For more information about customizing your locale, see *OS/390 UNIX System Services User's Guide*, SC28-1891, and *OS/390 UNIX System Services Planning*, SC28-1890.

E.4.2 Activate sample configuration files

It is important to preserve the owner, group, and mode settings when copying sample files. This can be done using the -p option of the `cp` command from a superuser (UID=0). For example:

```
    cp -p /usr/lpp/fw/etc/security/fwrules.cfg /etc/security/fwrules.cfg
```

A sample syslog.conf logging server configuration file is shipped with OS/390 Firewall Technologies in /usr/lpp/fw/etc.

To use this sample, copy it into the /etc directory.

If you do not install the sample syslog.conf file before IPLing your system and starting the firewall, an existing syslog.conf file will be used if one exists. If none exists, default logging will be in effect. Default logging sends all messages with a priority of error and above to the OS/390 operator console (to file /dev/console). You must create the special character file from a superuser by issuing:

```
   mknod /dev/console  c 9 0
```

See the *OS/390 UNIX System Services Command Reference*, SC28-1892, for further details on the `mknod` command.

In addition, the following files are shipped with OS/390 Firewall Technologies in /usr/lpp/fw/etc/security. They contain firewall default definitions:

- fwaudio.cfg - real audio configuration file
- fwdaemon.cfg - firewall servers configuration file
- fwobjects.cfg - network objects configuration file
- fwrules.cfg - firewall configured filter rules configuration file
- fwservices.cfg - services configuration file
- fwsocks.cfg - SOCKS configuration file
- logmgmt.cfg - log management configuration file
- fwahtran.cfg - AH transform configuration file
- fwesptran.cfg - ESP transform configuration file
- fwkeypol.cfg - key policy configuration file
- fwkeyprop.cfg - key proposal configuration file
- fwkeyring.cfg - keyring configuration file
- fwkeytran.cfg - key transform configuration file

If you are not migrating from a previous release of OS/390 Firewall Technologies, you must copy these files into the /etc/security directory during installation, if they are not already there.

If you are migrating from a previous release, use the command `fwmigrate` to preserve your current configuration, and copy the following files into the /etc/security directory, if they are not already there. Also see *OS/390 V2R10.0 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835 for more information.

- fwahtran.cfg
- fwesptran.cfg
- fwkeypol.cfg
- fwkeyprop.cfg
- fwkeyring.cfg
- fwkeytran.cfg

Whether or not you are migrating from a previous release of OS/390 Firewall Technologies, you must copy the following files into the /etc/security directory during installation:

- fwguicmds.En_US
- fwguicmds.Ja_JP (if Japanese version is installed)

In our configuration we changed the file /etc/syslog.conf. Please see our configuration below:

```
 BROWSE -- /etc/syslog.conf ------------------------ Line 00000000 Col 001 034
 Command ===>                                                 Scroll ===> CSR
********************************* Top of Data **********************************
#local0.* /tmp/firewall.local0.log
#local4.* /tmp/firewall.local4.log
#syslog.* /tmp/firewall.syslog.log
*.*      /tmp/firewall.all.log
******************************** Bottom of Data ********************************
```

Figure 236. syslogd configuration file

Using this configuration all messages directed to the syslogd server will be written in /tmp/firewall.all.log.

E.4.3 Defining the firewall stack

To define the firewall stack we have to use the `fwstack` command in the UNIX System Services shell prompt:

```
$ fwstack cmd=add stack=tcpipb
```

Figure 237. Defining the TCPIPB stack to the firewall kernel

In this release the firewall code can control many TCP/IP stacks in the same OS/390 image.

E.4.4 Configuring firewall servers

All the OS/390 Firewall Technologies servers run in their own address spaces. Servers are controlled by the control task running in the firewall kernel referred to as the FWKERN address space.

The FWKERN address space must be started before any of the servers are started. All requests to start, stop, or query the firewall servers (either collectively or individually) are made through the FWKERN control task through the `START`, `STOP`, or `MODIFY` commands, which you issue from the OS/390 operator console.

Use the `fwdaemon` command to list and change server configuration attributes, query server status, and start and stop servers.

At this time we only have to define four servers to the firewall stack. Go to the UNIX System Services shell prompt and use the `fwdaemon` command:

```
$ fwdaemon cmd=change daemon=syslogd started=yes
$ fwdaemon cmd=change daemon=fwstackd started=yes
$ fwdaemon cmd=change daemon=isakmpd started=yes
$ fwdaemon cmd=change daemon=cfgsrv started=yes \
    daemonopts="-f /etc/security/fwcfgsrv.kdb 1 -p 1014" 2
```

Figure 238. Defining firewall servers

1 is the key database file for the SSL connection. When you create the key database for the firewall client, follow the steps in E.4.6, “Firewall GUI configuration client” on page 368.

2 is the TCP/IP port number where the server will be listening.

E.4.5 Configuring filter rules for the configuration client

Now the firewall server has to be configured to support the configuration client connection. We have to define some filter rules to permit the connection to the secure interface. We have to create a filter rule to permit all connections through the secure interface or only permit some services, such as the configuration client service using port 1014. In our environment, we permit all types of traffic in the secure interface.

The following steps show how to customize a filter rule to permit all types of traffic to the secure interface(s). You have to go to the UNIX System Services shell environment using the authorized user ID defined previously and issue the following commands:

```
GIANCA @ RA03:/u/gianca>su 1
GIANCA @ RA03:/u/gianca>fwfrule cmd=add name="R2612.Permit.All.Secure" \
    desc="Permit all in the Secure Interface" type=permit protocol=all \
    srcopcode=any srcport=0 destopcode=any destport=0 interface=secure \
   routing=local direction=both log=yes    2
GIANCA @ RA03:/u/gianca>fwfrule cmd=list name="R2612.Permit.All.Secure"
               id = 529
             type = permit
             name = R2612.Permit.All.Secure
             desc = Permit all traffic in the secure interface
         protocol = all
        srcopcode = any
          srcport = 0                        3
       destopcode = any
         destport = 0
        interface = secure
          routing = local
        direction = both
              log = yes
           tunnel =
         fragment =
```

Figure 239. Defining firewall configuration client rules

1 To use the firewall commands you have to be in *superuser mode* or a member of the FWGRP group. Using the `su` command, you can change to superuser mode.

2 The `fwrule` command creates a rule that allows any kind of traffic over the secure interface to the local host.

3 List the rule you have created to get the ID. It will be used in the `fwservice` command.

```
GIANCA @ RA03:/u/gianca>fwservice cmd=create name="R2612.Permit.All.Secure" \
    desc="Permit all in the secure interface" rulelist=529/f,529/b  4
GIANCA @ RA03:/u/gianca>fwservice cmd=list name="R2612.Permit.All.Secure"
               id = 509
             name = R2612.Permit.All.Secure
             desc = Permit all in the secure interface
         rulelist = 529/f,529/b
              log =
         fragment =                          5
           tunnel =
             time =
            month =
              day =
          weekday =
       timefilter =
```

Figure 240. Defining firewall configuration client services

4 Create a service that contains the rule created previously.

5 List the service you have created to get the ID. It will be used in the `fwconns` command.

```
GIANCA @ RA03:/u/gianca>fwnwobj cmd=add name="Network.9.0.0.0"  \
  desc="IBM Intranet" type=network addr=9.0.0.0 mask=255.0.0.0    6
GIANCA @ RA03:/u/gianca>fwnwobj cmd=add name="Host.9.24.104.33"  \
  desc="Host 9.24.104.33" type=host addr=9.24.104.33 mask=255.255.255.255  7
GIANCA @ RA03:/u/gianca>fwnwobj cmd=list name="Host.9.24.104.33"
               id = 501
             type = Host
             name = Host.9.24.104.33
             desc = Host 9.24.104.33
             addr = 9.24.104.33                    8
             mask = 255.255.255.255
        startaddr =
          endaddr =

GIANCA @ RA03:/u/gianca>fwnwobj cmd=list name="Network.9.0.0.0"
               id = 502
             type = Network
             name = Network.9.0.0.0
             desc = IBM Intranet
             addr = 9.0.0.0
             mask = 255.0.0.0                   9
        startaddr =
          endaddr =
```

Figure 241. Defining firewall configuration client network objects

6 Create a network object that represents your intranet. In this case, we used the IBM intranet.

7 Create a network object that represents the host where the firewall server is running.

8 List the network object to get the ID. It will be used in the `fwconns` command.

9 List the network object to get the ID. It will be used in the `fwconns` command.

```
GIANCA @ RA03:/u/gianca>fwconns cmd=create name="R2612.Permit.All.Secure" \
    desc="Permit all traffic over secure interface" source=502  \
    destination=501 servicelist=509 10
GIANCA @ RA03:/u/gianca>fwfilter cmd=update   11
ICAC1577i Processing firewall TCP/IP stack TCPIPB:


Filter support (level 2.80) initialized at 13:54:15 on Jul-06-1999


                                 12
ICAC1531w Unable to inform the sock daemon to refresh configuration data.
```

Figure 242. Defining configuration client connection and updating filter rules

10 Create a connection associating the two network objects with the service.

11 Update the filter rules.

12 You will receive this warning message unless you have the SOCKS server running. If you do not need it, ignore this message.

Now you can start any type of connection with the host 9.24.104.33 using a workstation in the IBM Intranet.

All other configurations related to the IKE function will be made using the firewall configuration client GUI. For more information about the firewall commands, see *OS/390 V2R10.0 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835.

E.4.6 Firewall GUI configuration client

The firewall configuration server uses the Secure Sockets Layer (SSL) protocol of OS/390 for communication between the graphical user interface (GUI) clients and the server.

The administrator must run the GSKKYMAN utility to create a new key database, if one was not already created. The administrator needs to use the **Create a self-signed certificate** option to create and store a self-signed certificate, then use the **Store encrypted database password** option. See *OS/390 System Secure Sockets Layer Programming Guide and Reference*, SC24-5877, for information about the GSKKYMAN utility.

GSKKYMAN uses the DLLs that are installed with System SSL and must have access to these at run time. GSKKYMAN must also have access to the message catalogs. /bin includes a symbolic link to /usr/lpp/gskssl/bin/gskkyman. Therefore, if your PATH environment variable contains this directory, you will find the GSKKYMAN utility. If your PATH environment variable does not contain this directory, add /usr/lpp/gskssl/bin to your PATH using the following:

```
PATH=$PATH:/usr/lpp/gskssl/bin
```

The directories /usr/lib/nls/msg/C and /usr/lib/nls/msg/En_US.IBM-1047 (and /usr/lib/nls/msg/Ja_JP for JCPT272 installations) include symbolic links to the message catalogs for GSKKYMAN. If they do not include these links, add /usr/lpp/gskssl/lib/nls/msg to your NLSPATH using the following command:

```
export NLSPATH=$NLSPATH:/usr/lpp/gskssl/lib/nls/msg/%L/%N
```

This setting assumes that your environment has the LANG environment variable set to En_US.IBM-1047 (Ja_JP for JCPT272 installations that expect Japanese messages and prompts). If LANG is not set properly, set the NLSPATH environment variable using the following command:

```
export NLSPATH=$NLSPATH:/usr/lpp/gskssl/lib/nls/msg/En_US.IBM-1047/%N
```

Or for JCPT272 installations that expect Japanese messages and prompts:

```
export NLSPATH=$NLSPATH:/usr/lpp/gskssl/lib/nls/msg/Ja_JP/%N
```

The DLLs for System SSL are installed in a partitioned data set (PDS). These DLLs are not installed in the LINKLIB or LPALIB by default. To access these DLLs, if they have not been placed in LINKLIB or LPALIB, you must set the STEPLIB environment variable to find the DLLs. Consult your system programmer for the high-level qualifier of the System SSL PDS. In the example below, the high-level qualifier for the System SSL PDS is `GSK`. In the following command, replace the value to meet your installation:

```
export STEPLIB=GSK.SGSKLOAD
```

Using the GSKKYMAN utility to create a key database for the configuration server:

```
GIANCA @ RA03:/u/gianca/firewall>gskkyman


            IBM Key Management Utility

Choose one of the following options to proceed.

   1  - Create new key database
   2  - Open key database
   3  - Change database password

   0  - Exit program

Enter your option number: 1
Enter key database name or press ENTER for "key.kdb": fwcfgsrv.kdb
Enter password for the key database.......>
Enter password again for verification.....>
Should the password expire? (1 = yes, 0 = no) : 0

The database has been successfully created, do you want to continue to work with
 the database now? (1 = yes, 0 = no) : 1                             Enter your
```

Figure 243. GSKKYMAN utility: main menu

Enter option `1` to create a new .kdb file and press Enter. Type the key database name and press Enter. We used fwcfgsrv.kdb. Enter the password for the key database file twice (press Enter after you type the password). Do not forget this password because each time you have to access this .kdb file you will be prompted for this particular password. Type `0` to ensure that the password does not expire. Then type `1` and press Enter to continue to work with this database.

```
          Key database menu

Current key database is /u/gianca/firewall/os390/fwcfgsrv.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 5
Enter version number of the certificate to be created (1, 2, or 3):  3
Enter a label for this key................> Firewall GUI Key
Select desired key size from the following options (512):
    1:    512
    2:    1024
Enter the number corresponding to the key size you want: 1
Enter certificate subject name fields in the following.
    Common Name (required)................> Firewall GUI Key
    Organization (required)..............> IBM
    Organization Unit (optional).........> ITSO
    City/Locality (optional).............> Cary
    State/Province (optional)............> North Carolina
    Country Name (required 2 characters)..> US
Enter number of valid days for the certificate :  365
Do you want to set the key as the default in your key database? (1 = yes, 0 = no
): 1
Do you want to save the certificate to a file? (1=yes, 0=no): 0

Please wait while self-signed certificate is created...

Your request has completed successfully, exit gskkyman? (1=yes,0=no) : 0
```

Figure 244. GSKKYMAN: creating a self-signed certificate

Type 5 and press Enter to create a self-signed certificate. You can create a certificate request and have it signed by a CA, but in this case we used a self-signed certificate. Type 3 and press Enter to create a Version 3 certificate. The version number refers to the X.509 standard version number. Type a label for the key and press Enter. Type a common name and press Enter. Type an organization name and press Enter. All the other fields are optional. Then type the number of days this certificate should be valid and press Enter. We chose one year. Type 1 and press Enter to set this key as the default key in this database. Type 0 and press Enter not to save this certificate into a file. Type 0 and press Enter to return to the previous menu.

```
           Key database menu

Current key database is /u/gianca/firewall/os390/fwcfgsrv.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 11

The encrypted password has been stored in file /u/gianca/firewall/os390/fwcfgsrv
.sth

Your request has completed successfully, exit gskkyman? (1=yes, 0=no) >: 1
```

Figure 245. GSKKYMAN: storing an encrypted password

Type `11` and press Enter to store the encrypted database password in a stashed file. Finally, type `1` and press Enter to leave GSKKYMAN.

Now you have two files in the directory in which you were running GSKKYMAN: a key database file and a stash file whose file names in our example were fwcfgsrv.kdb and fwcfgsrv.sth, respectively. The .kdb file contains the keys and certificates you created and the .sth file contains the database password. Copy these two files to the directory you specified in the fwdaemon daemonopts parameter for the CFGSRV server.

E.4.7 Start the firewall kernel

Before starting the firewall code you have to be sure that all the previous configurations are available: the TCP/IP configuration and the BPXPRMxx configuration. ICA.SICALMOD and GSK.SGSKLOAD must be APF authorized and the firewall code at /usr/lpp/fw must be mounted and available. We inserted both libraries in the LNKLSTxx to make them available to the firewall servers and to make it unnecessary to update many procedures.

You have to copy all members of the ICA.SICAPROC data set to a library in the JES concatenated started procedure libraries, or concatenate this data set to SYS1.PROCLIB in the JES procedure.

Starting the firewall kernel:

```
S FWKERN
$HASP100 FWKERN   ON STCINRDR
IEF695I START FWKERN   WITH JOBNAME FWKERN   IS ASSIGNED TO USER FWKERN
 , GROUP OMVSGRP
$HASP373 FWKERN   STARTED
IEF403I FWKERN - STARTED - TIME=16.16.38
ICAM1057i Release 2.8.0 Service Level 0000000. Created on Jun 22 1999.
$HASP100 ICAPSLOG ON STCINRDR
$HASP373 ICAPSLOG STARTED
IEF403I ICAPSLOG - STARTED - TIME=16.16.42
ICAM1069i Daemon SYSLOGD has been started.
$HASP100 ICAPCFGS ON STCINRDR
$HASP373 ICAPCFGS STARTED
IEF403I ICAPCFGS - STARTED - TIME=16.16.48
ICAM1069i Daemon CFGSRV has been started.
$HASP100 ICAPIKED ON STCINRDR
$HASP373 ICAPIKED STARTED
IEF403I ICAPIKED - STARTED - TIME=16.16.57
ICAM1069i Daemon ISAKMPD has been started.
$HASP100 ICAPSTAK ON STCINRDR
$HASP373 ICAPSTAK STARTED
IEF403I ICAPSTAK - STARTED - TIME=16.17.18
ICAM1069i Daemon FWSTACKD has been started.
ICAM1003i FWKERN initialization complete.
```

Figure 246. Starting FWKERN

To start the firewall kernel go to the OS/390 console and type `S FWKERN` and press Enter. FWKERN will start all the firewall servers configured previously. You have to check the message ICAM1003I to be sure that all servers were started successfully.

There are some console commands you can use to check the firewall status, to stop or start a firewall server, and to stop FWKERN.

Some examples are:

```
F FWKERN,QUERY ISAKMPD
ICAM1001i Firewall daemon ISAKMPD status is READY and process id is 145
335544366.
F FWKERN,QUERY SYSLOGD
ICAM1001i Firewall daemon SYSLOGD status is READY and process id is 147
33554455.
F FWKERN,QUERY FWSTACKD
ICAM1001i Firewall daemon FWSTACKD status is READY and process id is 149
50331695.
+JSX015 JESX     SLU=RMT3     T011,RTNCD=1000 LU NOT AVAILABLE
F FWKERN,QUERY CFGSRV
ICAM1001i Firewall daemon CFGSRV status is READY and process id is 152
67108917.
F FWKERN,QUERY LEVEL
ICAM1057i Release 2.8.0 Service Level 0000000. Created on Jun 22 1999.
```

Figure 247. Examples of FWKERN console commands

After starting the FWKERN, check the socket connections that are open. You can use either the console command `Display TCPIP,TCPIPB,Netstat,SOCKETS` or the UNIX shell command `netstat -p tcpipb -s`. We used the shell command:

```
GIANCA @ RA03:/u/gianca>netstat -p tcpipb -s
MVS TCP/IP onetstat CS V2R8       TCPIP Name: TCPIPB          14:36:43
Sockets interface status:
Type   Bound to              Connected to          State    Conn
====   ========              ============          =====    ====
Name: FTPD1     Subtask: 007E7390
Stream 0.0.0.0..21           0.0.0.0..0            Listen   00000048
Name: HODSRV3   Subtask: 007E4E78
Stream 0.0.0.0..8999         0.0.0.0..0            Listen   00000021
Name: HODSRV3   Subtask: 007E7390
Stream 0.0.0.0..8989         0.0.0.0..0            Listen   0000003C
Name: ICAPCFGS  Subtask: 007E7420
Stream 0.0.0.0..1014  1      0.0.0.0..0            Listen   0000014A
Name: ICAPIKED  Subtask: 007E7420
Dgram  9.24.104.33..500      *..*                  UDP      0000014C
Dgram  192.168.100.100..500  *..*  2               UDP      0000014D
Dgram  172.16.233.4..500     *..*                  UDP      0000014E
Name: ICAPSLOG  Subtask: 007E7420
Dgram  0.0.0.0..514  3       *..*                  UDP      00000149
Name: TCPIPB    Subtask: 007D2AB0
Stream 0.0.0.0..9923         0.0.0.0..0            Listen   00000019
Name: TCPIPB    Subtask: 007D2CD0
Stream 0.0.0.0..8823         0.0.0.0..0            Listen   00000018
Name: TCPIPB    Subtask: 007D2E68
Stream 0.0.0.0..7723         0.0.0.0..0            Listen   00000017
Name: TCPIPB    Subtask: 007E1630
Stream 0.0.0.0..23           0.0.0.0..0            Listen   00000015
Name: TCPIPB    Subtask: 007E1AC8
Stream 0.0.0.0..6623         0.0.0.0..0            Listen   00000016
Name: TCPIPB    Subtask: 007E33F8
Stream 127.0.0.1..1025       127.0.0.1..1026       Establsh 00000013
Stream 0.0.0.0..1025         0.0.0.0..0            Listen   0000000C
Name: TCPIPB    Subtask: 007E3BF8
Dgram  0.0.0.0..1252         *..*                  UDP      000001FF
Name: TCPIPB    Subtask: 007EC1B0
Stream 127.0.0.1..1026       127.0.0.1..1025       Establsh 00000012
Name: WEBSRV    Subtask: 007EC9B0
Stream 0.0.0.0..80           0.0.0.0..0            Listen   00000020
```

Figure 248. Report from netstat -p OS/390 UNIX command

1 This is the firewall server CFGSRV server listening on port 1014.

2 This is the firewall server ISAKMPD server waiting for UDP packets on all active interfaces on port 500.

3 This is the firewall server syslogd server waiting for UDP packets on port 514.

Please see *OS/390 V2R10.0 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835 for more information about the commands and firewall configuration.

E.4.8 Installing the firewall GUI configuration client

Now we will install the GUI on a Windows 95 or Windows NT system. The GUI code is located in the /usr/lpp/fw/bin/fwtech.zip file. We have to move this code to the workstation using FTP or any other file transfer program. We did this using the FTP server in OS/390.

The workstation prerequisites to install the firewall GUI code are:

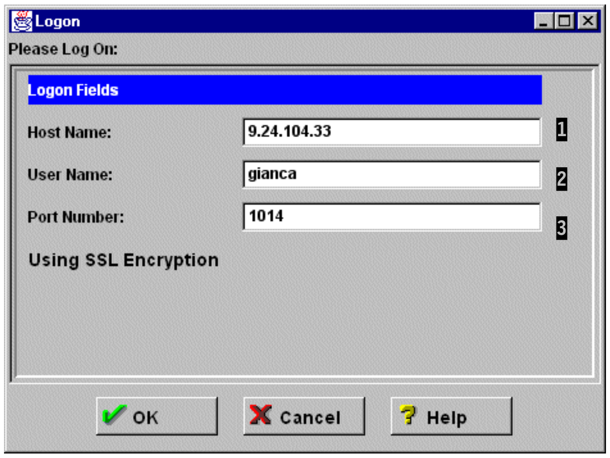
- Windows NT 4.0 or Windows 95
- A browser with Java and frames support
- A ZIP tool such as WinZip32 that handles long file names

Follow the steps below to install the GUI:

- Download the code /usr/lpp/fw/fwtech.zip to your workstation. Do not forget to transfer the code in binary format.
- Unzip the code and run the SETUP.EXE program.
- Follow the instructions given to install the code.

E.4.9 Using the configuration client

Start the GUI from the Windows start menu. You will see the following window:



Host Name: 9.24.104.33 1
User Name: gianca 2
Port Number: 1014 3
Using SSL Encryption

Figure 249. Firewall configuration client menu

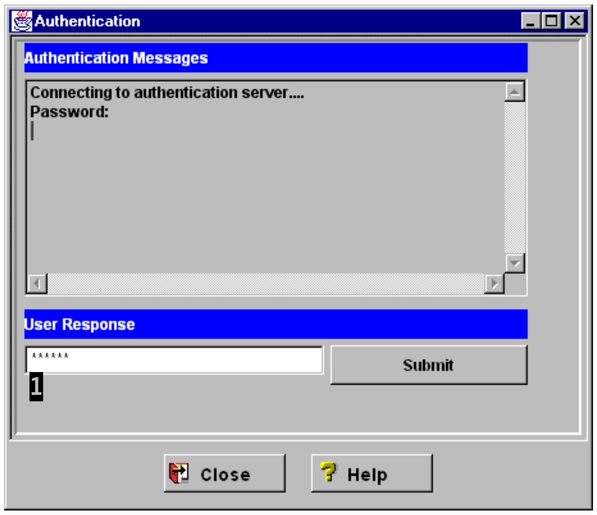
1 This is the host IP address where the firewall is running.

2 The administrator user name you defined previously.

3 The port number you configured in the `FWDAEMON` command (-p parameter).

Note: Do not forget that this user must be permitted to the ICA.CFGSRV profile.

Click **OK** to continue. You will receive a password prompt:



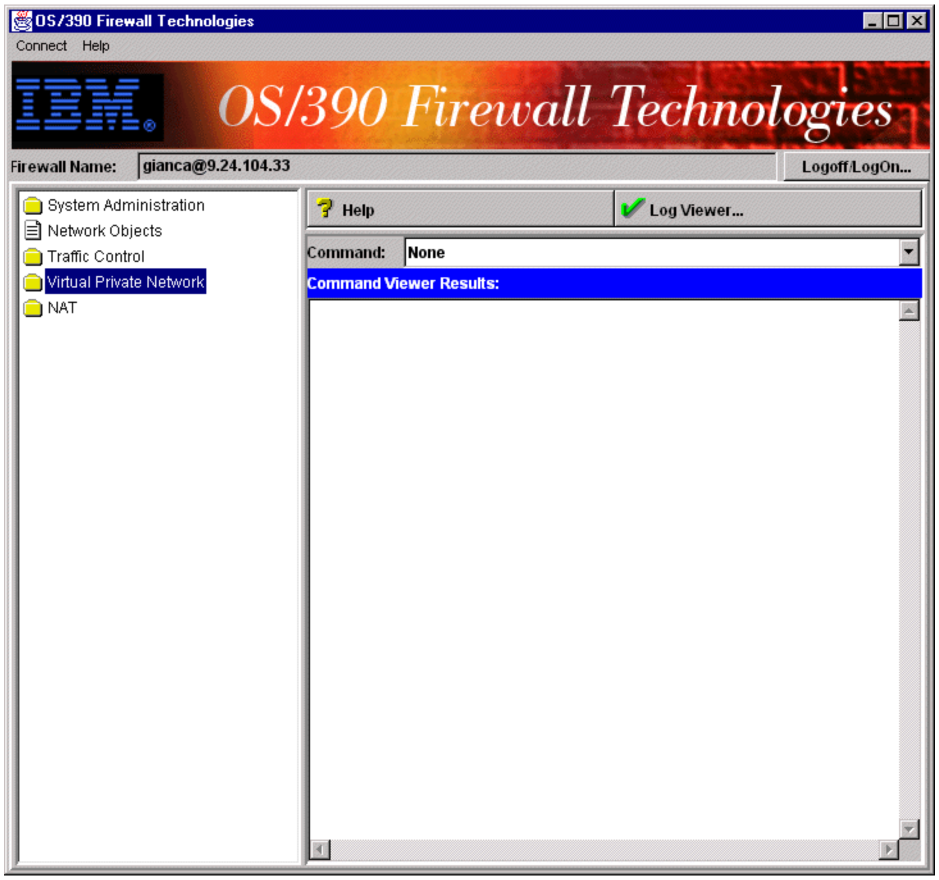
Connecting to authentication server....
Password:

******

Figure 250. Firewall GUI password prompt

1 This is the RACF password for the user ID.

Click **Submit** to continue. Now you will see the firewall configuration client main window:



Firewall Name: gianca@9.24.104.33

Command: None

Figure 251. Firewall configuration client main window

For more information about how to use the firewall configuration client see *OS/390 V2R10.0 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835.

E.5 Configuration for three VPN scenarios

This section describes in detail how to define the three scenarios introduced in 6.5.1.1, “Branch office connection network” on page 274, 6.5.1.2, “Business partner/supplier network” on page 277, and 6.5.1.3, “Remote access network” on page 279. Your other references for the configuration of IPSec are:

- A white paper entitled "VPN Interoperability Demonstrated for OS/390" by Lin Overby of z/OS Communications Server Strategy and Design and Thomas Cosenza of z/OS Communications Server System Test, available at `www.ibm.com/servers/eserver/zseries/networking/pdf/GM13-0029-00.pdf`
- *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 1: Configuration and Routing*, SG24-5227
- *SecureWay CS OS/390 V2R8 TCP/IP: Guide to Enhancements*, SG24-5631
- *A Comprehensive Guide to Virtual Private Networks, Volume I: IBM Firewall, Server and Client Solutions*, SG24-5201
- *A Comprehensive Guide to Virtual Private Networks, Volume III: Cross-Platform Key and Policy Management,* SG24-5339
- *SecureWay CS OS/390 V2R8 TCP/IP: Guide to Enhancements*, SG24-5631
- *OS/390 V2R10.0 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835

Blank worksheets that may be used to plan your own configuration are available in many of the above manuals. However, for your convenience, we have also included such worksheets in Appendix F, “VPN planning worksheets” on page 439.

E.5.1 Branch office interconnect over the Internet scenario details

Suppose your company wants to minimize the costs incurred in communicating to and among its own branches. Today, your company uses frame relay or leased lines, but you want to explore other options for transmitting internal confidential data that is less expensive, more secure, and globally accessible. By exploiting the public Internet, you can easily establish a branch office virtual private network (VPN) to meet the needs of your company.

In this scenario, your company wants to establish a VPN between the corporate gateway and the gateway of a branch office. The first thing you should do is gather the information you need to complete your configurations. The following planning checklists illustrate the type of information you will need to use.

In our case, we know that we are working with a trusted network, so we decided to use just low security level in our key management and data management policies. You might choose to use a high security level for key management and low security level for data management.

Table 31. Initial design worksheet for dynamic connections

This is the information needed to create dynamic VPNs	Answers
What is the type of connection to be created? - Gateway-to-gateway - Host-to-gateway - Gateway-to-host - Host-to-host - Gateway-to-dynamic IP user - Host-to-dynamic IP user	OS/390 gateway-to-OS/390 gateway
What type of security and system performance is required to protect the keys? - Highest security, lowest performance - Balance security and performance - Lowest security and highest performance - Add your own definition	lowest security and highest performance
What type of security and system performance is required to protect the data? - Highest security, lowest performance - Balance security and performance - Lowest security and highest performance - Add your own definition	lowest security and highest performance

The next step is the definition of the key management policies; remember that these definitions can be used later for other VPN configurations. Table 32 is just an example; you can modify it if necessary.

Table 32. Key management planning

VPN Parameter	Value
Key Management Policy : Key Policy, Proposal, Transform	
Key Policy Name : Key Policy Medium	
Initiator Negotiation	Main
Responder Negotiation	Main
Key Proposal Name : Key Proposal Low-Med Security	
1. Key Transform Name : Lowest Security and High Performance	
Authentication Method	RSA
Hash Algorithm	MD5
Encryption Algorithm	DES_CBC_8
Diffie-Hellman Group	Group 1
Maximum Key Lifetime	480
Maximum Size Limit	0
Key Lifetime Range	60-1440
Size Limit Range	0

VPN Parameter	Value
2. Key Transform Name : Balance Security and Performance	
Authentication Method	RSA
Hash Algorithm	SHA
Encryption Algorithm	DES_CBC_8
Diffie-Hellman Group	Group 1
Maximum Key Lifetime	480
Maximum Size Limit	0
Key Lifetime Range	60-1440
Size Limit Range	0

To define your data management policies, as before you can use this information in order to create other VPN definitions. Please see Table 33:

Table 33. Data management planning

Data Management Policy : Data Policy, Proposal, AH and ESP Transform:	
Data Policy Name : Data Policy based on low_ESP Transport	
Perfect Forward Secrecy (PFS)	None
Data Proposal Name : Data Proposal based only on low_ESP Transport	
1. ESP Transform Object Name : IBM_low_ESP_transport	
ESP Encapsulation Mode	Transport
ESP Authentication Algorithm	HMAC_MD5
ESP Encryption Algorithm	DES_CBC_8
ESP Maximum Data Lifetime	30
ESP Maximum Size Limit	0
ESP Data Lifetime Range	30-480
ESP Size Limit Range	0

Next we define the attributes of the dynamic tunnel policy. Remember that this information is going to be associated later with the data management policy.

Table 34. Dynamic tunnel policy

Dynamic Tunnel Policy:	
Initiation	Either
Connection Lifetime	1440

Finally, you plan the specific definitions of your tunnel; information such as key server management and authentication information will be put here.

Table 35. Key server planning

VPN Parameter	Value
Authentication Information:	
Remote Key Server	192.168.100.100
Authentication Method	RSA
Shared Key	Non-used
Certificate Authority:	
Racdcert Label	CA Entrust
Keyring:	
User ID	FWKERN
Keyring Name	ikekeyrin2
Dynamic Connection:	
Source	192.168.100.99
Destination	192.168.100.100
Source Port	0
Destination Port	0
Automatic Activation	No
Protocol	all
Remote Key Server	192.168.100.100
Key Servers:	
Local Key Server ID Type	IPV4
Local Key Server ID	192.168.100.99
Remote Key Server ID Type	USER@FQDN
Remote Key Server ID	jessicab@itso.ral.ibm.com

E.5.1.1 Scenario overview

A branch office has the following attributes:

- It is an extension of the corporate intranet.
- It is maintained in a geographically dispersed location.
- It is a trusted network.
- Network-to-network security is the main concern.

Figure 252 is an example of a branch office environment:

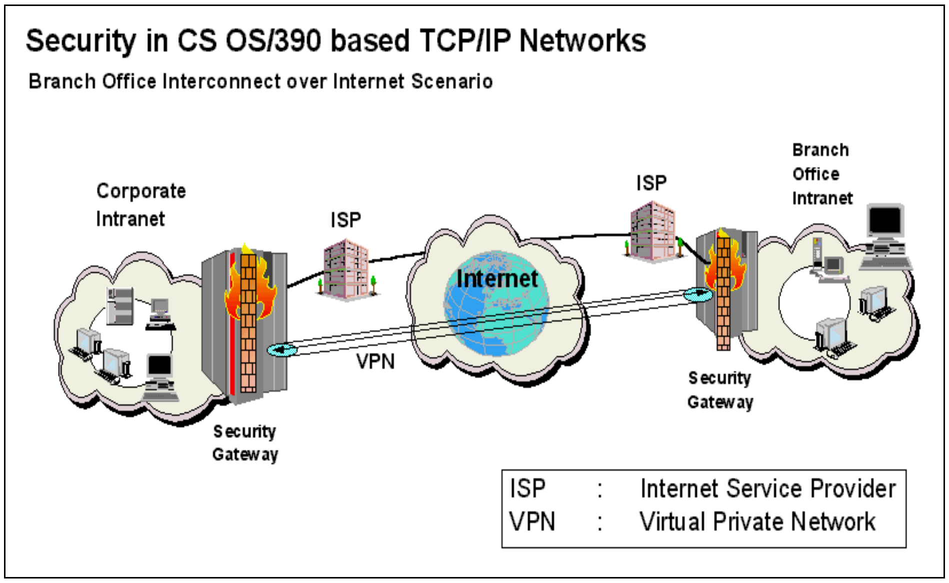

Security in CS OS/390 based TCP/IP Networks
Branch Office Interconnect over Internet Scenario
Corporate Intranet
ISP
Internet
ISP
Branch Office Intranet
VPN
Security Gateway
Security Gateway
ISP : Internet Service Provider
VPN : Virtual Private Network

Figure 252. Branch office scenario

Note that the connection is between the two security gateways, in this case two OS/390 systems. The configuration is made in transport mode. In our test laboratory, we recreated the branch office scenario as shown in Figure 253:

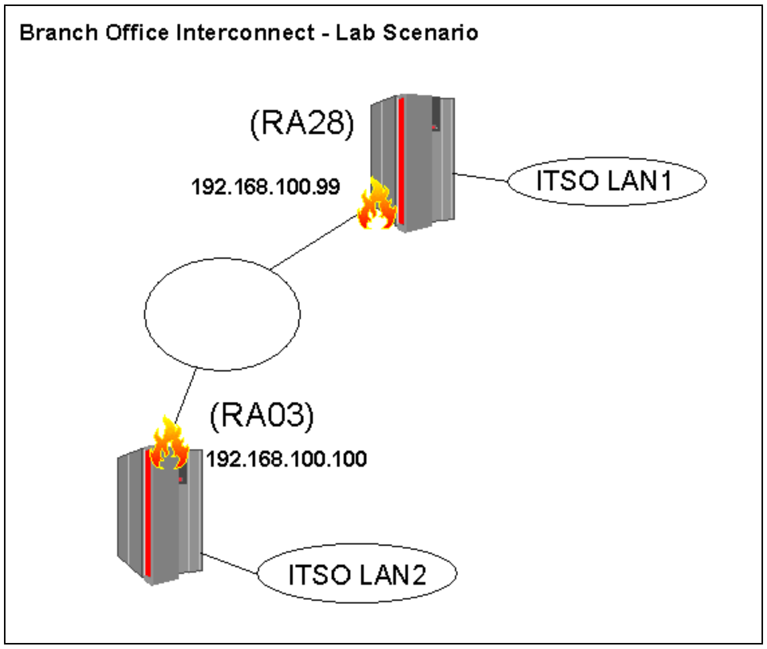


Figure 253. Branch office scenario - our lab

E.5.1.2 Scenario setup

We create the key management, data management and dynamic tunnel definitions using the OS/390 firewall GUI. Please refer to “Policies” on page 287 to see how to create and manage key management, data management and dynamic tunnel definitions.

Key management definitions
In this example, we use the definitions referred to in Table 32 on page 377.

Log on to the configuration client. In the main window double-click **Virtual Private Networks**, then expand the **Dynamic** tree, and expand the **VPN Connection Templates** trees. Then expand the **Key Management** tree. You will see the window shown in Figure 254.

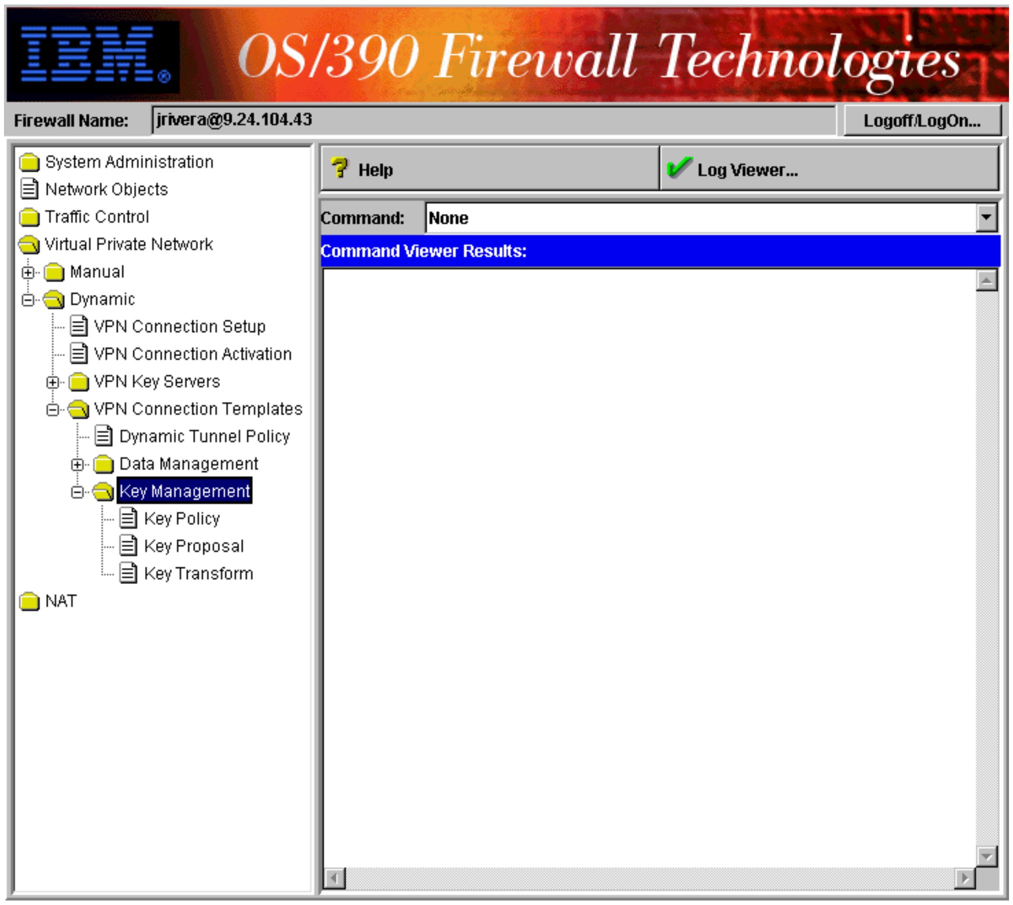
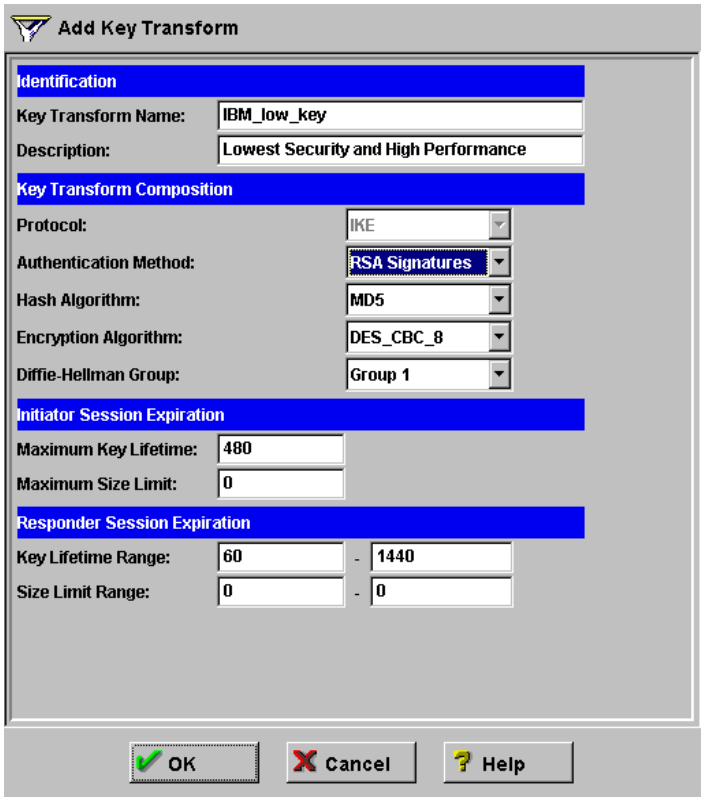


Figure 254. Firewall Technologies key management main window

In the main window double-click **Key Transform** and in the next window double-click **NEW** to add a key transform. You will see Figure 255.


Add Key Transform

Identification

Key Transform Name:	IBM_low_key
Description:	Lowest Security and High Performance

Key Transform Composition

Protocol:	IKE
Authentication Method:	RSA Signatures
Hash Algorithm:	MD5
Encryption Algorithm:	DES_CBC_8
Diffie-Hellman Group:	Group 1

Initiator Session Expiration

Maximum Key Lifetime:	480
Maximum Size Limit:	0

Responder Session Expiration

Key Lifetime Range:	60	1440
Size Limit Range:	0	0

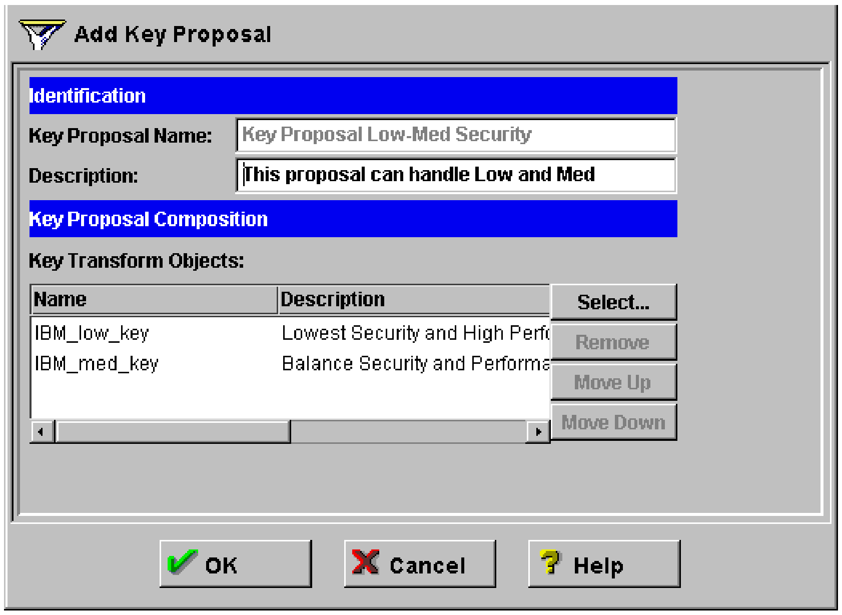
Figure 255. Adding a key transform on OS/390

Fill in the fields based on Table 25 on page 290 and click **OK**. Now click **Close** and return to the main window. This particular example uses the low key level of the policies outlined in Table 25; you now need to repeat the exercise using the values defined in the medium key level to define the second transform object.

Key transform

The key transform object defines the protection mechanisms used to secure subsequent exchanges between key servers. Key transforms specify how to use the Internet Key Exchange (IKE) protocol and include an authentication method and algorithm, a Diffie-Hellman group, and an encryption algorithm.

Having created your key transforms, you now need to define a key proposal. In the main window double-click **Key Proposal** and then **NEW** to add a proposal. You will see Figure 256.


Add Key Proposal

Identification

Key Proposal Name: Key Proposal Low-Med Security

Description: This proposal can handle Low and Med

Key Proposal Composition

Key Transform Objects:

Name	Description
IBM_low_key	Lowest Security and High Perf
IBM_med_key	Balance Security and Performa

Figure 256. Adding a key proposal on OS/390

Click **Select ...** and choose the key transform you decide to include.

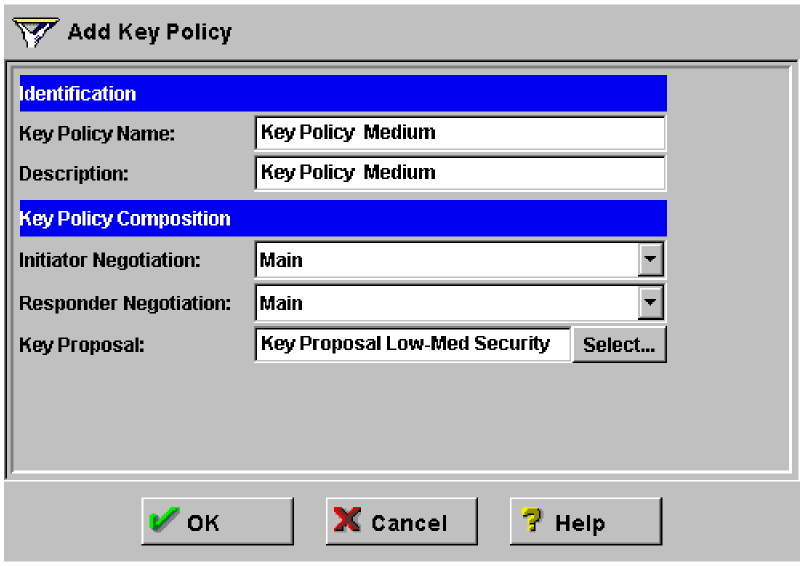
In our example, we include the low and medium transform objects we have just created, butyou can select only a subset of the objects displayed. What you select for this proposal depends on your policy for this particular type of scenario.

Fill in the fields as shown above and click **OK**. Now click **Close** and return to the main window.

Key proposal

The key proposal object contains an ordered list of key transforms that will be proposed during key management negotiation. This ordering is important when acting as an initiator of a dynamic connection. In this case, the key transforms are sent to the remote key server in the initiator's order of preference as defined in the key proposal definition. When acting as a responder, the initiator's ordering takes precedence.

Now, in the main window, double-click **Key Policy**, then double-click **NEW** to add a key policy. You will see the window in Figure 257.



Identification	
Key Policy Name:	Key Policy Medium
Description:	Key Policy Medium
Key Policy Composition	
Initiator Negotiation:	Main
Responder Negotiation:	Main
Key Proposal:	Key Proposal Low-Med Security Select...

Figure 257. Adding Key Policy on OS/390

According to our medium key management policy, we fill in the fields as shown in Table 25 on page 290. Click **Select ...** and choose the key proposal defined previously. Then click **OK.** In the next window click **Close** and return to the main window.

Key policy

The key policy object contains the information required when initiating or responding to a key management security negotiation. The key policy defines this system's initiator and responder negotiation modes and the key proposal.

Now you have completed the key management policy definition.

Data management definitions

In this example, we use the definitions referred to in Table 33 on page 378.

The data management policy is defined using a data policy template. It defines a generic strategy for how IP packets that flow through a VPN are to be protected. Later it will be specified on a dynamic tunnel policy template and could be specified on multiple dynamic tunnel policy templates. Figure 258 highlights the objects to be defined when creating a data management policy.

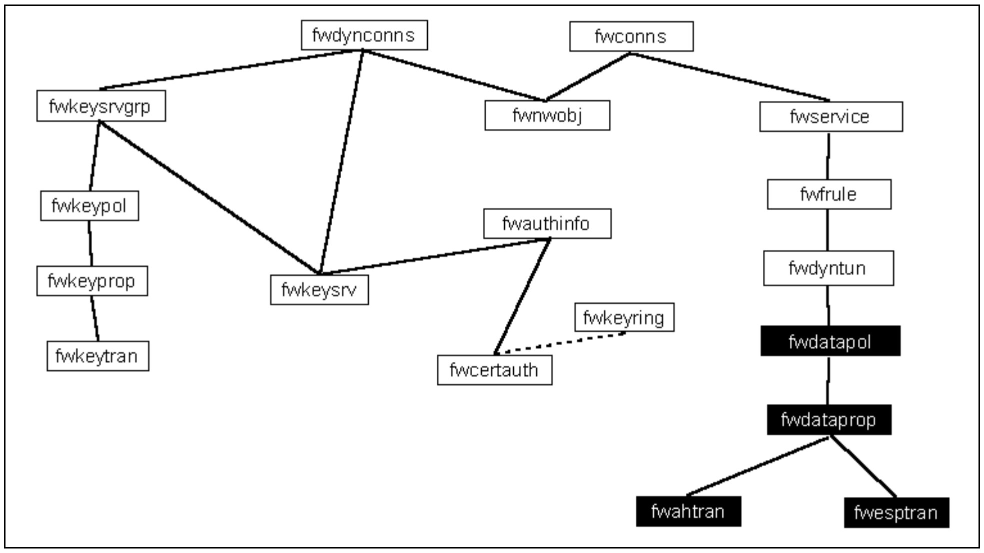


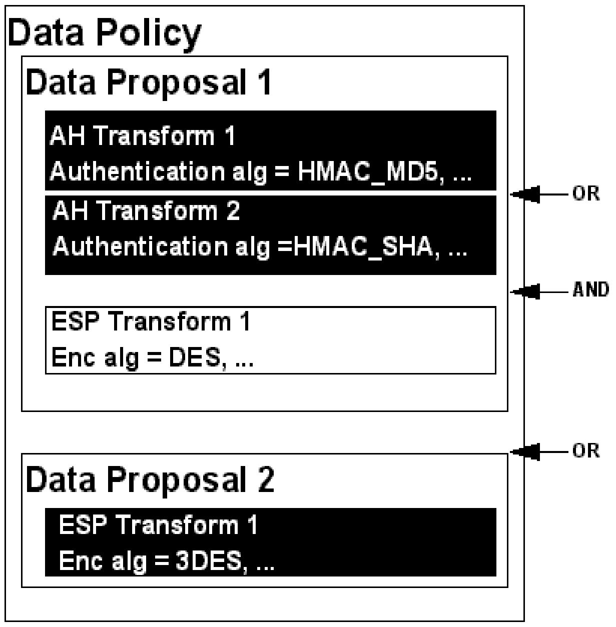
Figure 258. Data management objects

The principles of defining a data management policy are:

- A data policy is comprised of one or more data proposals.
- Data proposals define acceptable security services.
- Data proposals are comprised of AH transforms and ESP transforms.
- AH and ESP transforms identify acceptable values of security service attributes.

The diagram in Figure 259 shows a data policy where it would be acceptable to protect IP traffic in one of the following ways:

- AH with HMAC_MD5 and ESP with DES
- AH with HMAC_SHA and ESP with DES
- ESP with Triple-DES



Note: when a data proposal contains both AH and ESP, the ESP protocol is applied before the AH protocol.

Figure 259. Data policy sample

A data policy is sent by the initiating device to its peer; if multiple proposals are sent, the peer picks one proposal. If the proposal contains multiple transforms for the same security service (for example, multiple AH transforms), the peer picks one. After this, the peer sends back the choices made.

The following tables show examples of data management policies that we used in our scenarios. The four combinations are AH and ESP, transport mode and tunnel mode.

Table 36. Sample data management policies (ESP, tunnel mode)

Data Management Policy	**Low**	**Medium**	**High**
Name	IBM_low_ESP_tunnel	IBM_med_ESP_tunnel	IBM_high_ESP_tunnel
Transform	ESP	ESP	ESP
Encryption Algorithm	DES	DES	Triple-DES
Auth Algorithm	MD5	MD5	SHA
PFS	No	No	No
Encap Mode	Tunnel	Tunnel	Tunnel
SA Lifetime	30 min	20 min	10 min

Table 37. Sample data management policies (ESP, transport mode)

Data Management Policy	**Low**	**Medium**	**High**
Name	IBM_low_ESP_transport	IBM_med_ESP_transport	IBM_high_ESP_transport
Transform	ESP	ESP	ESP
Encryption Algorithm	DES	DES	Triple-DES
Auth Algorithm	MD5	MD5	SHA
PFS	No	No	No
Encap Mode	Transport	Transport	Transport
SA Lifetime	30 min	20 min	10 min

Table 38. Sample data management policies (AH, tunnel mode)

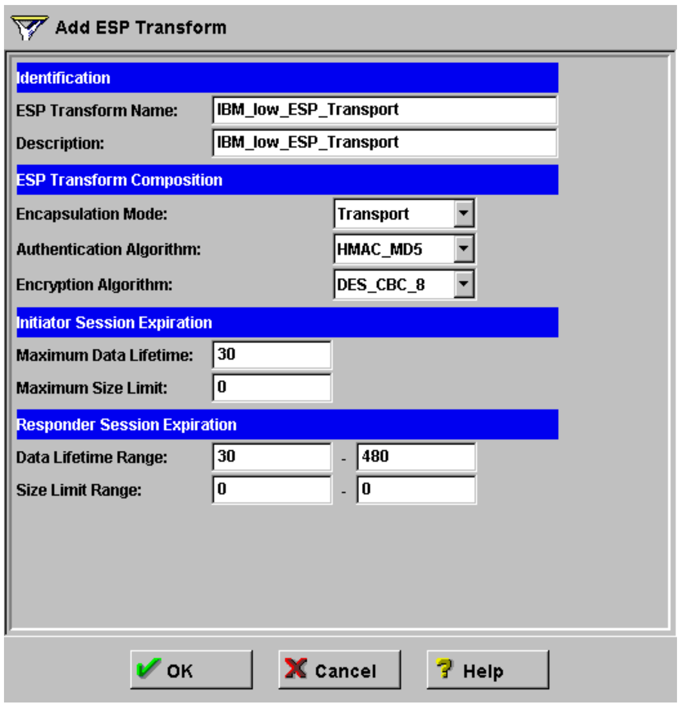
Data Management Policy	**Low**	**Medium**	**High**
Name	IBM_low_AH_tunnel	IBM_med_AH_tunnel	IBM_high_AH_tunnel
Transform	AH	AH	AH
Auth Algorithm	MD5	MD5	SHA
PFS	No	No	No
Encap Mode	Tunnel	Tunnel	Tunnel
SA Lifetime	30 min	20 min	10 min

Table 39. Sample data management policies (AH, transport mode)

Data Management Policy	**Low**	**Medium**	**High**
Name	IBM_low_AH_transport	IBM_med_AH_transport	IBM_high_AH_transport
Transform	AH	AH	AH
Auth Algorithm	MD5	MD5	SHA
PFS	No	No	No
Encap Mode	Transport	Transport	Transport
SA Lifetime	30 min	20 min	10 min

In the process of data management policy definition, we have to define (based on our company policies) AH transform objects or ESP transform objects or both together, then associate them to a data proposal object, and finally create a data policy using the data proposal object.

In the main window, double-click **ESP Transform**, then double-click **NEW** to add an ESP transform. You will see the window below:


Add ESP Transform

Identification

ESP Transform Name:	IBM_low_ESP_Transport
Description:	IBM_low_ESP_Transport

ESP Transform Composition

Encapsulation Mode:	Transport
Authentication Algorithm:	HMAC_MD5
Encryption Algorithm:	DES_CBC_8

Initiator Session Expiration

Maximum Data Lifetime:	30
Maximum Size Limit:	0

Responder Session Expiration

Data Lifetime Range:	30	480
Size Limit Range:	0	0

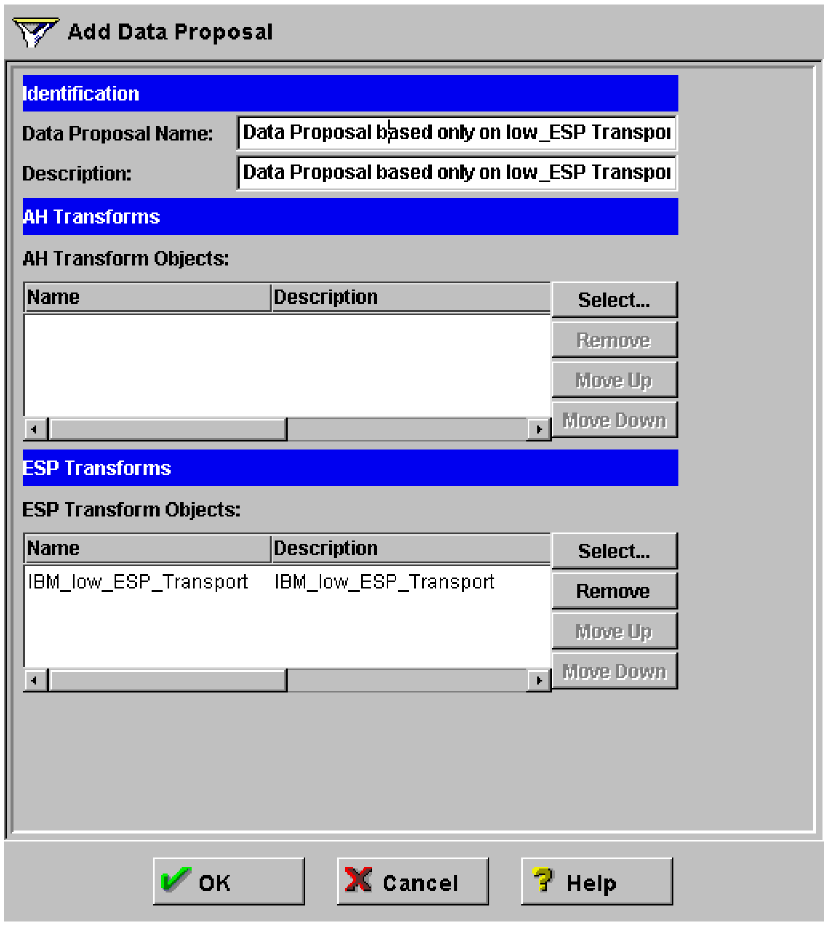
Figure 260. Adding ESP transform on OS/390

Please fill in the fields based on your local policies. In our sample we decided to create an ESP transform object. Click **OK**. In the next window click **Close** and return to the main window.

ESP transform

The ESP transform object defines protection mechanisms used to secure exchanges between the data endpoints through encryption and optionally with authentication.

Now we add the data proposal. In the main window, double-click **Data Proposal**, then double-click **NEW** to add a data proposal. You will see Figure 261:


Add Data Proposal

Identification

Data Proposal Name: Data Proposal based only on low_ESP Transpoı

Description: Data Proposal based only on low_ESP Transpoı

AH Transforms

AH Transform Objects:

Name	Description

ESP Transforms

ESP Transform Objects:

Name	Description
IBM_low_ESP_Transport	IBM_low_ESP_Transport

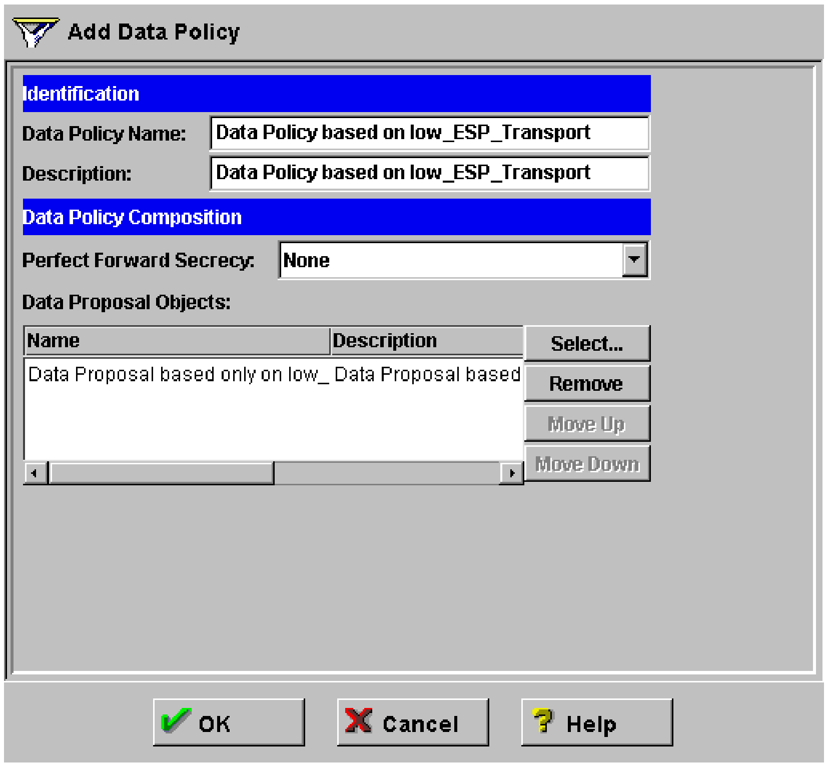
Figure 261. Adding data proposal on OS/390

Please fill in the fields as shown. In the ESP Transforms section click **Select ...** and choose the ESP transform created in Figure 260 on page 388. In this particular case we are not using AH transform objects. Click **OK**. In the next window click **Close** and return to the main window.

Data proposal

The data proposal contains ordered lists of AH and ESP transforms used when negotiating a security association for data transmission. This ordering is important when acting as an initiator of a dynamic connection. The ESP and AH transforms are sent to the remote key server in initiator's order of preference as defined in the data proposal definition. When acting as responder, the initiator's ordering takes precedence.

Now we need to add the data policy. In the main window, double-click **Data Policy**, then double-click **NEW** to add a data policy. You will see Figure 262:



Data Policy Name: Data Policy based on low_ESP_Transport
Description: Data Policy based on low_ESP_Transport
Perfect Forward Secrecy: None
Data Proposal Objects:

Name	Description
Data Proposal based only on low_	Data Proposal based

Figure 262. Adding data policy on OS/390

Click **Select ...** and choose the data proposal created in Figure 261 on page 389. Then click **OK.** In the next window click **Close** and return to the main window.

Data policy

The data policy object defines information required when negotiating keys for data exchanges. This information includes the perfect forward secrecy selection and list of data proposals. The ordering is important when acting as an initiator of a dynamic connection. In this case, the policies are sent to the remote key server in initiator's order of preference as defined in the data policy definition. When acting as a responder, the initiator's ordering takes precedence.

Dynamic tunnel definitions

In this example, we use the definitions in Table 34 on page 378.

Dynamic tunnel policies are defined using a dynamic tunnel template. Such a policy defines a generic strategy for how a dynamic tunnel is to be managed. The dynamic tunnel policy is closely related to the data management policy and has to be specified on an anchor rule. A dynamic tunnel policy could be specified on multiple anchor rules. Take a look at Figure 263 where the relevant objects have been highlighted.

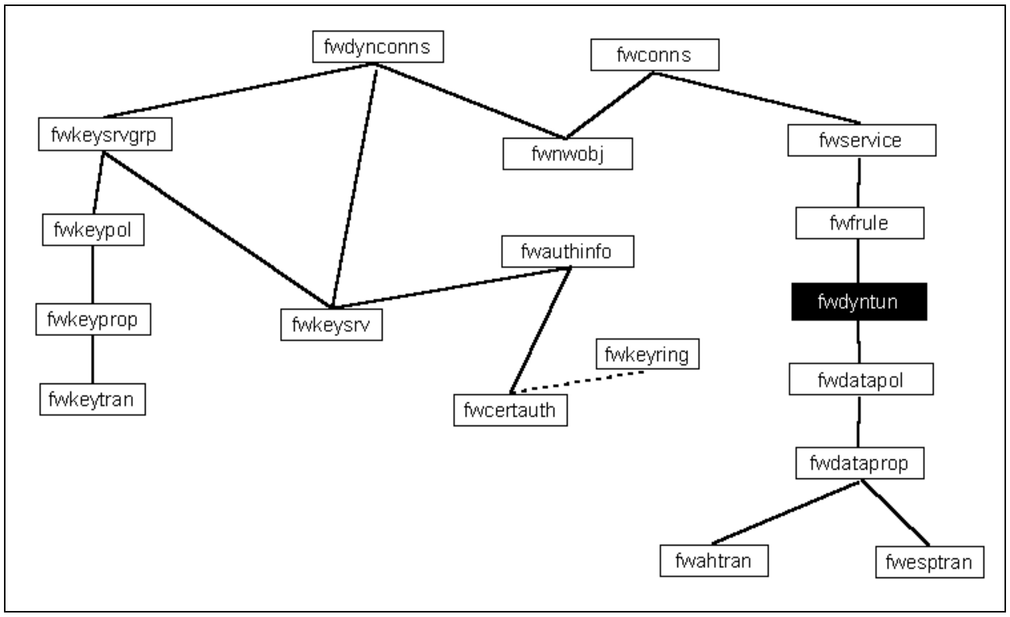


Figure 263. Dynamic tunnel objects

It is important in this context to understand the difference between a dynamic *tunnel* and a dynamic *connection*.

A dynamic connection is the communication circuit between the source and destination of data sent through a VPN.

A dynamic tunnel is the portion of a dynamic connection that is protected by IPSec. It is established between two IPSec devices running IKE. Please refer to Figure 264:

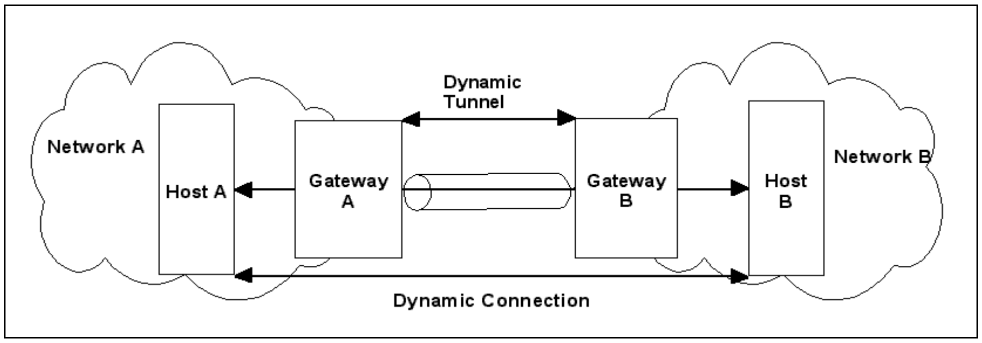
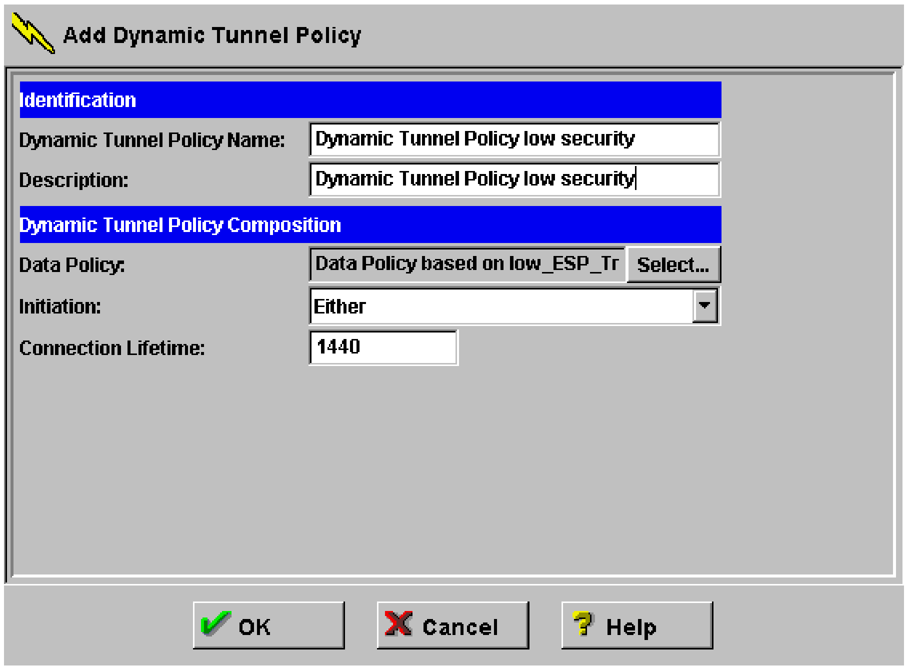


Figure 264. Dynamic tunnel versus dynamic connection

The dynamic tunnel policy object will be associated with an anchor filter rule object. To define the dynamic tunnel policy object, proceed as follows. In the main window, double-click **Dynamic Tunnel Policy**, then double-click **NEW** to add a dynamic tunnel policy. You will see the window in Figure 265 on page 392.


Add Dynamic Tunnel Policy

Identification

Dynamic Tunnel Policy Name: Dynamic Tunnel Policy low security

Description: Dynamic Tunnel Policy low security

Dynamic Tunnel Policy Composition

Data Policy: Data Policy based on low_ESP_Tr

Initiation: Either

Connection Lifetime: 1440

Figure 265. Adding dynamic tunnel policy on OS/390

Choose your initiation option. Click **Select ...** and choose the data policy created in Figure 262 on page 390. Then click **OK**. In the next window click **Close** and return to the main window.

Dynamic tunnel policy

The dynamic tunnel policy object defines generic information relative to a set of tunnels. This information includes the data policy object, Initiation role, and connection lifetime to use. Dynamic tunnel policy objects will be specified in filter rule objects.

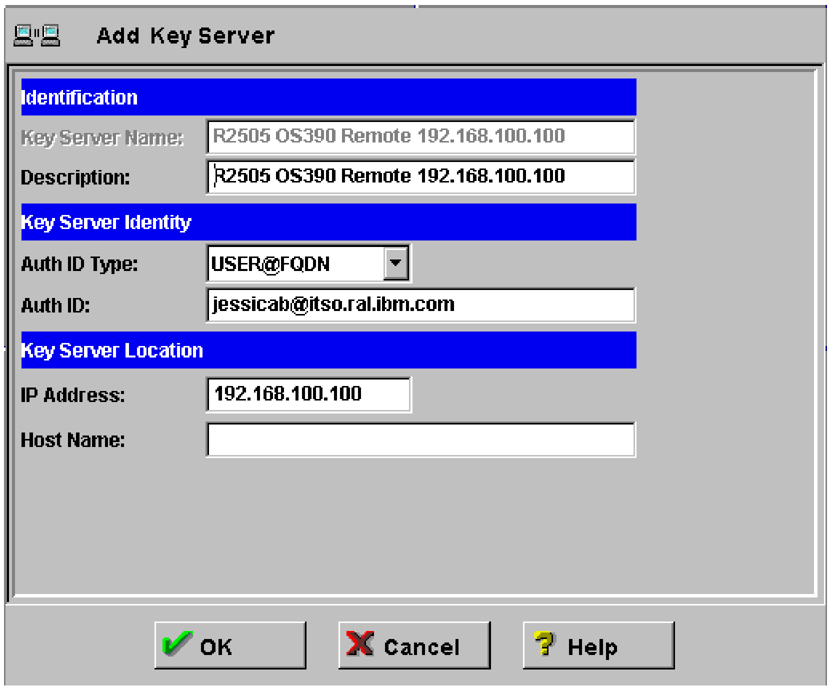
At this point, we have shown how to create and manage your key management, data management and dynamic tunnel policies. You will use these concepts in order to implement your VPN solutions.

Key Server definitions

Now we will define the key server objects. We have to define two such objects: the local key server (OS/390 RA28) and the remote key server (OS/390 RA03). Then we have to define a key server group using the key server definitions. Please refer to Table 35 on page 379.

During the definition of each key server, we have to select the way the server will be authenticated. This parameter (Auth ID Type) depends on the policy of each of the OS/390 systems with which we are communicating.

We invoke the Firewall Technologies configuration client from the main firewall window. See Figure 173 on page 282. In the main window, double-click **Key Server**, then double-click **NEW** to add a key server. You will see Figure 266 on page 393. We then complete the necessary fields for our remote key server. Here the remote server has the ID type USER@FQDN (a user’s address in traditional e-mail format). Again, double-click **NEW** to add the second key server. Fill in the fields as shown in Figure 267 for the local key server definition. This second server uses IPv4 (a four-byte traditional dotted IP address) as the authentication ID type. Click **OK** to finish your definition. In the next window, click **Close** and return to the main window.



Identification

Key Server Name:	R2505 OS390 Remote 192.168.100.100
Description:	R2505 OS390 Remote 192.168.100.100

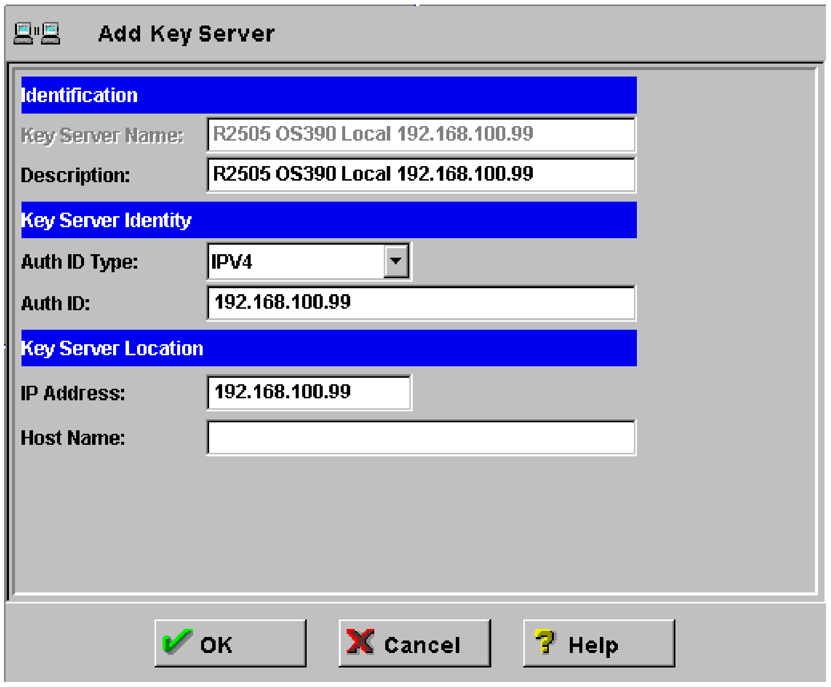
Key Server Identity

Auth ID Type:	USER@FQDN
Auth ID:	jessicab@itso.ral.ibm.com

Key Server Location

IP Address:	192.168.100.100
Host Name:	

Figure 266. Adding remote key server on OS/390



Identification

Key Server Name:	R2505 OS390 Local 192.168.100.99
Description:	R2505 OS390 Local 192.168.100.99

Key Server Identity

Auth ID Type:	IPV4
Auth ID:	192.168.100.99

Key Server Location

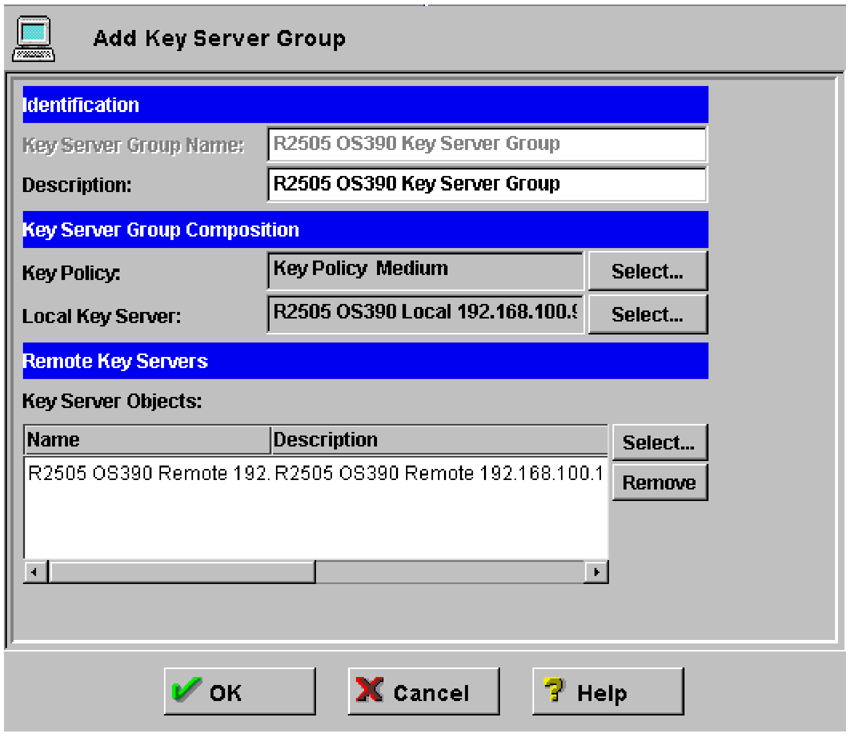
IP Address:	192.168.100.99
Host Name:	

Figure 267. Adding local key server on OS/390

Key server

The key server object defines information about key servers. Key servers negotiate security associations using the Internet Key Exchange Protocol. A key server object defines an authentication identity by which a key server is known. A key server may have multiple identities.

The next step is the creation of the server group definition. In the main window, double-click **Key Server Group**, then double-click **NEW** to add a key server group. You will see the window shown in Figure 268:


Add Key Server Group

Identification

Key Server Group Name: R2505 OS390 Key Server Group

Description: R2505 OS390 Key Server Group

Key Server Group Composition

Key Policy: Key Policy Medium

Local Key Server: R2505 OS390 Local 192.168.100.!

Remote Key Servers

Key Server Objects:

Name	Description
R2505 OS390 Remote 192.	R2505 OS390 Remote 192.168.100.1

Figure 268. Adding Key Server Group on OS/390

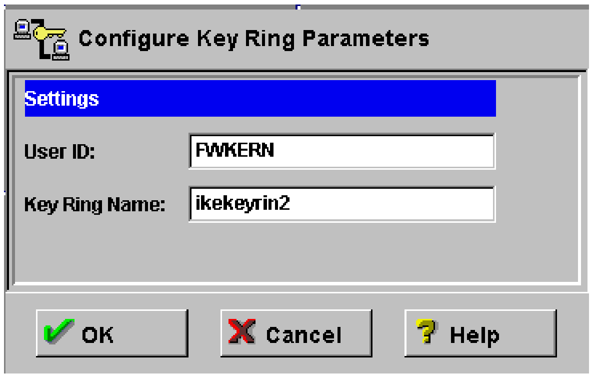
Click **Select ...** beside Key Policy and choose the key policy created in Figure 257 on page 384. Click **Select ...** beside Local Key Server and choose the local key server created in Figure 267 on page 393. Click **Select ...** under Remote Key Servers and choose the remote key server created in Figure 266 on page 393. Then click **OK**. In the next window click **Close** and return to the main window.

Key server group

The key server group object defines information about key server groups. A key server group is an association of a local key server with a list of remote key servers. This establishes which combinations of local and remote key servers can negotiate key management security associations and the key policy that will be used during these negotiations. Key server groups are ordered among themselves. Searches will find the first instance of the key server found in all key server group objects. Therefore, the key server groups must be ordered. Key server group objects will be specified in dynamic VPN connection objects.

E.5.1.3 Authentication information definition

Next, we define the authentication information. In the main window, double-click the **Authentication Information** tree to expand it. Double-click the option **Key Ring** to see the keyring configuration window as shown in Figure 269:



User ID: FWKERN

Key Ring Name: ikekeyrin2

Figure 269. Checking keyring parameters on OS/390

Figure 269 shows two fields with useful information: one shows the user ID that is associated with the ring definition, and the second one shows the name of the keyring.

By default these objects are *not* associated with the RACF database, so you have to create the ring object and then associate it to the firewall user ID with the `RACDCERT` command.

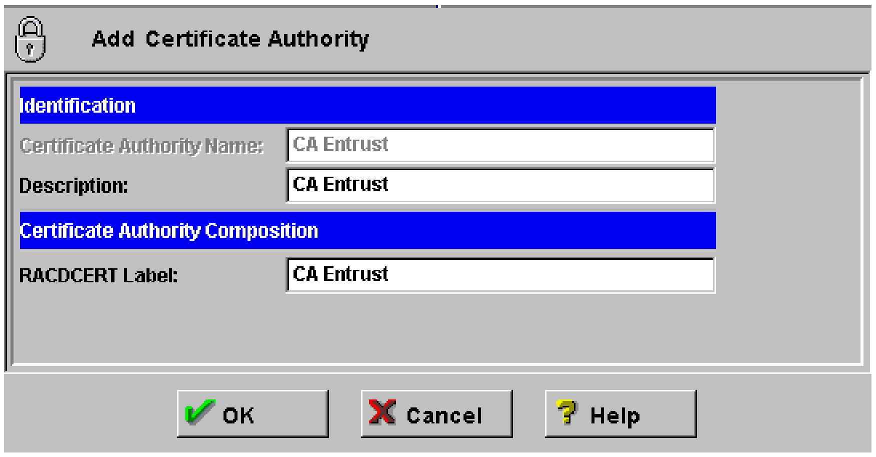
By default the name of the keyring in this window is ikekeyring; we have changed it to ikekeyrin2.

We are working in RSA mode, as defined in our planning sheets, so we need to have some Certification Authority that validates our identity. In this case, the CA Entrust has been added (using `RACDCERT`) to our RACF database. Here, in this step, we refer to this already created RACF definition.

In RSA mode, you must have definitions for all the Certificate Authorities that validate your peers.

For more details on certificate management, please refer to Appendix D, “OS/390 certificate management using RACF” on page 321.

From the Firewall Technologies main window, double-click **Certificate Authority** in order to define the definitions you need for the CA that validates you and your peer. In our configuration we use the CA Entrust:



Add Certificate Authority

Identification

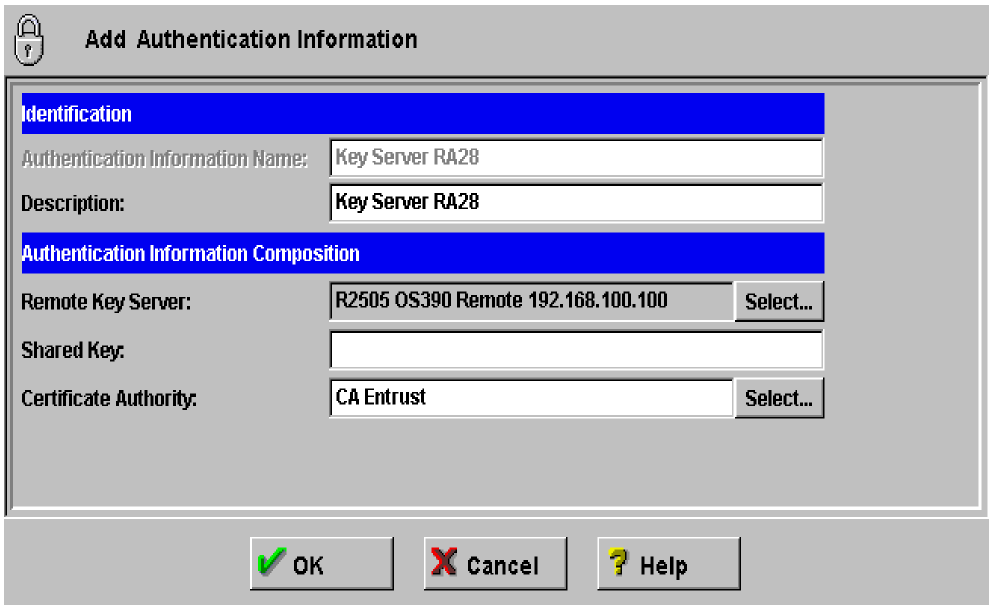
Certificate Authority Name:	CA Entrust
Description:	CA Entrust

Certificate Authority Composition

RACDCERT Label:	CA Entrust

Figure 270. Adding Certificate Authority

In the main window, double-click **Authentication Information**, then double-click **NEW** to add authentication information. You will see the window below. Click **Select ...** and choose the remote key server as defined in Figure 266 on page 393. Then click **OK.** In the next window click **Close** and return to the main window.



Add Authentication Information

Identification

Authentication Information Name:	Key Server RA28
Description:	Key Server RA28

Authentication Information Composition

Remote Key Server:	R2505 OS390 Remote 192.168.100.100
Shared Key:	
Certificate Authority:	CA Entrust

Figure 271. Adding authentication information

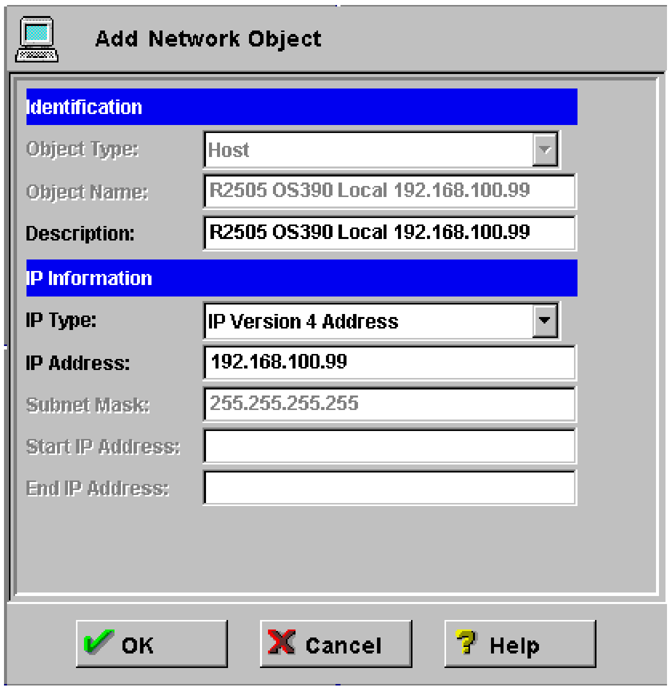
Authentication Information

The authentication information object defines information required to authenticate the identity of the communicating peer. This information includes the remote key server object and the pre-shared key and/or certificate authority to use with it. Authentication information will be used when a dynamic connection is activated.

Dynamic VPN connection definition

Now we define the Dynamic VPN connection (as opposed to the dynamic VPN tunnel). We associate the network objects from both endpoints of the tunnel with the remote key server and key server group.

First, we have to create two Network objects identifying the OS/390 RA28 and OS/390 RA03 systems. In the main window, double-click **Network Objects**, then double-click **NEW** to add a network object. You will see the window in Figure 272 where you fill in the fields as shown. Then click **OK**,


Add Network Object

Identification

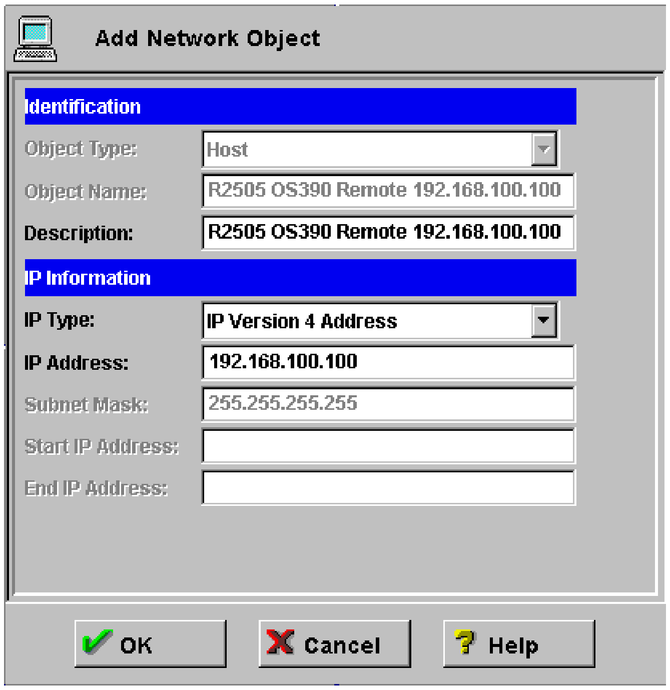
Object Type:	Host
Object Name:	R2505 OS390 Local 192.168.100.99
Description:	R2505 OS390 Local 192.168.100.99

IP Information

IP Type:	IP Version 4 Address
IP Address:	192.168.100.99
Subnet Mask:	255.255.255.255
Start IP Address:	
End IP Address:	

Figure 272. Adding Network Object for OS/390 RA28

Double-click **NEW** again to add a network object. You will see the window in Figure 273 on page 398.



Identification

Object Type:	Host
Object Name:	R2505 OS390 Remote 192.168.100.100
Description:	R2505 OS390 Remote 192.168.100.100

IP Information

IP Type:	IP Version 4 Address
IP Address:	192.168.100.100
Subnet Mask:	255.255.255.255
Start IP Address:	
End IP Address:	

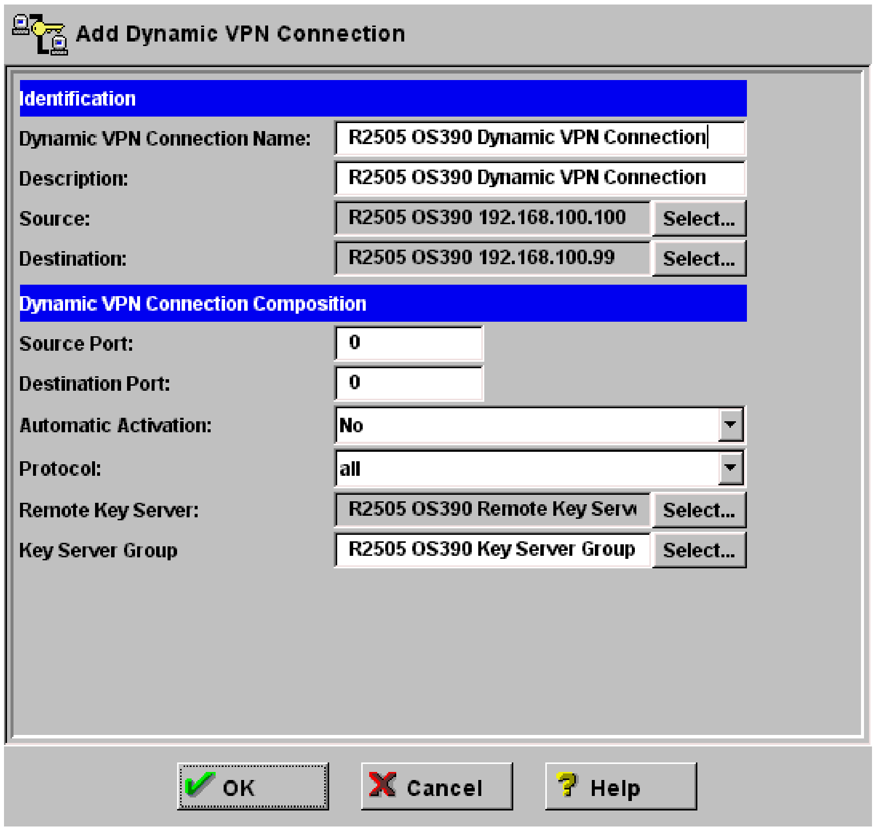
Figure 273. Adding Network Object for OS/390 RA03

Fill in the fields as shown, click **OK**, and in the next window click **Close** and return to the main window.

Network object

The network objects function allows you to maintain information about network addressable components on your network. This function acts as a central repository for use by other functions in the OS/390 system. Primarily, network objects are used to designate source and destination addresses when you create your connections.

Having created the network objects, we create the dynamic VPN connection itself. In the main window, double-click **VPN Connection Setup**, then double-click **NEW** to add a dynamic VPN connection. You will see Figure 274.


Add Dynamic VPN Connection

Identification

Dynamic VPN Connection Name:	R2505 OS390 Dynamic VPN Connection
Description:	R2505 OS390 Dynamic VPN Connection
Source:	R2505 OS390 192.168.100.100
Destination:	R2505 OS390 192.168.100.99

Dynamic VPN Connection Composition

Source Port:	0
Destination Port:	0
Automatic Activation:	No
Protocol:	all
Remote Key Server:	R2505 OS390 Remote Key Serv
Key Server Group	R2505 OS390 Key Server Group

Figure 274. Adding dynamic VPN connection on OS/390

Fill in the fields as shown in Figure 274. Click **Select ...** beside Source and choose the OS/390 network object created in Figure 272 on page 397. Click **Select ...** beside Destination and choose the RA03 network object created in Figure 273 on page 398. Click **Select ...** beside Remote Key Server and choose the remote key server created in Figure 266 on page 393. Click **Select ...** beside Key Server Group and choose the key server group created in Figure 268 on page 394. Then click **OK**. In the next window click **Close** and return to the main window.

Dynamic VPN connection

The dynamic VPN connection object defines information that is used to activate a specific connection between data endpoints. This information includes the source and destination objects, source and destination ports, and protocol supported for this connection along with the remote key server and key server group objects and an indicator of whether to auto-activate the connection.

Creating the anchor filter rules, services and connections

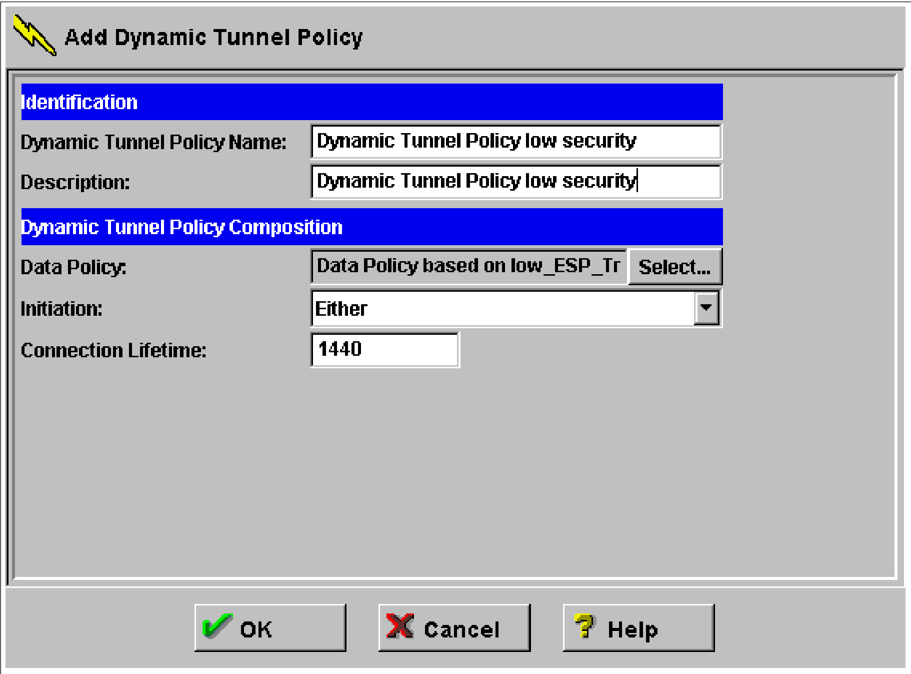
Now we have to create the anchor filter rules, the services that will allow the tunnel endpoints to use AH and ESP protocol, and the normal services and connections to create the filter rules and the dynamic filter rules.

In this example we define an anchor filter rule in which we will allow all types of traffic to flow in the tunnel. We will also associate this anchor filter rule to a dynamic tunnel policy.

Rules for dynamic VPN

Rules on the OS/390 system are used to screen traffic passing through the system. Rules can be set up to either allow or disallow traffic on the basis of certain criteria. When a dynamic VPN connection is activated, the anchor rule is used to determine the placement of the dynamically generated filter rules among the static permit and deny rules.

First, create a dynamic tunnel policy as shown in Figure 275.


Add Dynamic Tunnel Policy

Identification

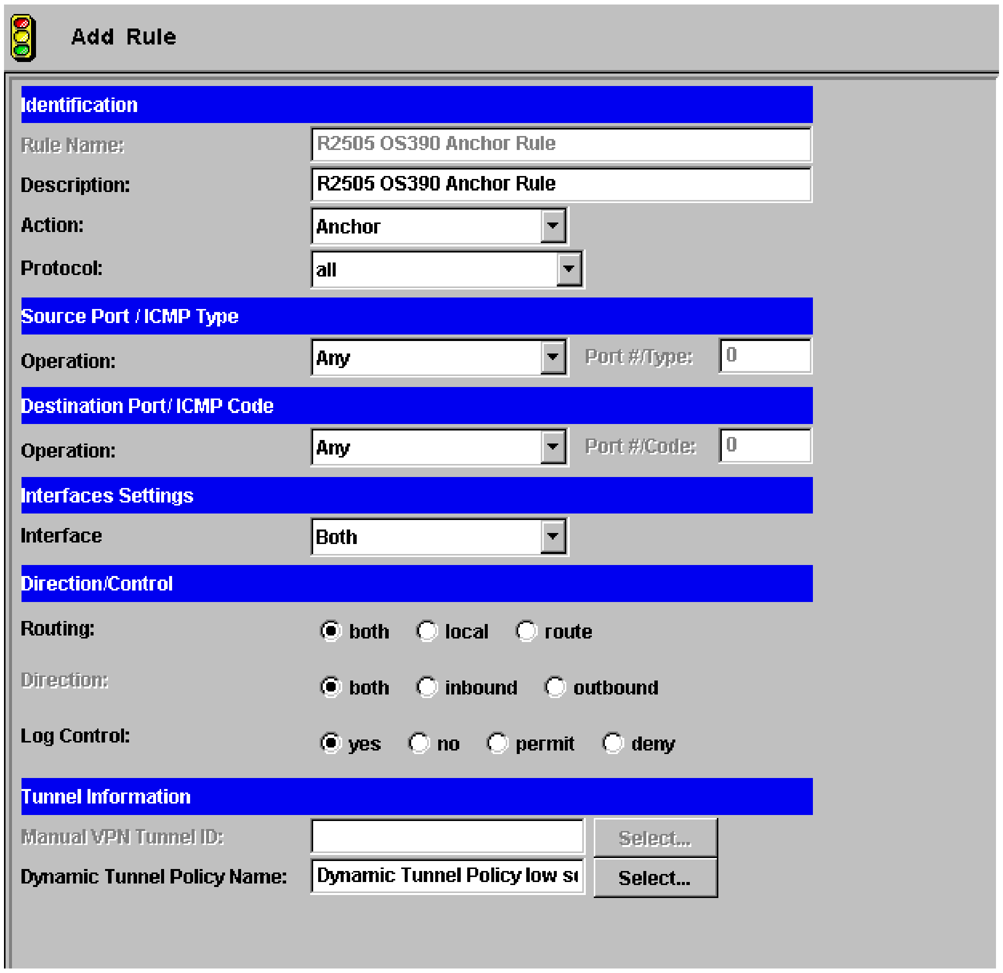
Dynamic Tunnel Policy Name:	Dynamic Tunnel Policy low security
Description:	Dynamic Tunnel Policy low security

Dynamic Tunnel Policy Composition

Data Policy:	Data Policy based on low_ESP_Tr
Initiation:	Either
Connection Lifetime:	1440

Figure 275. Adding a dynamic tunnel policy

In the main window, double-click **Connection Templates** and **Rules**, then double-click **NEW** to add a new rule. You will see Figure 276.



Identification

Rule Name:	R2505 OS390 Anchor Rule
Description:	R2505 OS390 Anchor Rule
Action:	Anchor
Protocol:	all

Source Port / ICMP Type

Operation:	Any
Port #/Type:	0

Destination Port/ ICMP Code

Operation:	Any
Port #/Code:	0

Interfaces Settings

Interface	Both

Direction/Control

Routing:	[✓ both] local route
Direction:	[✓ both] inbound outbound
Log Control:	[✓ yes] no permit deny

Tunnel Information

Manual VPN Tunnel ID:	
Dynamic Tunnel Policy Name:	Dynamic Tunnel Policy low s(

Figure 276. Creating Anchor Rule on OS/390

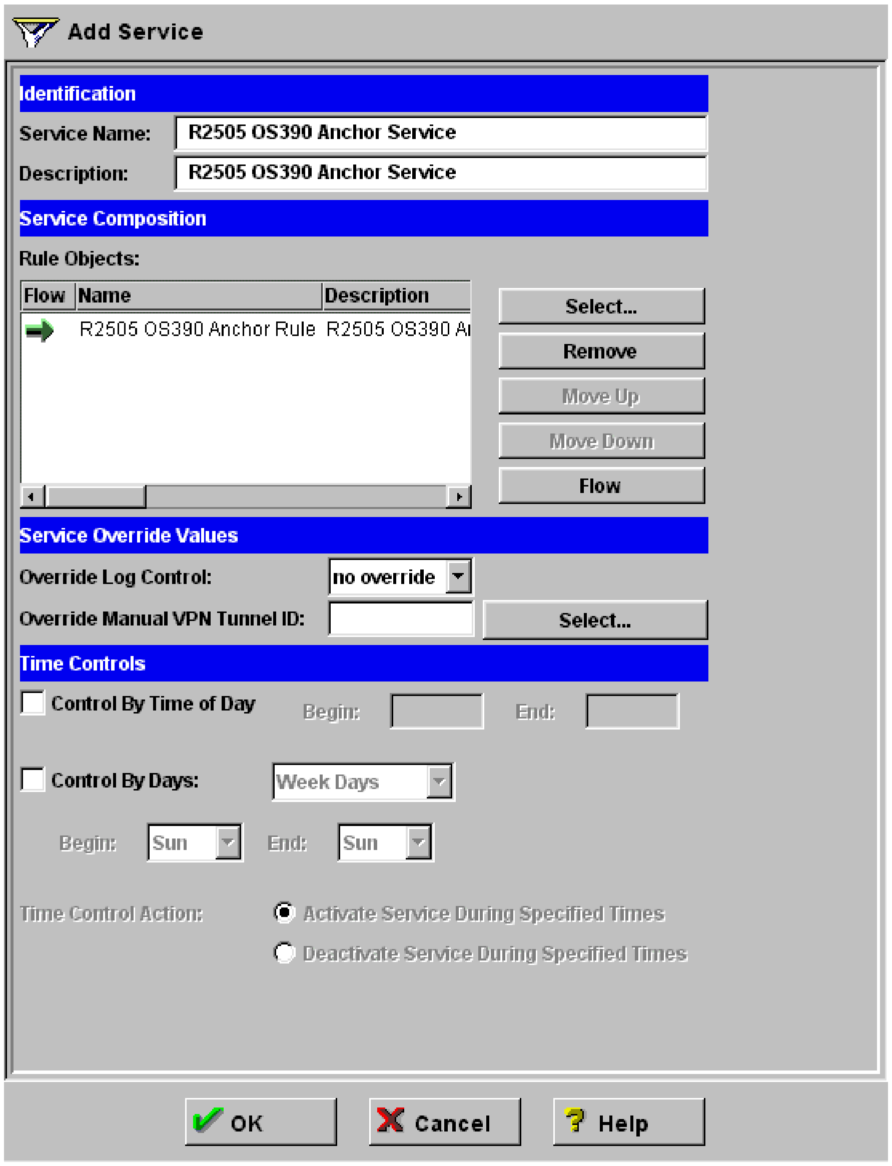
Fill in the fields as shown. Click **Select ...** beside Dynamic Tunnel Policy Name and choose the dynamic tunnel policy created in Figure 275 on page 400. Then click **OK**. In the next window click **Close** and return to the main window.

Anchor filter rule

Rules on the OS/390 system are used to screen traffic passing through the system. Rules can be set up to either allow or disallow traffic on the basis of certain criteria. When a dynamic VPN connection is activated, the anchor rule is used to determine the placement of the dynamically generated filter rules among the static permit and deny rules.

Now we have to create two services: one service establishes the anchor filter rule between the connection endpoints and the other establishes the permit filter rules for the key server traffic.

In the main window, double-click **Services**, then double-click **NEW** to add a new service. You will see the window in Figure 277:


Add Service

Identification

Service Name: R2505 OS390 Anchor Service

Description: R2505 OS390 Anchor Service

Service Composition

Rule Objects:

Flow	Name	Description
➡	R2505 OS390 Anchor Rule	R2505 OS390 A

Service Override Values

Override Log Control: no override

Override Manual VPN Tunnel ID:

Time Controls

[ ] Control By Time of Day Begin: End:

[ ] Control By Days: Week Days

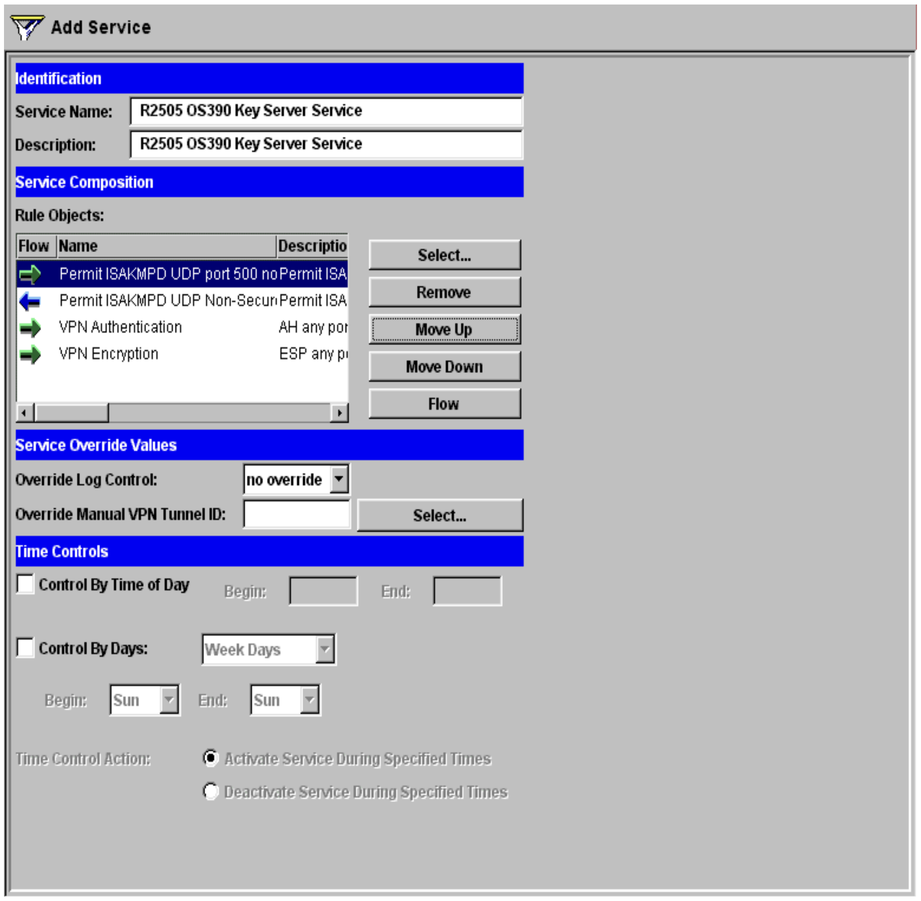
Begin: Sun End: Sun

Time Control Action: [● Activate Service During Specified Times] ○ Deactivate Service During Specified Times

Figure 277. Adding Anchor Service on OS/390

After filling in the fields, click **Select ...** under Rule Objects and choose the anchor filter rule created in Figure 276 on page 401. Then click **OK**.

Now double-click **NEW** to add the other (key server) service. You will see the window in Figure 278.



Service Name: R2505 OS390 Key Server Service
Description: R2505 OS390 Key Server Service

Flow	Name	Descriptio
➡	Permit ISAKMPD UDP port 500 no	Permit ISA
⬅	Permit ISAKMPD UDP Non-Secur	Permit ISA
➡	VPN Authentication	AH any por
➡	VPN Encryption	ESP any p

Override Log Control: no override
Override Manual VPN Tunnel ID:

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 278. Adding key server service on OS/390

Click **Select ...** under rule objects and choose the following filter rules:

- ISAKMPD UDP Non-Secure traffic in both directions
- VPN Authentication traffic
- VPN Encryption object

After choosing these rule objects, click **OK.** In the next window click **Close** and return to the main window.

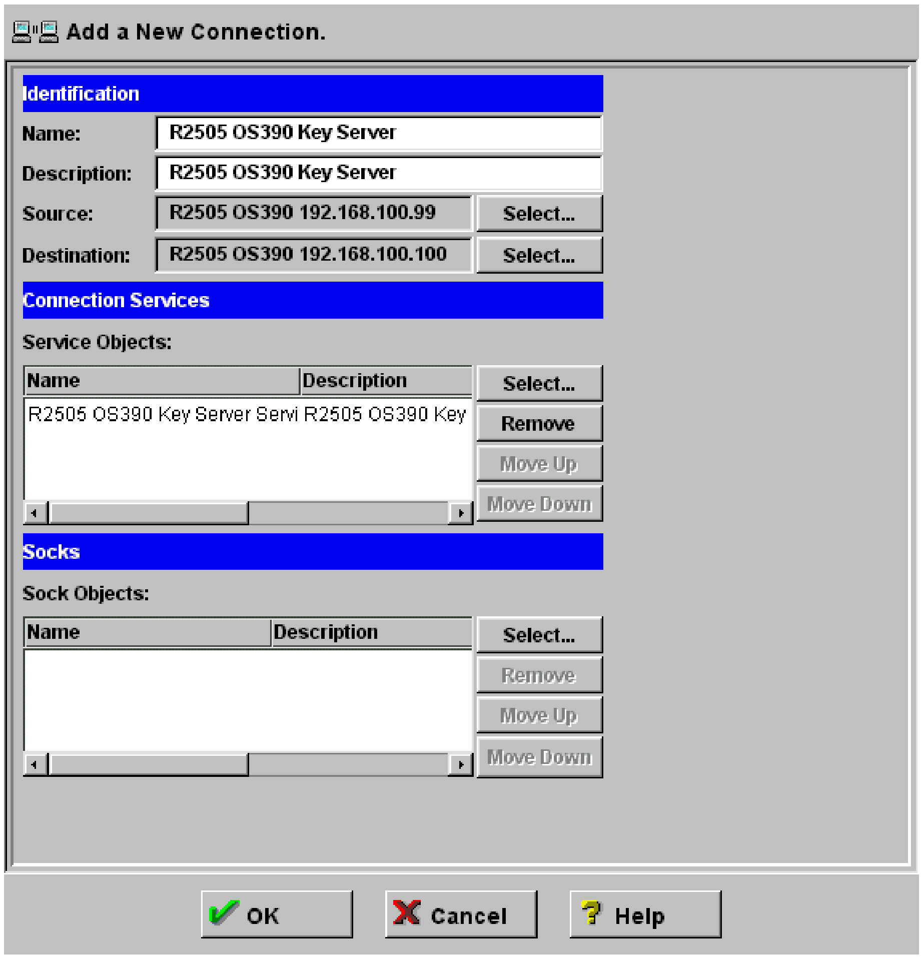
Note that the definitions selected here are predefined rule objects.

Services

Services are collections of rules or sets of instructions to permit or deny a particular type of traffic through the OS/390 system, for example, a Telnet session. Services figure prominently when defining connections, specifying the type of traffic that can or cannot take place between network objects.

Now we create the two connections to associate the service between the tunnel endpoints. Then we will activate the connection to create the filter rules.

In the main window, double-click **Connection Setup**, then double-click **NEW** to add a new connection. You will see Figure 279.


Add a New Connection.

Identification

Name:	R2505 OS390 Key Server
Description:	R2505 OS390 Key Server
Source:	R2505 OS390 192.168.100.99
Destination:	R2505 OS390 192.168.100.100

Connection Services

Service Objects:

Name	Description
R2505 OS390 Key Server Servi	R2505 OS390 Key

Socks

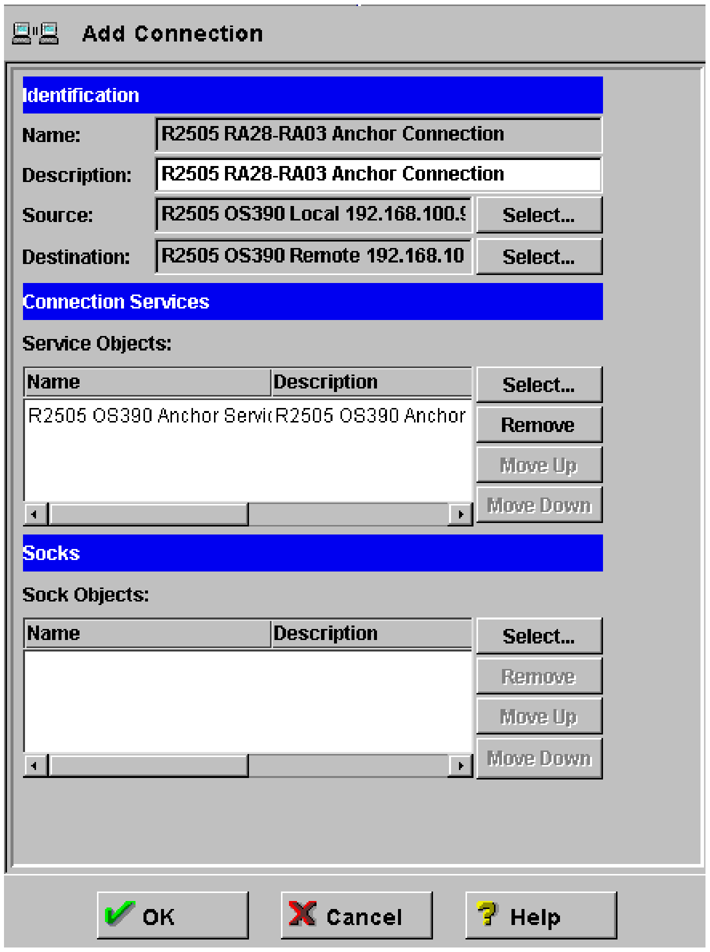
Sock Objects:

Name	Description

Figure 279. Adding Key Server Connection on OS/390

Fill in the fields as shown. Click **Select ...** beside Source and choose the network object created in Figure 272 on page 397. Click **Select ...** beside Destination and choose the network object created in Figure 273 on page 398. In addition, add to the Service Objects the key server service you created in Figure 278 on page 403. Then click **OK**.

Again, double-click **NEW** to add a new connection and fill in the fields as shown in Figure 280.



Identification

Name: R2505 RA28-RA03 Anchor Connection

Description: R2505 RA28-RA03 Anchor Connection

Source: R2505 OS390 Local 192.168.100.!

Destination: R2505 OS390 Remote 192.168.10

Connection Services

Service Objects:

Name	Description
R2505 OS390 Anchor Servi	R2505 OS390 Anchor

Socks

Sock Objects:

Name	Description

Figure 280. Adding an anchor connection on OS/390

Click **Select ...** beside Source and choose the network object created in Figure 272 on page 397. Click **Select ...** beside Destination and choose the network object created in Figure 273 on page 398. In addition, add to the Service Objects the anchor service definition you created in Figure 277 on page 402. Then click **OK**.

Connections

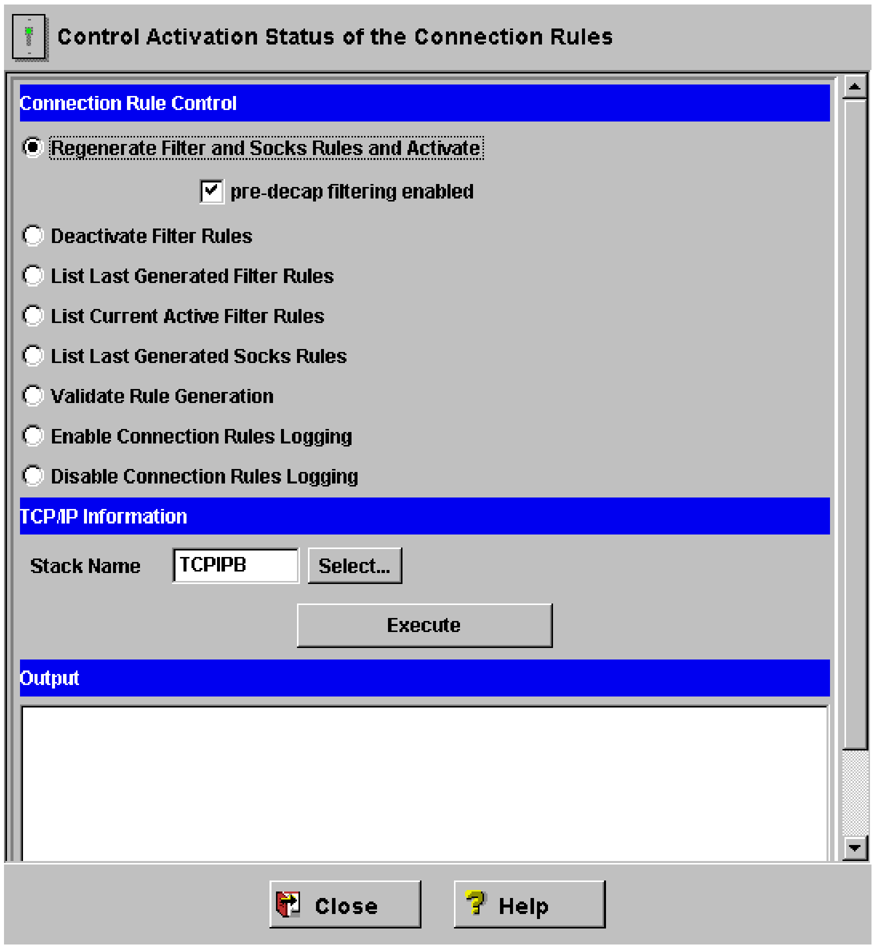
The connection function allows you to control the type of network traffic that can take place between two network entities that are connected through the OS/390 system. Connection definitions permit or deny specified types of communications between two named endpoints or any type of network object or group. After you have defined your network objects and services, you create connections. In building connections, you will select one network object to be the source and another network object to be the destination for the traffic flow through the OS/390 system.

Note: The connections must be ordered (sequenced) in the following order: key server connection, followed by anchor connection. Check this in the Connection Setup window and reorder the connections if necessary. Access Help from the Connection Setup window for more information.

Activating the filter rules and the dynamic VPN connection

Now we will activate the filter rules and the dynamic VPN connection.

In the main window, double-click **Connection Activation**, then mark the boxes shown in the window below and click **Execute**. Check the messages in the output area for errors. Then click **Close**.



Control Activation Status of the Connection Rules

Connection Rule Control

(●) Regenerate Filter and Socks Rules and Activate

[✓] pre-decap filtering enabled

( ) Deactivate Filter Rules

( ) List Last Generated Filter Rules

( ) List Current Active Filter Rules

( ) List Last Generated Socks Rules

( ) Validate Rule Generation

( ) Enable Connection Rules Logging

( ) Disable Connection Rules Logging

TCP/IP Information

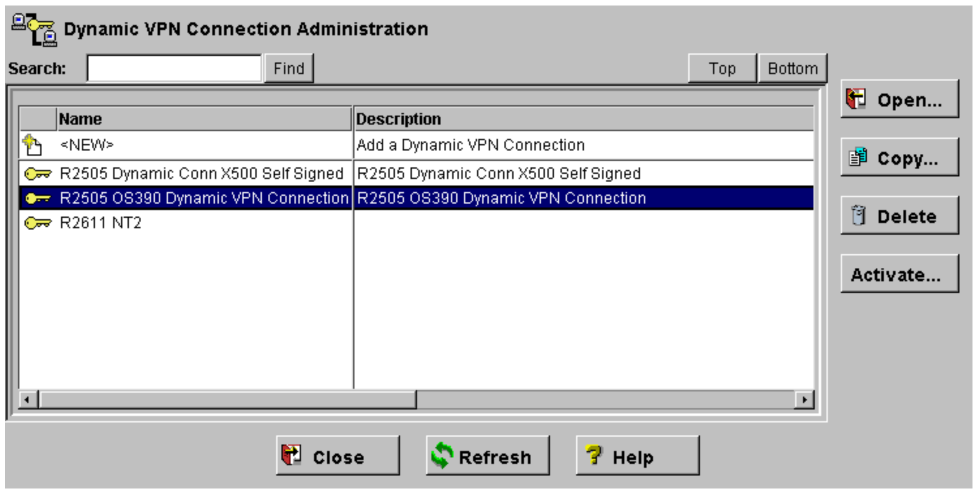
Stack Name: TCPIPB

Figure 281. Connection activation

Connection activation

Use this function to generate the rules based upon the configurations defined in the Connection Setup window and all of its subsidiary configurations (for example, services, rule templates, and SOCKS templates). Rules can also be generated depending upon settings in the Security Policy window. These rules become the active set through which the OS/390 system can evaluate network datagrams. If there is a set of connection rules already active, this procedure updates the active rules with the contents of the newly generated set. Feedback about a successful activation or any errors will be displayed in the output section. **Note:** Checking the **pre-decap filtering enabled** check box results in AH and ESP packets being filtered before they are decapsulated. If there are concerns about AH and ESP packets in your network, then you may want to have the AH and ESP packets filtered before they are decapsulated.

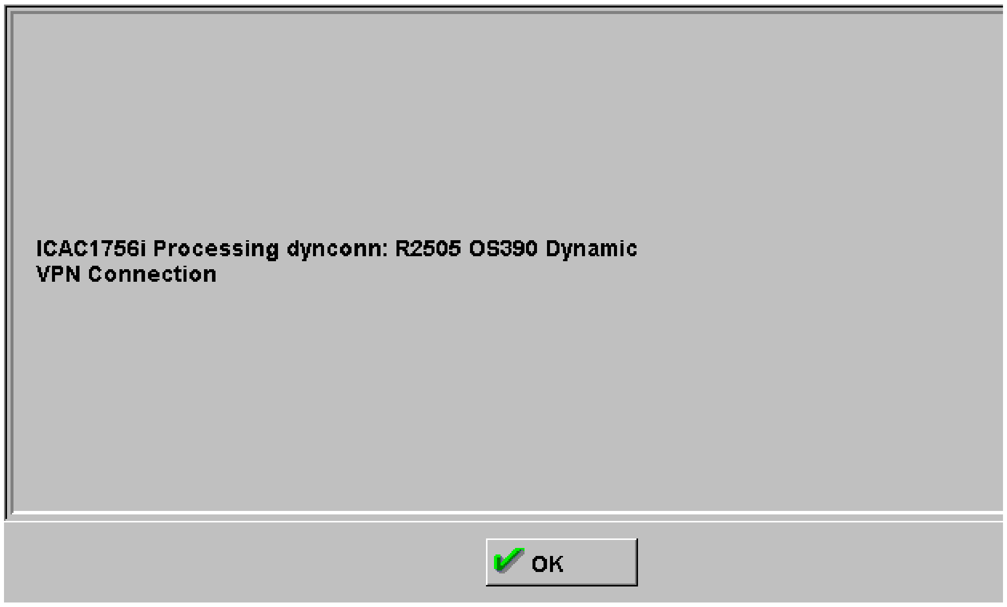
Now we will activate the dynamic VPN connection. In the main window, double-click **VPN Connection Setup**, then choose the dynamic VPN connection created in Figure 274 on page 399 and click **Activate** (see Figure 282).



Name	Description
<NEW>	Add a Dynamic VPN Connection
R2505 Dynamic Conn X500 Self Signed	R2505 Dynamic Conn X500 Self Signed
R2505 OS390 Dynamic VPN Connection	R2505 OS390 Dynamic VPN Connection
R2611 NT2	

Figure 282. Dynamic VPN connection activation

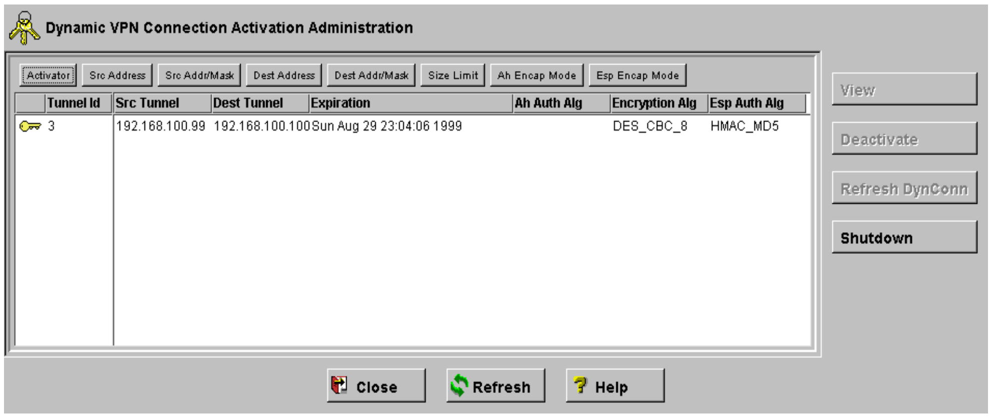
You will receive the following message:



ICAC1756i Processing dynconn: R2505 OS390 Dynamic VPN Connection

Figure 283. Dynamic connection activation message

This message only tells you that the activation of the dynamic VPN connection is in progress. You have to check the VPN Connection Activation window to see if the tunnel was activated. To check all the parameters in the tunnel, you can double-click the tunnel, or select the tunnel and click **View** (Figure 284).

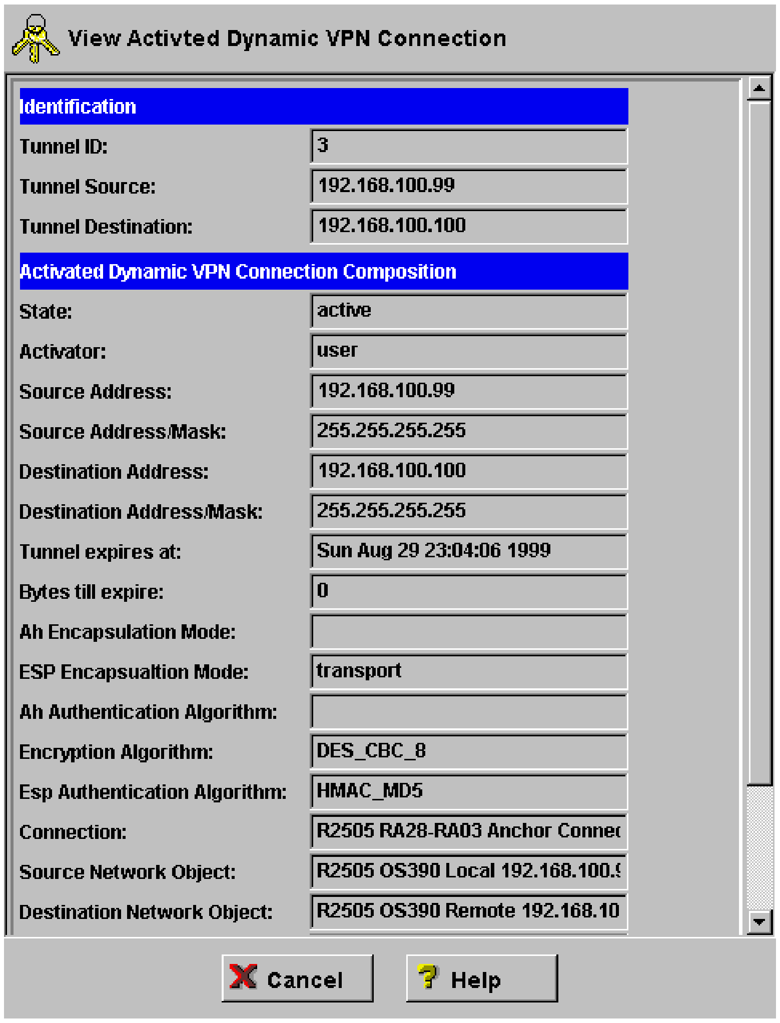


Dynamic VPN Connection Activation Administration

Tunnel Id	Src Tunnel	Dest Tunnel	Expiration	Ah Auth Alg	Encryption Alg	Esp Auth Alg
3	192.168.100.99	192.168.100.100	Sun Aug 29 23:04:06 1999		DES_CBC_8	HMAC_MD5

Figure 284. Dynamic connection activation list

You will receive the following window showing all the definitions:


View Activted Dynamic VPN Connection

Identification	
Tunnel ID:	3
Tunnel Source:	192.168.100.99
Tunnel Destination:	192.168.100.100
Activated Dynamic VPN Connection Composition	
State:	active
Activator:	user
Source Address:	192.168.100.99
Source Address/Mask:	255.255.255.255
Destination Address:	192.168.100.100
Destination Address/Mask:	255.255.255.255
Tunnel expires at:	Sun Aug 29 23:04:06 1999
Bytes till expire:	0
Ah Encapsulation Mode:	
ESP Encapsualtion Mode:	transport
Ah Authentication Algorithm:	
Encryption Algorithm:	DES_CBC_8
Esp Authentication Algorithm:	HMAC_MD5
Connection:	R2505 RA28-RA03 Anchor Connec
Source Network Object:	R2505 OS390 Local 192.168.100.9
Destination Network Object:	R2505 OS390 Remote 192.168.10

Figure 285. View activated dynamic VPN connection

Now you can start the traffic between the two hosts. All types of traffic will be permitted.

E.5.2 Business partner or supplier scenario: details

Many companies use frame relay or leased lines to provide secure communications with their business partners, subsidiaries, and vendors. Unfortunately, these solutions are often expensive and geographically limiting. OS/390 dynamic VPN offers an alternative for companies who desire private, cost-effective communications.

Suppose you are a major parts supplier to a manufacturer. Since it is critical that you have the specific parts and quantities at the exact time required by the manufacturing firm, you always need to be aware of the manufacturer's inventory status and production schedules. You handle this interaction manually today, and find it time consuming, expensive and even inaccurate at times. You want to find an easier, faster, and more effective way to communicate with your manufacturing company. However, given the confidentiality and time-sensitive nature of the information you exchange, the manufacturer does not want to publish it on its corporate Web site or distribute it monthly in an external report. By exploiting the public Internet, you can easily establish a virtual private network (VPN) to meet the needs of both companies.

E.5.2.1 Planning

To set up a VPN that connects your company to its business partners, there are several things you need to consider. First, what level of security do you require? Obviously, you want to keep your data confidential as it travels across the public Internet, but do you need to protect data as it flows through both company's intranets as well? Next, you need to understand what effect the VPN will have on system performance. For example, if you require maximum security from the VPN, then you can expect your system performance to be affected. Finally, you must develop a plan for implementing the VPN that includes information such as source and destination IP addresses (or some other means to identify your communicating systems), a mutually agreed upon pre-shared key, your authentication and encryption strategy, and various other factors.

In this scenario, your company wants to establish a VPN between an OS/390 host in your parts division and a client workstation in the manufacturing department of your business partner. The first step is to gather the information required to complete the definitions you need. The following planning checklists illustrate the type of information you will need.

In this case we are going to connect to an untrusted network, so we do not know how dangerous that could be. We choose to use the highest security and lowest performance specifications at the key management and data management levels. Table 40 and Table 41 show the main planning parameters and the key policy planning worksheet we use for this scenario.

Table 40. Initial design worksheet for dynamic connections

This is the information needed to create dynamic VPNs	**Answers**
What is the type of connection to be created? - Gateway-to-gateway - Host-to-gateway - Gateway-to-host - Host-to-host - Gateway-to-dynamic IP user - Host-to-dynamic IP user	OS/390 host Windows NT host
What type of security and system performance is required to protect the keys? - Highest security, lowest performance - Balance security and performance - Lowest security and highest performance - Add your own definition	Highest security and lowest performance
What type of security and system performance is required to protect the data? - Highest security, lowest performance - Balance security and performance - Lowest security and highest performance - Add your own definition	Highest security and lowest performance

Table 41. Key management planning - business partner

VPN Parameter	**Value**
Key Management Policy : Key Policy, Proposal, Transform	
Key Policy Name : Key Policy Medium	
Initiator Negotiation	Main
Responder Negotiation	Main
Key Proposal Name : Key Proposal Low-Med Security	
1. Key Transform Name : Lowest Security and High Performance	
Authentication Method	Pre-shared Keys
Hash Algorithm	SHA
Encryption Algorithm	3DES
Diffie-Hellman Group	Group 1
Maximum Key Lifetime	60
Maximum Size Limit	0
Key Lifetime Range	60-1440
Size Limit Range	0

Next, we need a data management policy; Table 42 shows the parameters we use here:

Table 42. Data management planning

Data Management Policy : Data Policy, Proposal, AH and ESP Transform:	
Data Policy Name : Data Policy based on low_ESP Transport	
Perfect Forward Secrecy (PFS)	Group1
Data Proposal Name : Data Proposal	
1. AH Transform Object Name :	
AH Encapsulation Mode	Tunnel
AH Authentication Algorithm	SHA
AH Maximum Data Lifetime	10
AH Maximum Size Limit	0
AH Data Lifetime Range	10-480
AH Size Limit Range	0
AH Size Limit Range	
1. ESP Transform Object Name : IBM_low_ESP_transport	
ESP Encapsulation Mode	Tunnel
ESP Authentication Algorithm	HMAC_MD5

Data Management Policy : Data Policy, Proposal, AH and ESP Transform:	
ESP Encryption Algorithm	DES_CBC_8
ESP Maximum Data Lifetime	30
ESP Maximum Size Limit	0
ESP Data Lifetime Range	30-480
ESP Size Limit Range	0

Now, we define the attributes of our dynamic tunnel policy. Remember that this information is going to be associated later to the data management policy.

Table 43. Dynamic tunnel policy

Dynamic Tunnel Policy:	
Initiation	Either
Connection Lifetime	1440

Finally, we plan the specific definitions of our tunnel; information such as key server management and authentication information will be put here.

Note that in this case we use pre-shared keys mode as the authentication method.

Table 44. Other planning information

VPN Parameter	Value
Authentication Information:	
Remote Key Server	10.11.12.102
Authentication Method	Pre-shared Keys
Shared Key	abcd
Certificate Authority:	
Racdcert Label	non-used
Dynamic Connection:	
Source	192.168.100.100
Destination	10.11.12.102
Source Port	0
Destination Port	0
Automatic Activation	No
Protocol	all
Remote Key Server	10.11.12.102
Key Servers:	
Local Key Server ID Type	IPV4
Local Key Server ID	192.168.100.100

VPN Parameter	Value
Remote Key Server ID Type	IPV4
Remote Key Server ID	10.11.12.102

7.3.3.2 Scenario overview

A business partner scenario has the following attributes:

- Partners do not trust each other's intranet
- Concerns include host-to-host security and access to the private intranet
- VPN is implemented in gateways and hosts

Figure 286 is an example of such a business partner scenario:

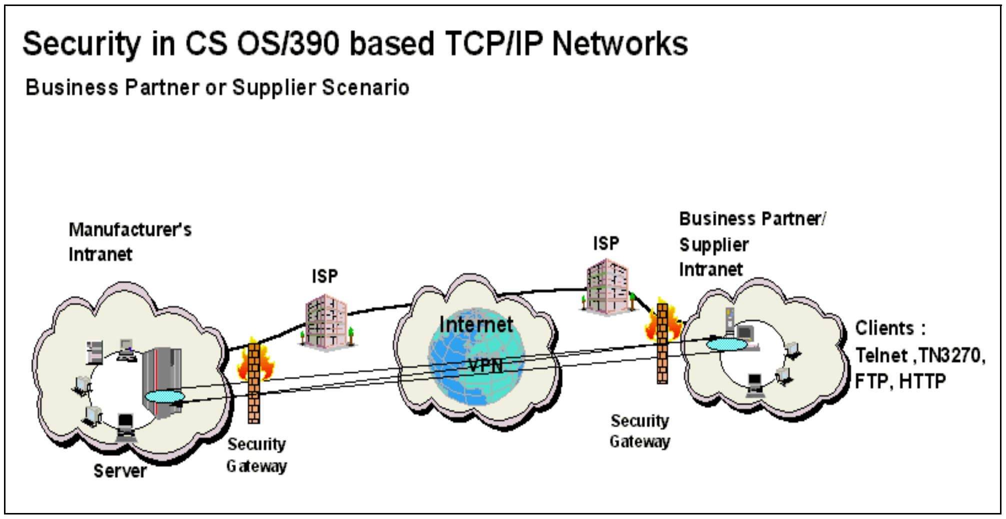

Security in CS OS/390 based TCP/IP Networks
Business Partner or Supplier Scenario
Manufacturer's Intranet
ISP
Internet
VPN
ISP
Business Partner/ Supplier Intranet
Clients : Telnet ,TN3270, FTP, HTTP
Security Gateway
Security Gateway
Server

Figure 286. Business partner scenario

Note that the connection is between the two hosts, in this case, an OS/390 system and a Windows NT machine. The configuration is made in tunnel mode. In our laboratory, we recreated this scenario as shown in Figure 287:

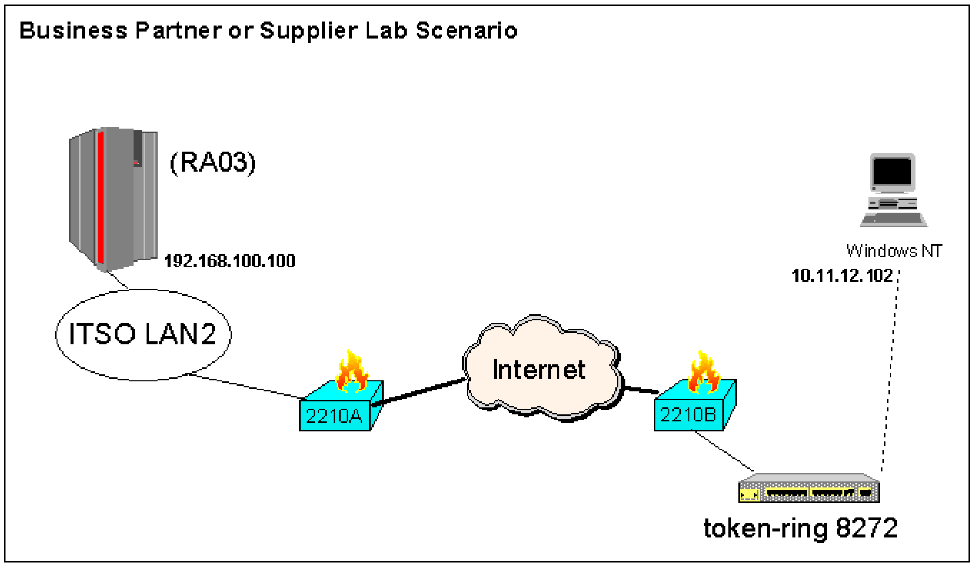


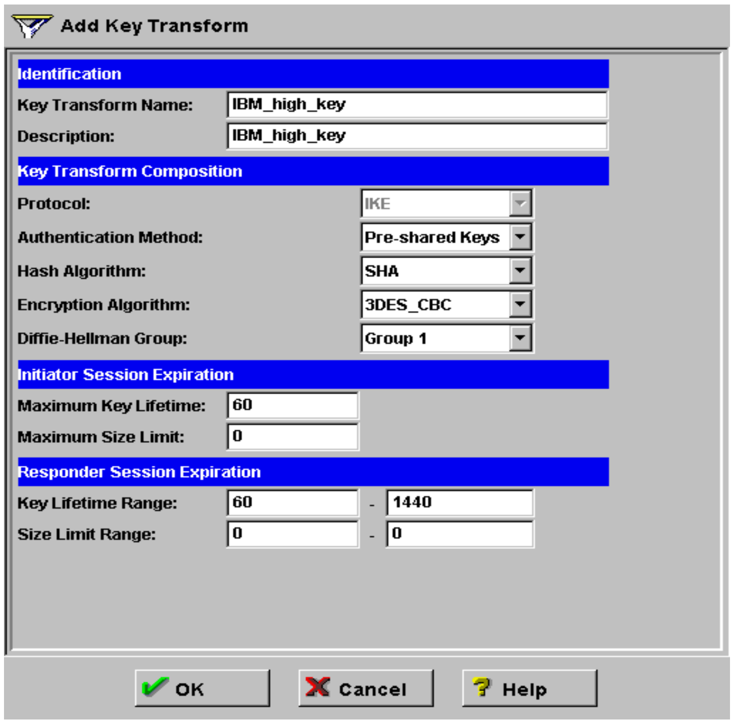
Figure 287. Business partner scenario - our lab

E.5.2.2 Scenario setup: OS/390 to Windows NT

We create a dynamic VPN connection using the Firewall Technologies GUI, between the OS/390 host RA03 and the Windows NT machine.

Key management definitions

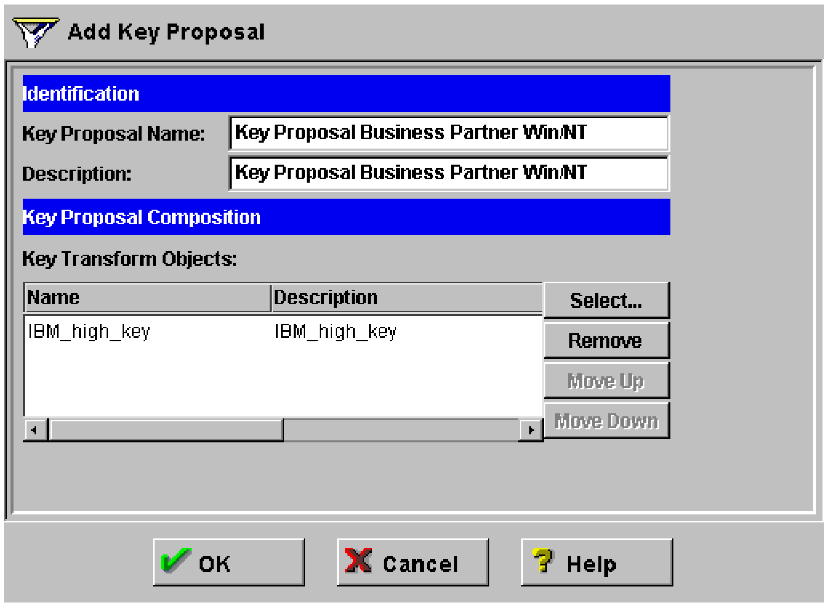
We define a new key transform object based on our chosen policy; please refer to Table 41 on page 411. Figure 288 shows the parameters we defined:



Identification	
Key Transform Name:	IBM_high_key
Description:	IBM_high_key
Key Transform Composition	
Protocol:	IKE
Authentication Method:	Pre-shared Keys
Hash Algorithm:	SHA
Encryption Algorithm:	3DES_CBC
Diffie-Hellman Group:	Group 1
Initiator Session Expiration	
Maximum Key Lifetime:	60
Maximum Size Limit:	0
Responder Session Expiration	
Key Lifetime Range:	60 - 1440
Size Limit Range:	0 - 0

Figure 288. Adding a high level security key transform

Now, we define our key proposal; please refer to Table 41 on page 411 and to Figure 289.


Add Key Proposal

Identification

Key Proposal Name:	Key Proposal Business Partner Win/NT
Description:	Key Proposal Business Partner Win/NT

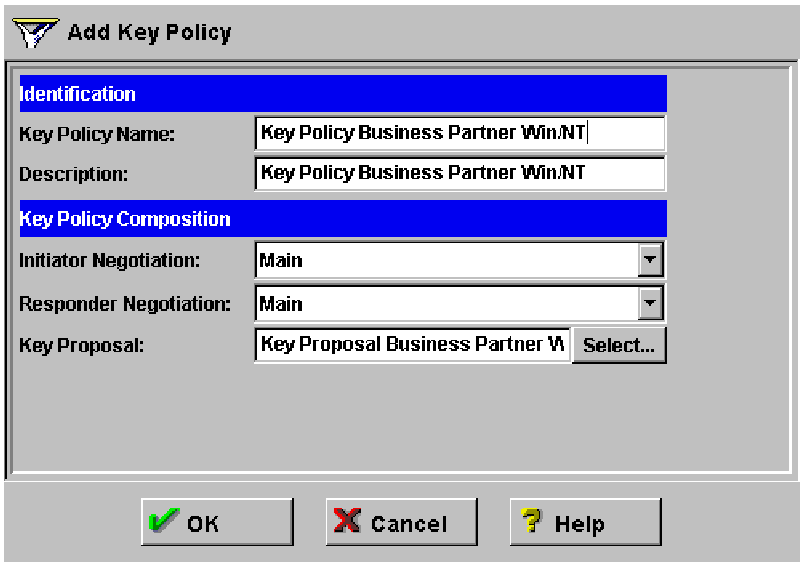
Key Proposal Composition

Key Transform Objects:

Name	Description
IBM_high_key	IBM_high_key

Figure 289. Adding a new key proposal

Finally, we define our key policy, referring to the previous definitions of the key proposal and key transform objects. See Figure 290:


Add Key Policy

Identification

Key Policy Name:	Key Policy Business Partner Win/NT
Description:	Key Policy Business Partner Win/NT

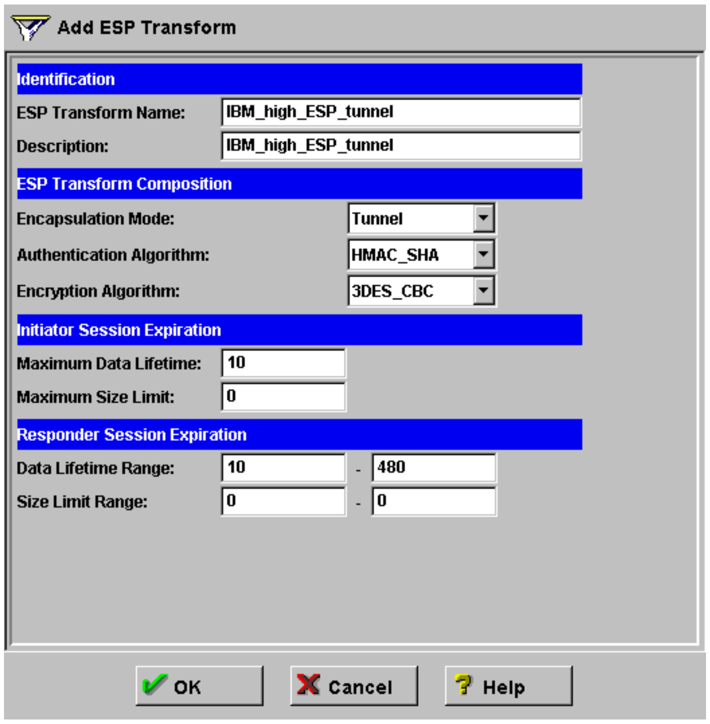
Key Policy Composition

Initiator Negotiation:	Main
Responder Negotiation:	Main
Key Proposal:	Key Proposal Business Partner W

Figure 290. Adding business partner Key Policy

Data management definitions

In this scenario, we choose to define two objects of data transform. The first one is a high level security ESP transform object, and the second one is a high level security AH transform object. Figure 291 and Figure 292 show our definitions for these two objects.



Identification

ESP Transform Name:	IBM_high_ESP_tunnel
Description:	IBM_high_ESP_tunnel

ESP Transform Composition

Encapsulation Mode:	Tunnel
Authentication Algorithm:	HMAC_SHA
Encryption Algorithm:	3DES_CBC

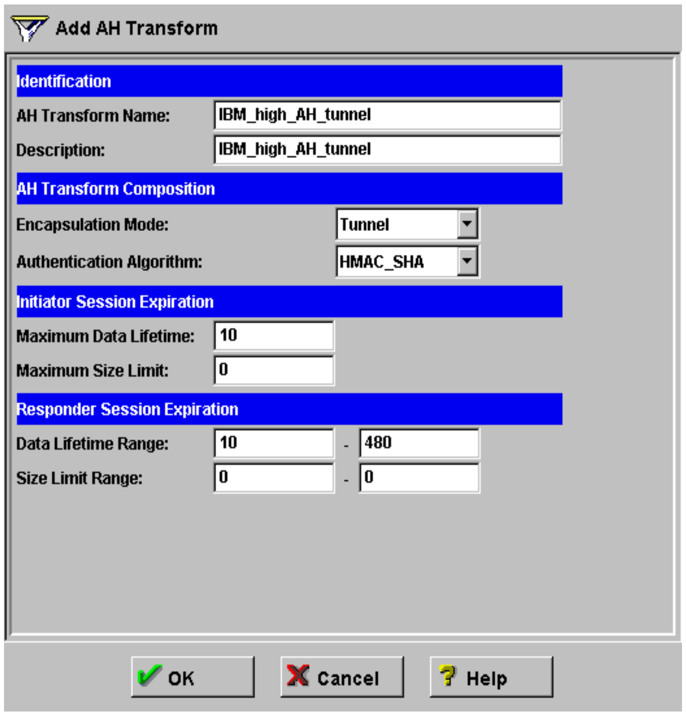
Initiator Session Expiration

Maximum Data Lifetime:	10
Maximum Size Limit:	0

Responder Session Expiration

Data Lifetime Range:	10	- 480
Size Limit Range:	0	- 0

Figure 291. Defining the high level security ESP transform object



Identification

AH Transform Name:	IBM_high_AH_tunnel
Description:	IBM_high_AH_tunnel

AH Transform Composition

Encapsulation Mode:	Tunnel
Authentication Algorithm:	HMAC_SHA

Initiator Session Expiration

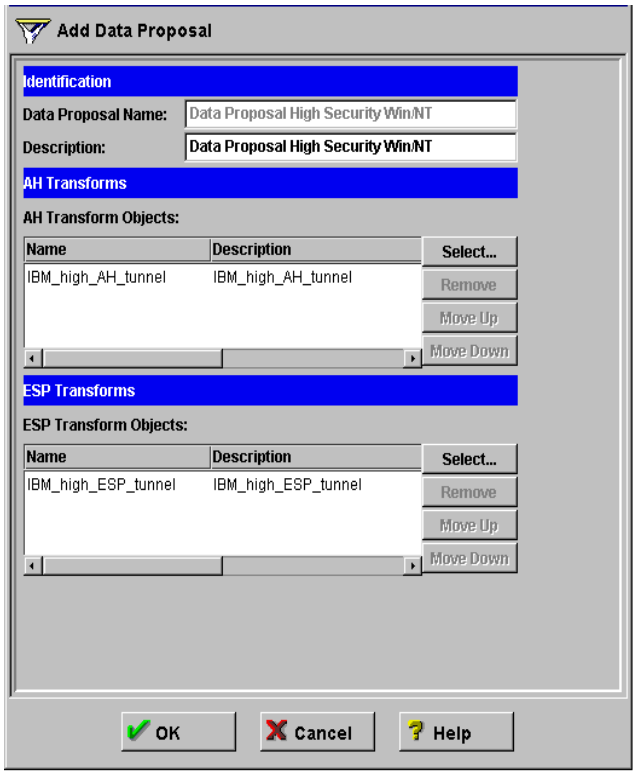
Maximum Data Lifetime:	10
Maximum Size Limit:	0

Responder Session Expiration

Data Lifetime Range:	10	- 480
Size Limit Range:	0	- 0

Figure 292. Adding high level security AH transform object

We create the AH and ESP transform objects according to Table 42 on page 411. After this, we create our data proposal, and select the previously defined transform objects. See Figure 293.


Add Data Proposal

Identification

Data Proposal Name: Data Proposal High Security Win/NT

Description: Data Proposal High Security Win/NT

AH Transforms

AH Transform Objects:

Name	Description
IBM_high_AH_tunnel	IBM_high_AH_tunnel

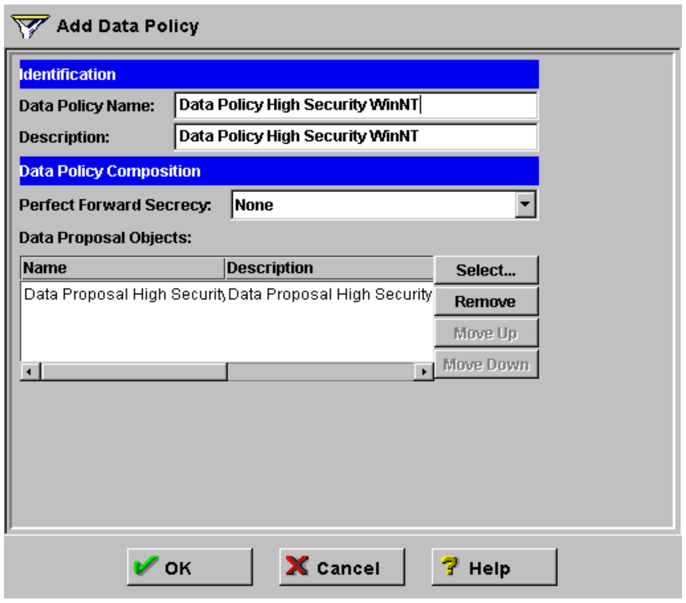
ESP Transforms

ESP Transform Objects:

Name	Description
IBM_high_ESP_tunnel	IBM_high_ESP_tunnel

Figure 293. Adding a high level security data proposal

Finally we define our data policy based on our previous definitions. Please see Figure 294:


Add Data Policy

Identification

Data Policy Name: Data Policy High Security WinNT

Description: Data Policy High Security WinNT

Data Policy Composition

Perfect Forward Secrecy: None

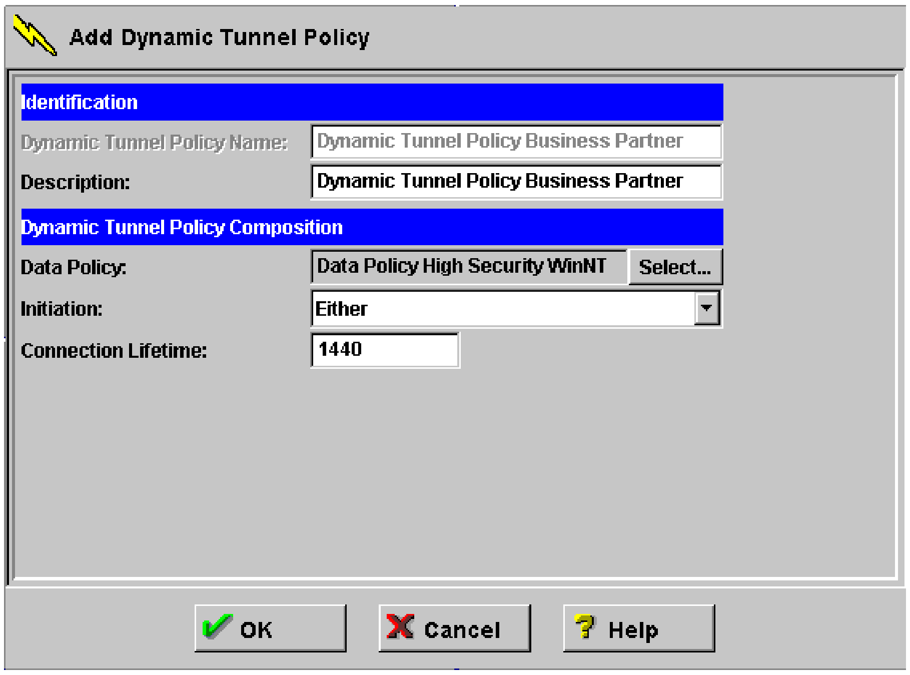
Data Proposal Objects:

Name	Description
Data Proposal High Securit	Data Proposal High Security

Figure 294. Adding high level security data policy

Dynamic tunnel definitions

Now we create our dynamic tunnel definition and associate it to our previously created data policy. Please see Figure 295:



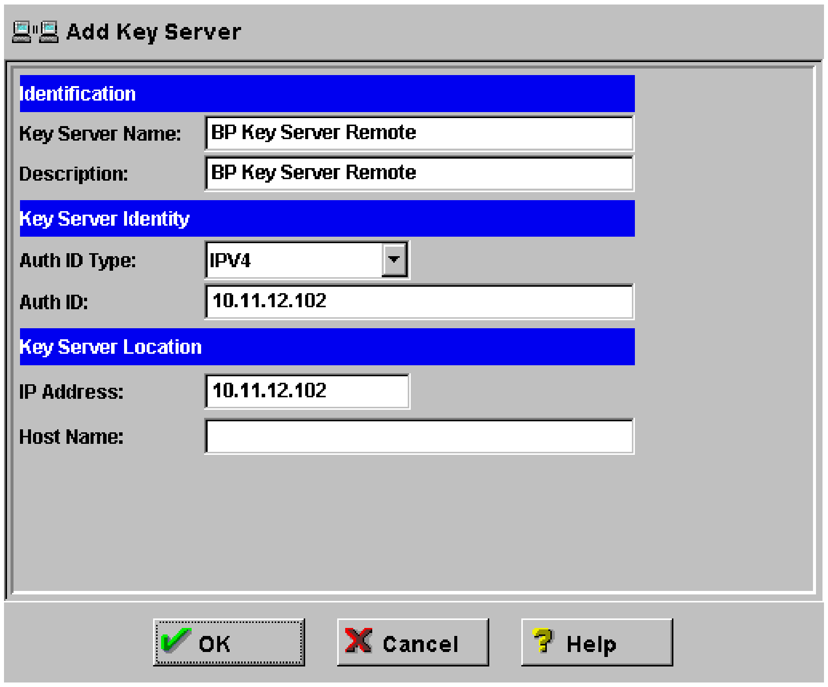
Dynamic Tunnel Policy Name: Dynamic Tunnel Policy Business Partner
Description: Dynamic Tunnel Policy Business Partner
Data Policy: Data Policy High Security WinNT
Initiation: Either
Connection Lifetime: 1440

Figure 295. Dynamic tunnel definition - business partner

Key server definitions

Now we define the key server objects. We have to define two such objects: the local key server (OS/390) and the remote key server (Windows NT). Then we have to define a key server group using the key server definitions.

In this case, we define our OS/390 RA03 and our business partner machine (Windows NT) with authentication ID type IPv4, in other words, normal IP addresses are used to identify them. In the main window, double-click **Key Server**, then double-click **NEW** to add a key server. Figure 296 and Figure 297 show the remote and local key server definitions, respectively.



Add Key Server

Identification

Key Server Name:	BP Key Server Remote
Description:	BP Key Server Remote

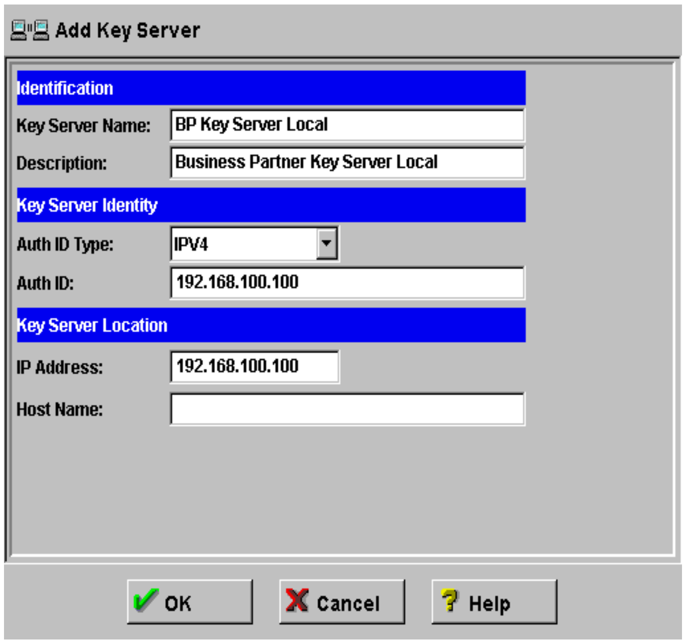
Key Server Identity

Auth ID Type:	IPV4
Auth ID:	10.11.12.102

Key Server Location

IP Address:	10.11.12.102
Host Name:	

Figure 296. Adding remote key server on OS/390 - definition for Windows NT



Add Key Server

Identification

Key Server Name:	BP Key Server Local
Description:	Business Partner Key Server Local

Key Server Identity

Auth ID Type:	IPV4
Auth ID:	192.168.100.100

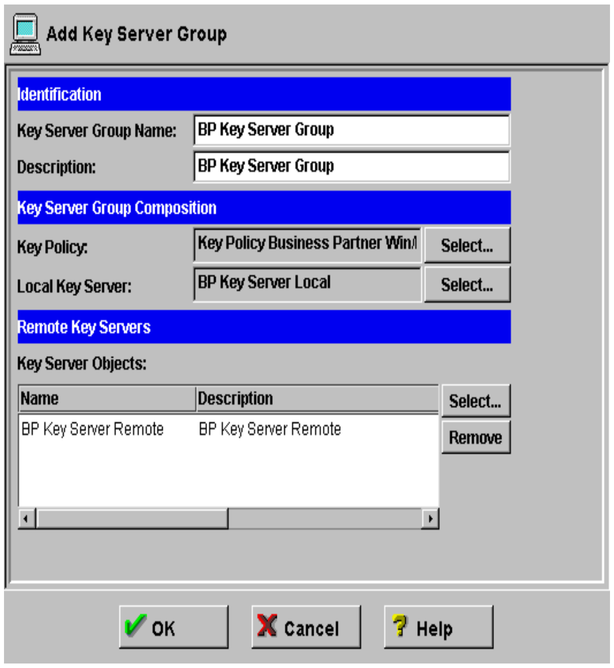
Key Server Location

IP Address:	192.168.100.100
Host Name:	

Figure 297. Adding local key server on OS/390 - OS/390 RA03 definition

In each case, we fill in the fields as shown, and click **OK**. Finally click **Close** and return to the main window.

Now, we have to associate both key servers with our key server group definition. In the main window, double-click **Key Server Group**, then double-click **NEW** to add a key server group. You will see the window in Figure 298.


Add Key Server Group

Identification

Key Server Group Name: BP Key Server Group

Description: BP Key Server Group

Key Server Group Composition

Key Policy: Key Policy Business Partner Win/

Local Key Server: BP Key Server Local

Remote Key Servers

Key Server Objects:

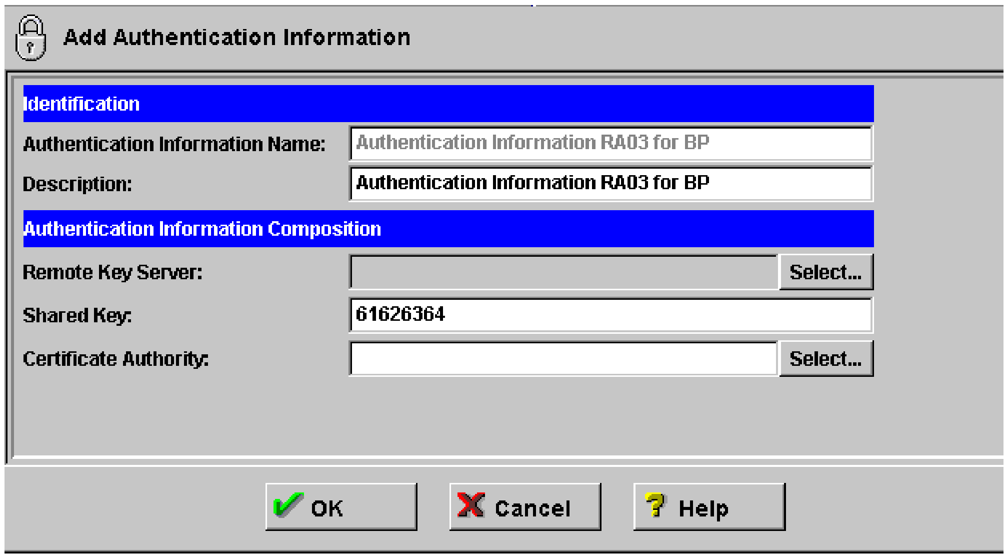
Name	Description
BP Key Server Remote	BP Key Server Remote

Figure 298. Adding Key Server Group on OS/390

Click **Select ...** beside Key Policy and choose the key policy created in Figure 290 on page 415. Click **Select ...** beside Local Key Server and choose the local key server created in Figure 297 on page 419. Click **Select ...** under Remote Key Servers and choose the remote key server created in Figure 296 on page 419. Then click **OK**. In the next window click **Close** and return to the main window.

Authentication information definition

Now we define the authentication information definition object; we authenticate the remote key server. In the main window, double-click **Authentication Information**, then double-click **NEW** to add authentication information. Figure 299 illustrates.


Add Authentication Information

Identification

Authentication Information Name: Authentication Information RA03 for BP

Description: Authentication Information RA03 for BP

Authentication Information Composition

Remote Key Server:

Shared Key: 61626364

Certificate Authority:

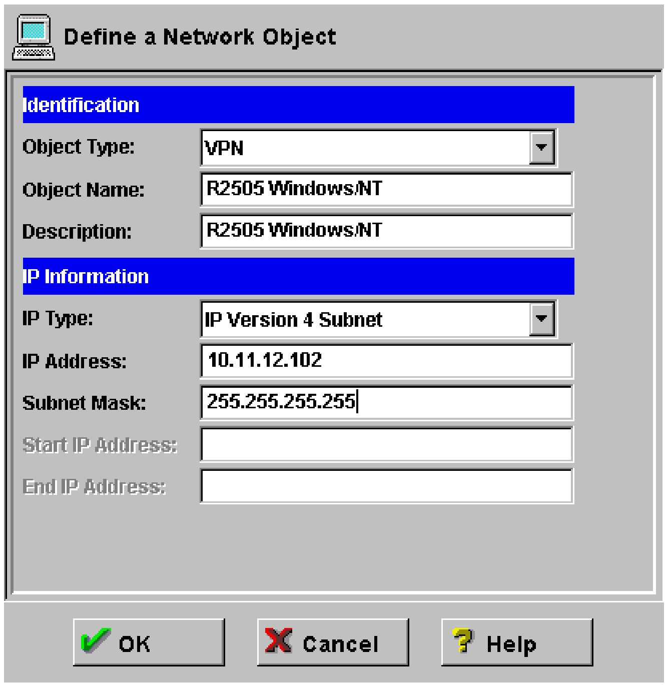
Figure 299. Adding authentication information on OS/390

Click **Select ...** and choose the Remote Key Server created in Figure 296 on page 419. Then click **OK.** In the next window click **Close** and return to the main window. Remember that we are using pre-shared key mode; we defined the key as abcd, but this field should be in hexadecimal format so we see 61626364 in this field.

Dynamic VPN connection definition

Now we define the dynamic VPN connection definition. We will associate the network objects from both endpoints of the tunnel to the remote key server and key server group.

We have to create two network objects identifying the OS/390 RA03 system and the Windows NT machine. In the main window, double-click **Network Objects**, then double-click **NEW** to add a network object. You will see the window in Figure 300.


Define a Network Object

Identification

Object Type:	VPN
Object Name:	R2505 Windows/NT
Description:	R2505 Windows/NT

IP Information

IP Type:	IP Version 4 Subnet
IP Address:	10.11.12.102
Subnet Mask:	255.255.255.255
Start IP Address:	
End IP Address:	

Figure 300. Adding a network object for Windows NT

Fill in the fields as shown and click **OK**.

In our previous scenario, we have already defined the OS/390 RA03 system with IP address 192.168.100.100.

Network object

The Network objects function allows you to maintain information about network addressable components on your network. This function acts as a central repository for use by other functions in the OS/390 system. Primarily, network objects are used to designate source and destination addresses when you create your connections.

Now we will define the dynamic VPN connection between these two network objects. In the main window, double-click **VPN Connection Setup**, then double-click **NEW** to add a dynamic VPN connection. Please refer to Figure 274 on page 399 for an example of the window to be filled in.

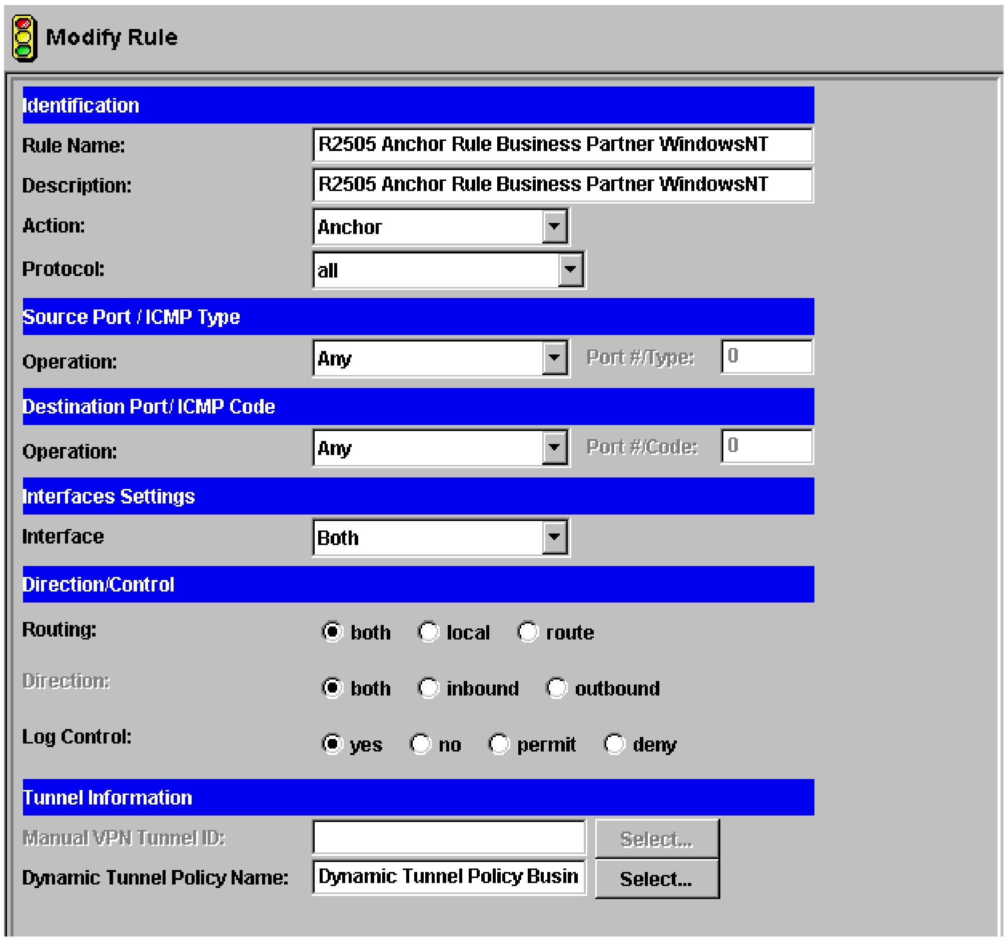
Click **Select ...** beside Source and choose the OS/390 network object created in Figure 273 on page 398 (from our previous scenario). Click **Select ...** beside Destination and choose the Windows NT network object created in Figure 300 on page 422. Click **Select ...** beside Remote Key Server and choose the remote key server created in Figure 296 on page 419. Click **Select ...** beside Key Server Group and choose the key server group created in Figure 298 on page 420. Then click **OK**. In the next window click **Close** and return to the main window.

Creating the anchor filter rules, services and connections

Now we have to create the anchor filter rules, the services that will allow the tunnel endpoints to use AH and ESP protocol, and the normal services and connections to create the filter rules and the dynamic filter rules.

In this anchor filter rule we will allow all types of traffic to flow in the tunnel. We will also associate this anchor filter rule with a dynamic tunnel policy.

In the main window, double-click **Connection Templates** and **Rules**, then double-click **NEW** to add a new rule.


Modify Rule

Identification

Rule Name:	R2505 Anchor Rule Business Partner WindowsNT
Description:	R2505 Anchor Rule Business Partner WindowsNT
Action:	Anchor
Protocol:	all

Source Port / ICMP Type

Operation:	Any	Port #/Type:	0

Destination Port/ ICMP Code

Operation:	Any	Port #/Code:	0

Interfaces Settings

Interface	Both

Direction/Control

Routing:	[✓ both] local route
Direction:	[✓ both] inbound outbound
Log Control:	[✓ yes] no permit deny

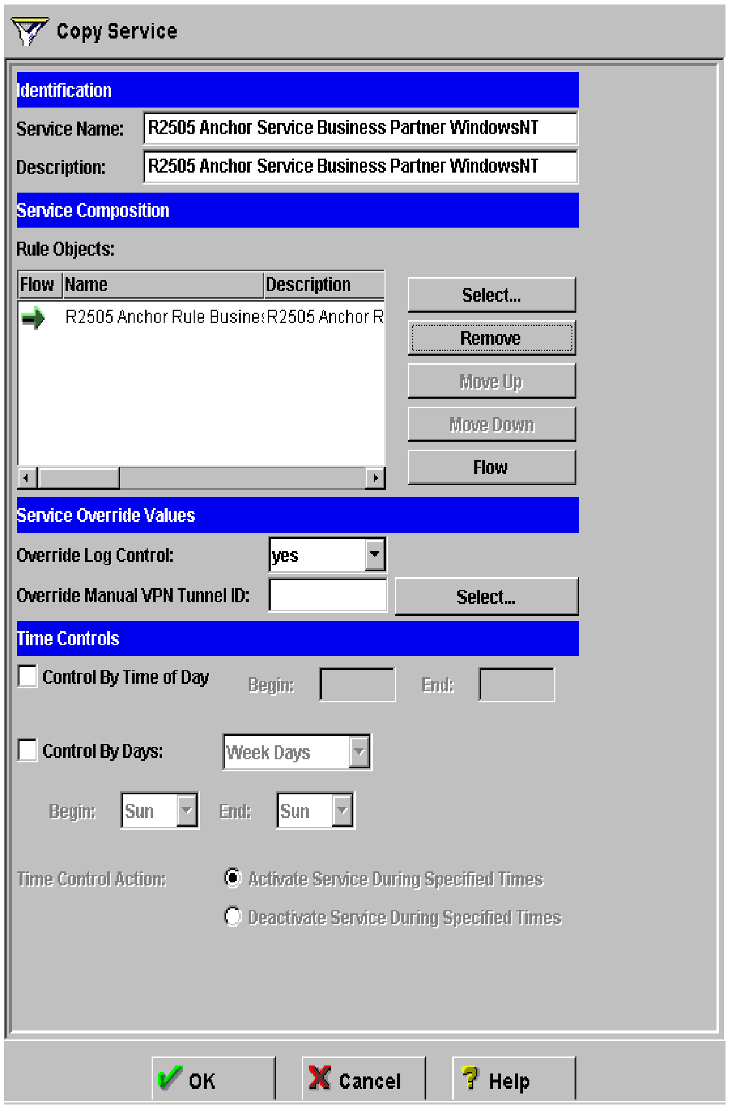
Tunnel Information

Manual VPN Tunnel ID:		Select...
Dynamic Tunnel Policy Name:	Dynamic Tunnel Policy Busin	Select...

Figure 301. Creating an anchor rule on OS/390

Click **Select ...** beside Dynamic Tunnel Policy Name and choose the dynamic tunnel policy created in Figure 295 on page 418. Then click **OK**. In the next window click **Close** and return to the main window.

We now have to create two Services: one service establishes the anchor filter rule between the connection endpoints and the other establishes the permit filter rules for the key server traffic. In the main window, double-click **Services**, then double-click **NEW** to add a new service.


Copy Service

Identification

Service Name: R2505 Anchor Service Business Partner WindowsNT

Description: R2505 Anchor Service Business Partner WindowsNT

Service Composition

Rule Objects:

Flow	Name	Description
➡	R2505 Anchor Rule Busines	R2505 Anchor R

Service Override Values

Override Log Control: yes

Override Manual VPN Tunnel ID:

Time Controls

☐ Control By Time of Day Begin: End:

☐ Control By Days: Week Days

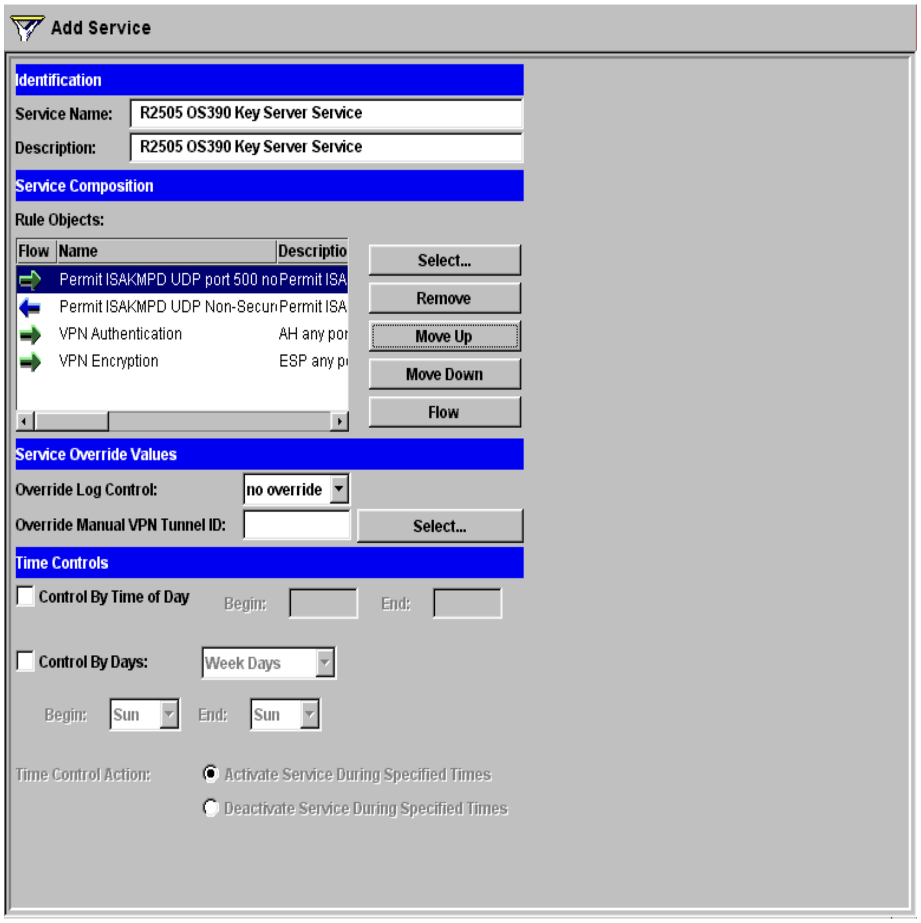
Begin: Sun End: Sun

Time Control Action: [✓ Activate Service During Specified Times] Deactivate Service During Specified Times

Figure 302. Adding anchor service on OS/390

You will see the window above. Fill in the fields as shown. Click **Select ...** under Rule Objects and choose the anchor filter rule created in Figure 301 on page 423. Then click **OK**.

For the second (key server) service, perform the same steps again (Figure 303).



Add Service

Identification

Service Name: R2505 OS390 Key Server Service

Description: R2505 OS390 Key Server Service

Service Composition

Rule Objects:

Flow	Name	Descriptio
⇨	Permit ISAKMPD UDP port 500 no	Permit ISA
⇦	Permit ISAKMPD UDP Non-Secur	Permit ISA
⇨	VPN Authentication	AH any por
⇨	VPN Encryption	ESP any p

Service Override Values

Override Log Control: no override

Override Manual VPN 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 303. Adding key server service on OS/390

Click **Select ...** under Rule Objects and choose the following filter rules:

- ISAKMPD UDP Non-Secure traffic in both directions
- VPN Authentication traffic
- VPN Encryption object

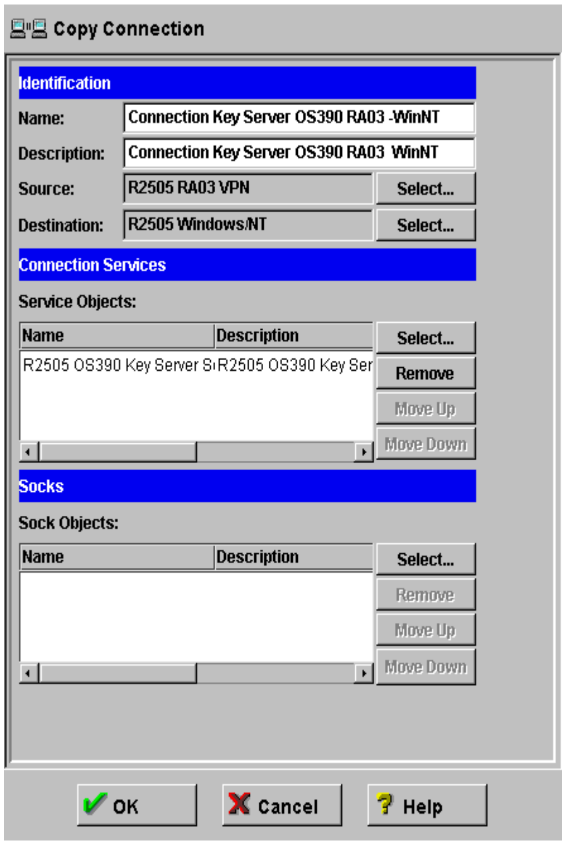
After choosing these rule objects, click **OK.** In the next window click **Close** and return to the main window.

Note that the definitions selected here are predefined rule objects.

Now we create the connections to associate the service between the tunnel endpoints. Then we will activate the connection to create the filter rules.

Figure 304 shows the first of these connections (the key server connection).

.



Name: Connection Key Server OS390 RA03 -WinNT

Description: Connection Key Server OS390 RA03 WinNT

Source: R2505 RA03 VPN

Destination: R2505 Windows/NT

Service Objects:

Name	Description
R2505 OS390 Key Server S	R2505 OS390 Key Ser

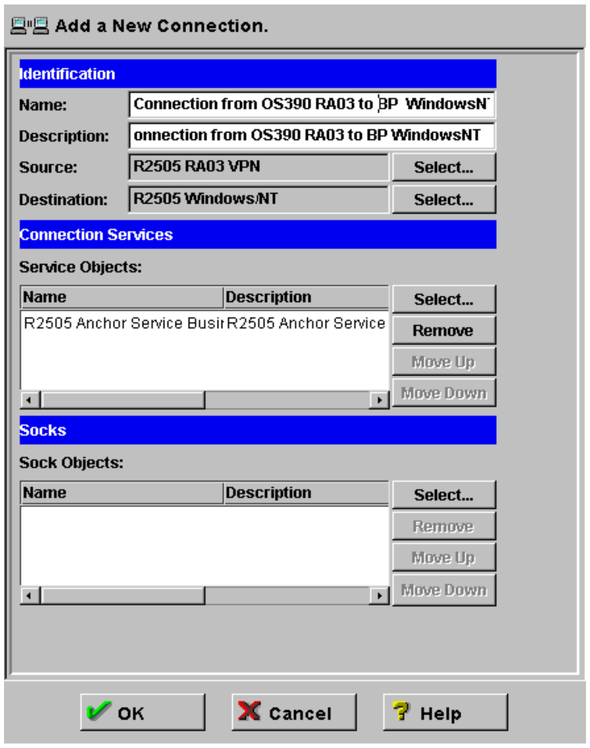
Sock Objects:

Name	Description

Figure 304. Adding a key server connection on OS/390

Click **Select ...** beside Source and choose the host's network object as created in Figure 273 on page 398. Click **Select ...** beside Destination and choose the network object created in Figure 300 on page 422. Add the key server object created in Figure 303 on page 425 to the Service Objects list. Then click **OK**.

Figure 305 shows the anchor connection.


Add a New Connection.

Identification

Name: Connection from OS390 RA03 to BP WindowsN

Description: onnection from OS390 RA03 to BP WindowsNT

Source: R2505 RA03 VPN

Destination: R2505 Windows/NT

Connection Services

Service Objects:

Name	Description
R2505 Anchor Service Busir	R2505 Anchor Service

Socks

Sock Objects:

Name	Description

Figure 305. Adding an anchor connection on OS/390

Click **Select ...** beside Source and choose the host's network object as created in Figure 273 on page 398. Click **Select ...** beside Destination and choose the network object created in Figure 300 on page 422. Click **Select** beside Service Objects and add the object created in Figure 302 on page 424. Then click **OK**.

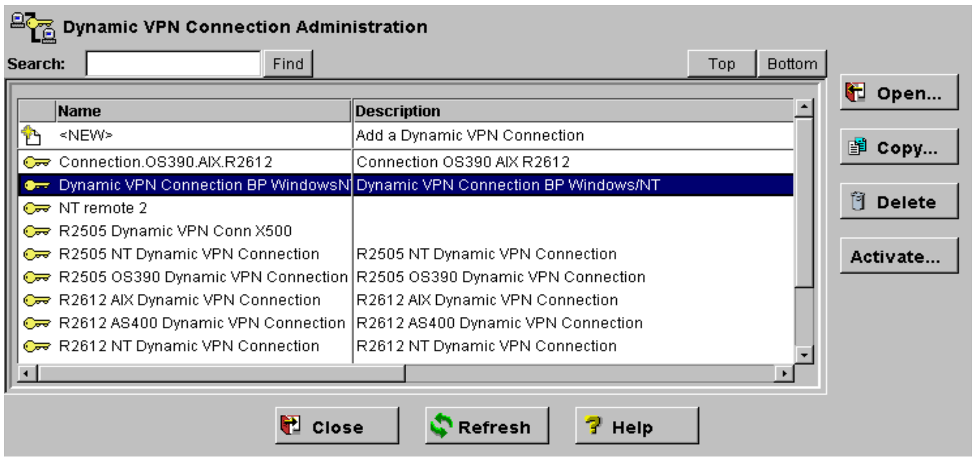
Activating the filter rules and the dynamic VPN connection

In the main window, double-click **Connection Activation**, then click **Execute**. Check the messages in the output area for errors. Then click **Close**.

Connection activation

Use this function to generate the rules based upon the configurations defined in the Connection Setup window and all of its subsidiary configurations (for example, services, rule templates, and SOCKS templates). Rules can also be generated depending upon settings in the Security Policy window. These rules become the active set through which the OS/390 system can evaluate network datagrams. If there is a set of connection rules already active, this procedure updates the active rules with the contents of the newly generated set. Feedback about a successful activation or any errors will be displayed in the output section.

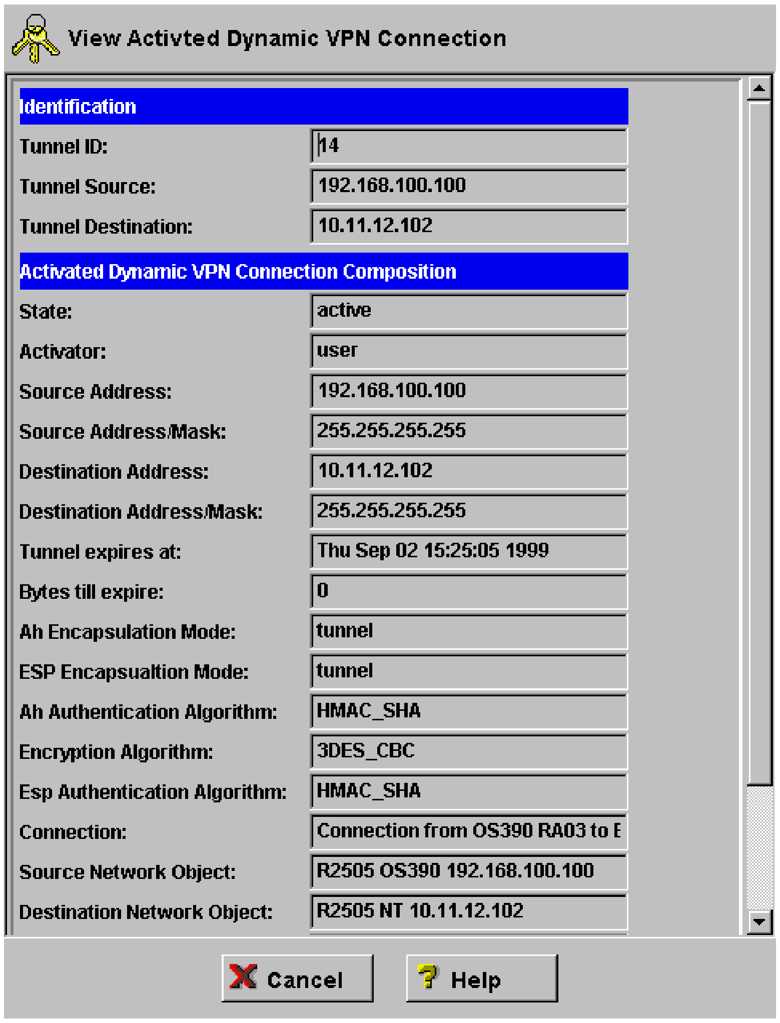
Now we will activate the dynamic VPN connection. In the main window, double-click **VPN Connection Setup**, then choose the dynamic VPN connection created in Figure 274 on page 399 and click **Activate**. You will see a window similar to Figure 306.



Name	Description
<NEW>	Add a Dynamic VPN Connection
Connection.OS390.AIX.R2612	Connection OS390 AIX R2612
Dynamic VPN Connection BP WindowsNT	Dynamic VPN Connection BP Windows/NT
NT remote 2	
R2505 Dynamic VPN Conn X500	
R2505 NT Dynamic VPN Connection	R2505 NT Dynamic VPN Connection
R2505 OS390 Dynamic VPN Connection	R2505 OS390 Dynamic VPN Connection
R2612 AIX Dynamic VPN Connection	R2612 AIX Dynamic VPN Connection
R2612 AS400 Dynamic VPN Connection	R2612 AS400 Dynamic VPN Connection
R2612 NT Dynamic VPN Connection	R2612 NT Dynamic VPN Connection

Figure 306. Dynamic VPN connection activation

You have to check the VPN connection activation window to see if the tunnel was activated. To check all the parameters in the tunnel, you can double-click the tunnel, or select the tunnel and click **View**. You will receive the following window showing all the definitions:


View Activted Dynamic VPN Connection

Identification	
Tunnel ID:	14
Tunnel Source:	192.168.100.100
Tunnel Destination:	10.11.12.102

Activated Dynamic VPN Connection Composition	
State:	active
Activator:	user
Source Address:	192.168.100.100
Source Address/Mask:	255.255.255.255
Destination Address:	10.11.12.102
Destination Address/Mask:	255.255.255.255
Tunnel expires at:	Thu Sep 02 15:25:05 1999
Bytes till expire:	0
Ah Encapsulation Mode:	tunnel
ESP Encapsualtion Mode:	tunnel
Ah Authentication Algorithm:	HMAC_SHA
Encryption Algorithm:	3DES_CBC
Esp Authentication Algorithm:	HMAC_SHA
Connection:	Connection from OS390 RA03 to E
Source Network Object:	R2505 OS390 192.168.100.100
Destination Network Object:	R2505 NT 10.11.12.102

Figure 307. View activated dynamic VPN connection

Now you can start the traffic between the two hosts. All types of traffic will be permitted.

E.5.2.3 Scenario setup: VPN client on Windows NT

To set up the Windows NT workstation, we used an early version of the IBM SecureWay VPN Client for Windows. *This product was never released*, but we have retained the description of the setup for the SecureWay VPN Client simply to show you the principles involved and the information that you need to configure. *The windows shown do not represent any publicly available IBM product*.

One popular VPN client that can be used as an alternative is the VPN Client from Information Resource Engineering (IRE). This client is sometimes built into other vendors' products, but the generally available version is known as SafeNet. We were unable to test the IRE client in conjunction with OS/390, but the redbook *A Comprehensive Guide to Virtual Private Networks Volume III: Cross-Platform Key and Policy Management*, SG24-5309, describes some scenarios in which this client is used. Please note that the version of SafeNet available at the time of writing does not support token-ring attachment. For more information on the availability of the IRE client, please refer to the IRE Web site at:

```
http://www.ire.com
```

A white paper entitled *VPN Interoperability Demonstrated for OS/390* by Linwood Overby of z/OS Communications Server Strategy and Design and Thomas Cosenza of z/OS Communications Server System Test is available at:

```
www.ibm.com/servers/eserver/zseries/networking/pdf/GM13-0029-00.pdf.
```

It describes how OS/390 IPSec interoperates in a variety of configuration scenarios with the IRE SafeNet/Soft-PK VPN Client in Windows NT and Windows 95, the integrated VPN client in Windows 2000, and Cisco security gateways.

Figure 308 shows the main menu that you get when you invoke the IBM SecureWay VPN client configuration. As you define each element, another button is enabled and you can define the next function.

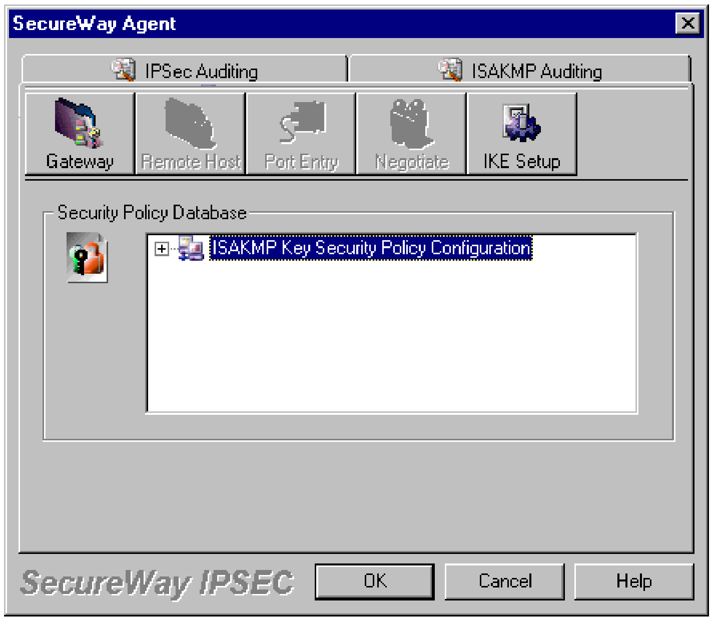
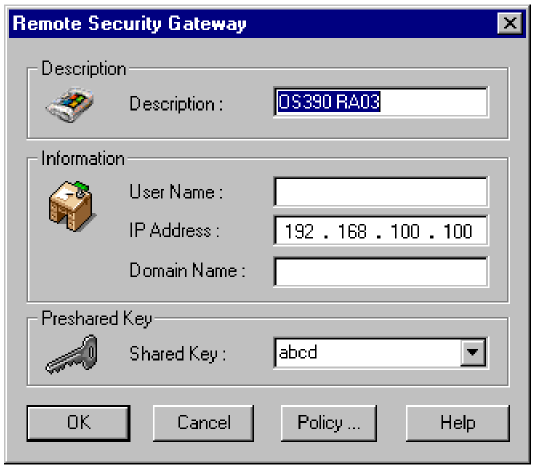


Figure 308. IPSec client principal menu

First we define the remote security gateway, or the VPN partner. We click **Gateway** on the main IBM SecureWay VPN Client window. In the window shown in Figure 309, enter:

Description	OS390 RA03
IP Address	192.168.100.100
Shared Key	abcd



Description : OS390 RA03

User Name :

IP Address : 192 . 168 . 100 . 100

Domain Name :

Shared Key : abcd

Figure 309. Defining the remote security gateway

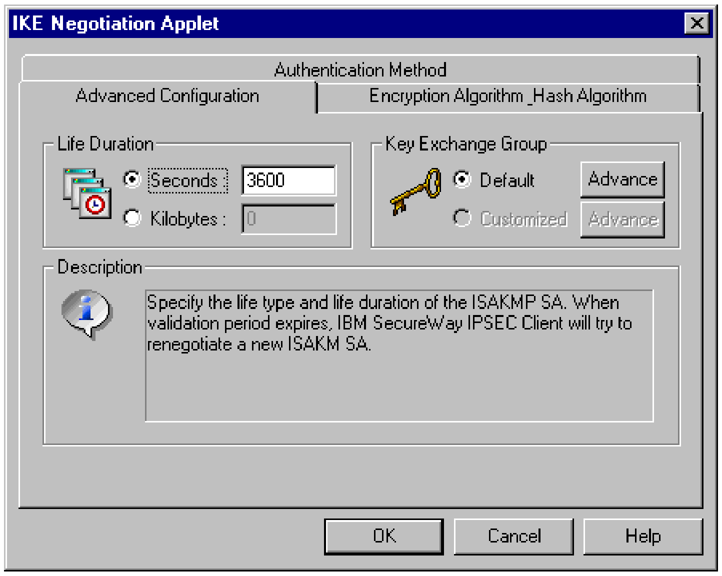
Our shared key value is abcd. Please see Figure 299 on page 421 where we defined this on OS/390. We are using pre-shared key mode.

Now, we click the **Policy** button in order to define the security association values and the characteristics of our encryption and hash algorithm values.

In the Advanced Configuration tab (Figure 310), we define the type and life duration of the IKE security association. In addition, you can define the key exchange group.

We click **Advanced Configuration** and set:

Life Duration (Seconds)	3600
Key Exchange Group	Oakley Default Group 1



(●) Seconds 3600

( ) Kilobytes : 0

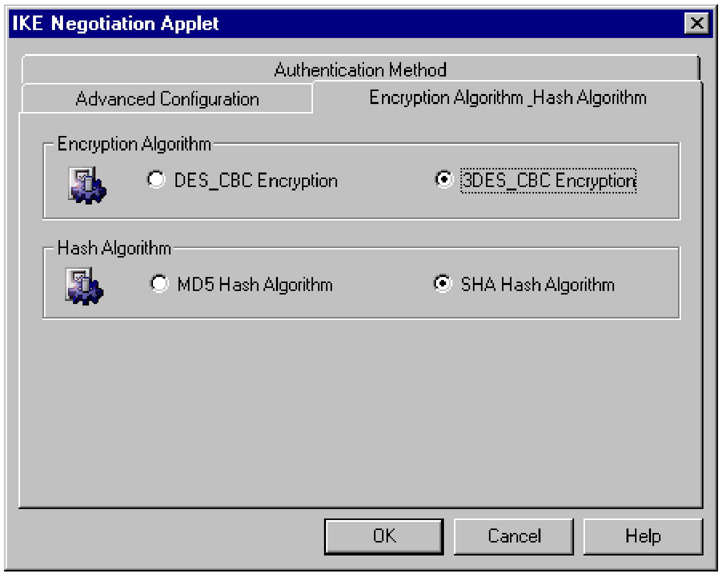
(●) Default

Specify the life type and life duration of the ISAKMP SA. When validation period expires, IBM SecureWay IPSEC Client will try to renegotiate a new ISAKM SA.

Figure 310. Remote security gateway - advanced configuration

We then select the **Encryption Algorithm_Hash Algorithm** tab (Figure 311) in order to define our preferences:

Encryption Algorithm	3DES_CBC Encryption
Hash Algorithm	SHA Hash Algorithm



Encryption Algorithm: DES_CBC Encryption [✓ 3DES_CBC Encryption]

Hash Algorithm: MD5 Hash Algorithm [✓ SHA Hash Algorithm]

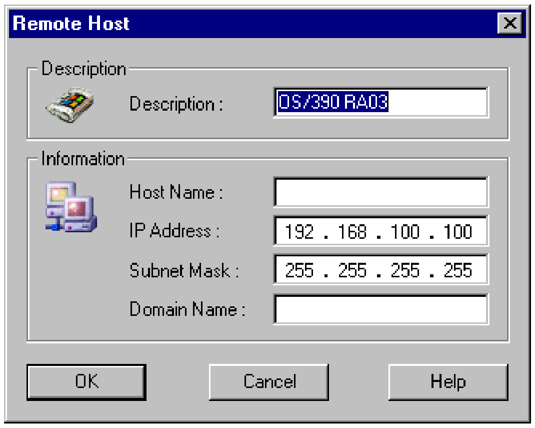
Figure 311. Encryption algorithm and hash algorithm

Now we configure the remote host. Since we are using a host-to-host connection, the gateway and the remote host will be the same physical machine. Therefore, we will use the same address we use in the gateway definition and set the subnet mask at 255.255.255.255, which will indicate that particular host. We perform the following steps:

1. On the main window, select the gateway just configured and click **Remote Host.**
2. Enter the following fields as in Figure 312:

Description	OS390 RA03
IP Address	192.168.100.100
Subnet Mask	255.255.255.255

Click **OK.**



Description : OS/390 RA03

Host Name :

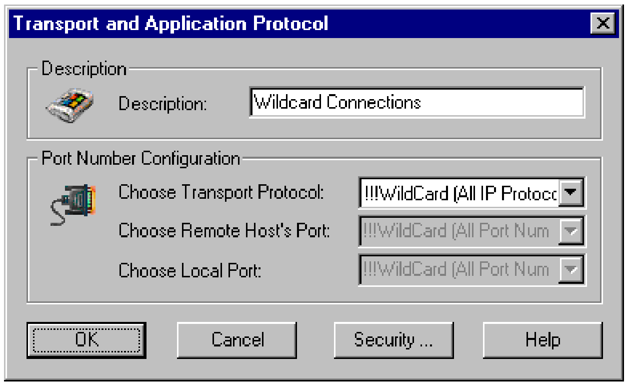
IP Address : 192 . 168 . 100 . 100

Subnet Mask : 255 . 255 . 255 . 255

Domain Name :

Figure 312. Remote host selection

3. In the main window, click the plus **(+)** sign next to the gateway definition (OS390 RA03) and then the plus sign next to the remote host just defined. This shows us the port entry (wildcard connections), which is generated automatically.
4. Right-click **Wildcard Connections** and click **Edit.** You will see Figure 313:

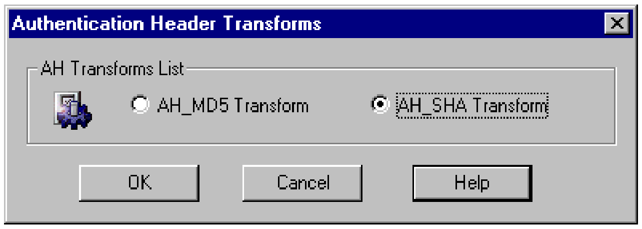


Description: Wildcard Connections
Choose Transport Protocol: !!!WildCard (All IP Protocc
Choose Remote Host's Port: !!!WildCard (All Port Num
Choose Local Port: !!!WildCard (All Port Num

Figure 313. Wildcard connections

In our example we selected wildcard connections that permitted all traffic to be passed. Your situation may require different selections.

5. Click **Security.**
6. Since only ESP is required in the IPSec parameters table, we only enabled **Require Data Confidentiality.**
7. Click **Advanced** for Require Data Confidentiality.
8. Mark **AH Transform List:** AH_SHA transform (see Figure 314):

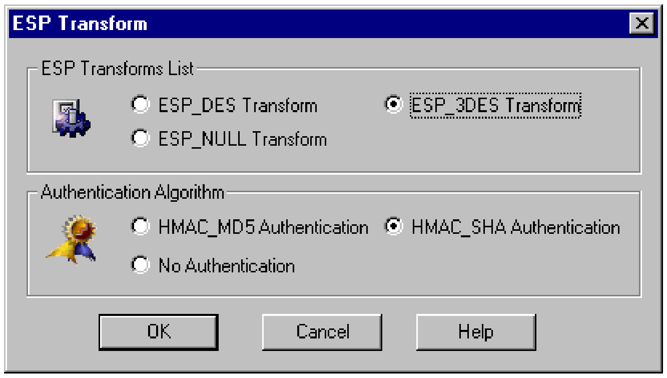


AH_MD5 Transform [✓ AH_SHA Transform]

Figure 314. AH selection

9. In addition mark the following (Figure 315):

ESP Transform List	ESP_3DES Transform
Authentication Algorithm	HMAC_SHA Authentication


ESP Transform

ESP Transforms List: ( ) ESP_DES Transform [✓ ESP_3DES Transform] ( ) ESP_NULL Transform

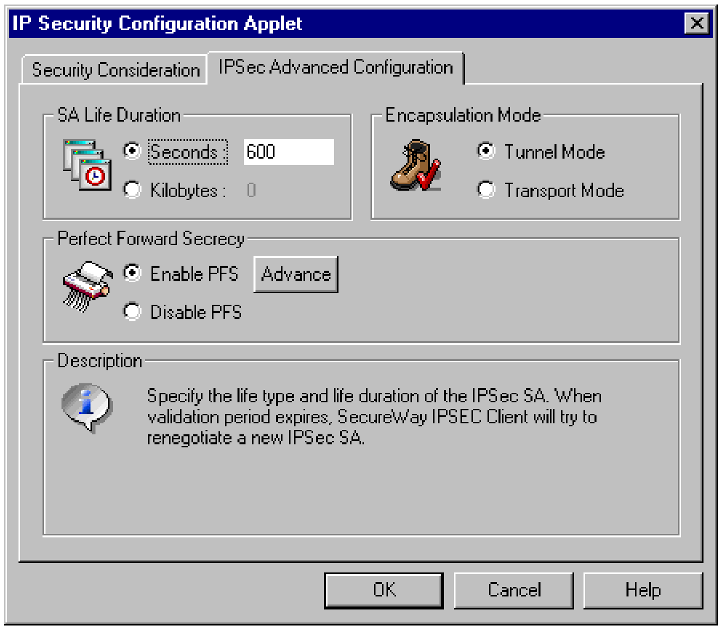
Authentication Algorithm: ( ) HMAC_MD5 Authentication [✓ HMAC_SHA Authentication] ( ) No Authentication

Figure 315. ESP selection

10.Click the **IPSec Advanced Configuration** tab to see Figure 316.

11.Mark:

SA Life Duration	600 sec. (default)
Perfect Forward Secrecy	Disable PFS
Encapsulation Mode	Tunnel Mode


IP Security Configuration Applet

SA Life Duration: [✓ Seconds] 600 ( ) Kilobytes : 0

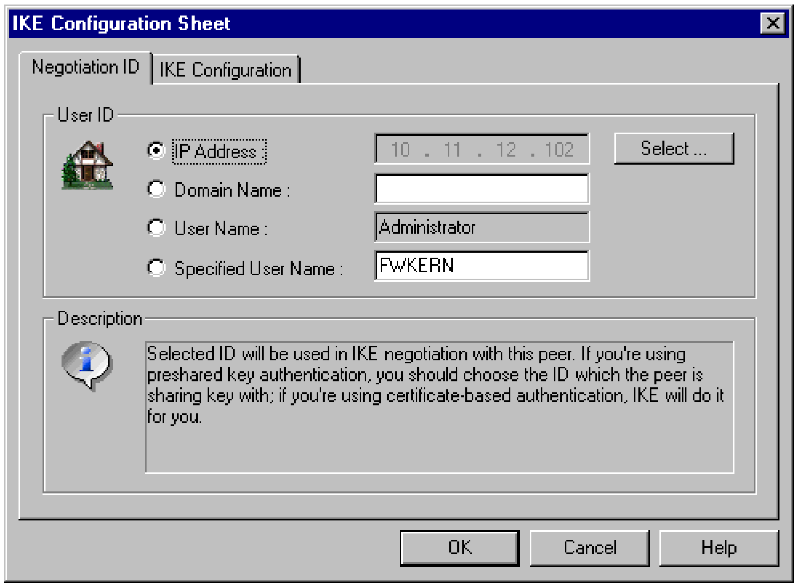
Encapsulation Mode: [✓ Tunnel Mode] ( ) Transport Mode

Perfect Forward Secrecy: [✓ Enable PFS] ( ) Disable PFS

Description: Specify the life type and life duration of the IPSec SA. When validation period expires, SecureWay IPSEC Client will try to renegotiate a new IPSec SA.

Figure 316. Security advanced configuration

Now we have finished configuring the Port Entry. Next, we perform the IKE setup by clicking the **IKE Setup** button in the main window. Figure 317 shows the window we get.



IKE Configuration Sheet

Negotiation ID | IKE Configuration

User ID

[✓ IP Address] 10 . 11 . 12 . 102

Domain Name :

User Name : Administrator

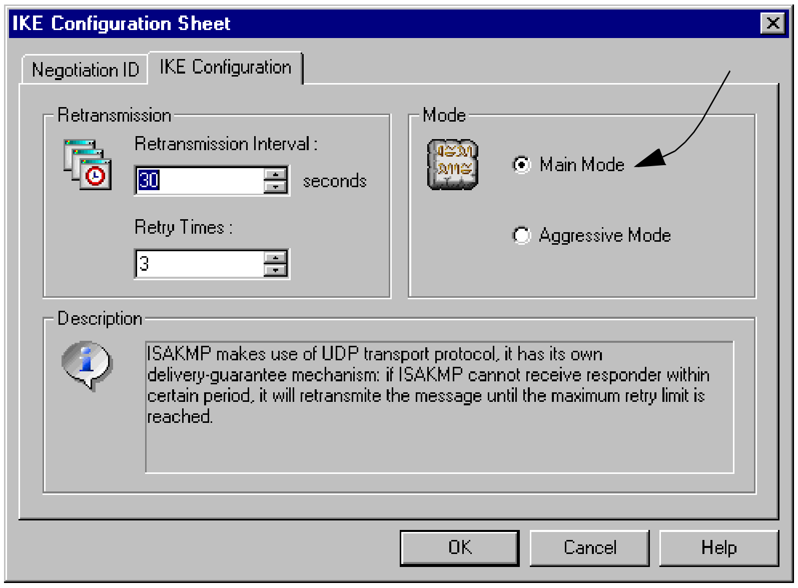
Specified User Name : FWKERN

Description

Selected ID will be used in IKE negotiation with this peer. If you're using preshared key authentication, you should choose the ID which the peer is sharing key with; if you're using certificate-based authentication, IKE will do it for you.

Figure 317. Negotiation ID setup

You should mark **IP Address** for the negotiation ID. Click the **IKE Configuration** tab and change the mode to **Main Mode** (Figure 318). Main mode refers to the Oakley protocol used to establish the security association between the VPN partners. Main mode keeps the identities of the partners secret but requires that you know your partner’s IP address.



IKE Configuration Sheet

Negotiation ID | IKE Configuration

Retransmission

Retransmission Interval : 30 seconds

Retry Times : 3

Mode

[✓ Main Mode]

Aggressive Mode

Description

ISAKMP makes use of UDP transport protocol, it has its own delivery-guarantee mechanism: if ISAKMP cannot receive responder within certain period, it will retransmite the message until the maximum retry limit is reached.

Figure 318. IKE configuration

Now that we have completed configuring the client, we can start negotiation.

After configuring a gateway and host you should not be able to access that endpoint unless you establish a VPN tunnel successfully because of the filter rules. You can try to issue the `ping` command to test the connection prior to configuring the client so that you can make sure that the native IP connection is working between two systems. To activate the connection:

- Right-click the Port Entry, which in our example, is **Wildcard Connections,** under the OS390 RA03 Gateway definition, and then click **Start ISAKMP/OAKLEY Negotiation**.
- You will see a status bar for a moment before it disappears, if the phase 1 negotiation is successful. If it is unsuccessful, you will see a status bar and finally an error message indicating the connection is unsuccessful.
- After completing the phase 1 negotiation successfully, the phase 2 negotiation will be invoked automatically.
- Again, you will see a status bar, which should disappear shortly if all is well.
- If phase 2 is successfully completed, click the **(+)** plus sign next to Wildcard Connections and you should see the security protocol in place, which in our case, is ESP.

E.5.2.4 Business partner: OS/390 to OS/390

If your business partner is an OS/390 system rather than Windows NT, the configuration is almost identical to that described in E.5.1, “Branch office interconnect over the Internet scenario details” on page 376. There are two main differences:

1. Define your ESP and/or AH transform objects in tunnel mode instead of transport mode.
2. Be sure that the IP filters of the security gateways between the two host-to-host connections permit the initial communication of the ISAKMP servers.

E.5.3 Remote user access from the Internet scenario: details

Suppose your company has several employees who work from home. Throughout any given workday they may require access to confidential information on an OS/390 within your corporate intranet. Your company currently uses an expensive toll-free telephone number for dial-in remote access. Although your company wants to continue providing remote users secure access to their intranet, they ultimately want to reduce the expense associated with the toll-free number. This can be done by accessing the public Internet through an Internet service provider (ISP). Remote users will then be able to communicate with the OS/390 in your intranet and access the information they need at a cost equal only to the ISP fee. The connection is secured by using OS/390 Dynamic VPN.

The remote user dials in to his or her ISP by using whatever procedures and protocols the ISP requires. For example, many ISPs require remote hosts to use the Point-to-Point Protocol (PPP), and to identify themselves by an account name and a password. Then, the ISP typically assigns the remote user a dynamic IP address. Because the server to which the user wants to connect has no way of knowing his or her IP address in advance, the remote host must always initiate the connection.

For this situation, you still need IPSec for protection against security exposures in the Internet. The principles of configuring IPSec in this case are the same as in E.5.2, “Business partner or supplier scenario: details” on page 409, with one essential difference. Because the IP address of the remote workstation is not known (it is usually assigned dynamically by the ISP), the aggressive mode of IKE negotiation (see Figure 318 on page 435) must be used. Aggressive mode uses only three messages to set up the association (compared to six in main mode), but does not protect the identities of the partners.

Appendix F. VPN planning worksheets

Using the following templates, specify the information needed to plan for your dynamic VPN configuration. Create as many worksheets as you need for each TCP/IP stack planned to be configured with a dynamic tunnel.

Table 45. Initial design worksheet for dynamic connections

Information needed to create dynamic VPNs	Answers
What is the type of connection to be created? - Gateway-to-gateway - Host-to-gateway - Gateway-to-host - Host-to-host - Gateway-to-dynamic IP user - Host-to-dynamic IP user	
What type of security and system performance is required to protect the keys? - Highest security, lowest performance - Balance security and performance - Lowest security and highest performance - Add your own definition	
What type of security and system performance is required to protect the data? - Highest security, lowest performance - Balance security and performance - Lowest security and highest performance - Add your own definition	

Table 46. Key management planning

VPN Parameter	**Value**
Key Management Policy: Key Policy, Proposal, Transform	
Key Policy Name:	
Initiator Negotiation	
Responder Negotiation	
Key Proposal Name:	
Key Transform Name:	
Authentication Method	
Hash Algorithm	
Encryption Algorithm	
Diffie-Hellman Group	
Maximum Key Lifetime	
Maximum Size Limit	
Key Lifetime Range	
Size Limit Range	

Table 47. Data management planning

VPN Parameter	Value
Data Management Policy: Data Policy, Proposal, AH and ESP Transform:	
Data Policy Name:	
Perfect Forward Secrecy (PFS)	
Data Proposal Name:	
AH Transform Object Name:	
AH Encapsulation Mode	
AH Authentication Algorithm	
AH Maximum Data Lifetime	
AH Maximum Size Limit	
AH Data Lifetime Range	
AH Size Limit Range	
ESP Transform Object Name:	
ESP Encapsulation Mode	
ESP Authentication Algorithm	
ESP Encryption Algorithm	
ESP Maximum Data Lifetime	
ESP Maximum Size Limit	
ESP Data Lifetime Range	
ESP Size Limit Range	

Table 48. Network object planning

VPN Parameter	Value
Source network object name:	
IP Type	
IP Address	
Subnet Mask	
Start IP Address	
End IP Address	
Destination network object name:	
IP Type	
IP Address	
Subnet Mask	
Start IP Address	
End IP Address	

Table 49. Connection planning

VPN Parameter	Value
Connection management: Connection, Services, Rules	
Connection Name:	
Source network object name:	
Destination network object name:	
Service Object Name:	
Rule Object Names	
Override Log Control	
Override Manual VPN Tunnel ID	
Control By Time of Day	
Control By Days	

Table 50. Rule objects planning

VPN Parameter	**Value**
Rule Name:	
Action	
Protocol	
Source Port Type and Operation	
Destination Port Type and Operation	
Interface Settings	
Routing	
Direction	
Log Control	
Dynamic Tunnel Policy Name	

Table 51. *Dynamic tunnel policy planning*

VPN Parameter	**Value**
Dynamic Tunnel Policy:	
Data Policy	
Initiation	
Connection Lifetime	

Table 52. Authentication Information planning

VPN Parameter	Value
Authentication Information:	
Remote Key Server	
Shared Key	
Certificate Authority:	
RACDCERT Label	
Key Ring	
User ID	
Key Ring Name	

Table 53. Dynamic VPN connection planning

VPN Parameter	**Value**
Dynamic VPN Connection:	
Source network object name	
Destination network object name	
Source Port	
Destination Port	
Automatic Activation	
Protocol	
Remote Key Server:	
Auth IP Type	
Auth ID	
IP address	
Host Name	
Key Server Group:	
Key Policy	
Local Key Server:	
Auth IP Type	
Auth ID	
IP address	
Host Name	
Remote Key Servers	

Table 54. VPN On-Demand planning

VPN Parameter	**Value**
On-Demand name	
Source IP granularity	
Destination IP granularity	
Gateway key server	
Key server group	

Appendix G. Special notices

This publication is intended to help readers to implement secure TCP/IP network environment with OS/390 or z/OS systems. The information in this publication is not intended as the specification of any programming interfaces that are provided by IBM Communications Server for OS/390 IP Services and z/OS IBM Communications Server. See the PUBLICATIONS section of the IBM Programming Announcement for IBM Communications Server 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, North Castle Drive, Armonk, NY 10504-1785.

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 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.

Appendix H. Related publications

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

H.1 IBM Redbooks

For information on ordering these ITSO publications see "How to get IBM Redbooks" on page 457.

- *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 1: Configuration and Routing*, SG24-5227
- *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228
- *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5228
- *IBM Communications Server for OS/390 TCP/IP 2000 Update Technical Presentation Guide*, SG24-6162
- *SecureWay CS OS/390 V2R8 TCP/IP: Guide to Enhancements*, SG24-5631
- *Managing OS/390 TCP/IP with SNMP*, SG24-5866
- *TCP/IP in a Sysplex*, SG24-5235
- *A Comprehensive Guide to Virtual Private Networks, Volume I: IBM Firewall, Server and Client Solutions*, SG24-5201
- *A Comprehensive Guide to Virtual Private Networks, Volume II: IBM Nways Router Solutions*, SG24-5234
- *A Comprehensive Guide to Virtual Private Networks, Volume III: Cross-Platform Key and Policy Management*, SG24-5309
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *The Basics of IP Network Design*, SG24-2580
- *OS/390 Security Server 1999 Updates: Installation Guide*, SG24-5629
- *Enterprise Web Serving with the Lotus Domino Go Webserver for OS/390*, SG24-2074
- *Global Server Certificate Usage with OS/390 Webservers*, SG24-5623
- *OS/390 Workload Manager Implementation and Exploitation*, SG24-5326
- *Deploying a Public Key Infrastructure*, SG24-5512

H.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at **ibm.com**/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	**Collection Kit Number**
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039

CD-ROM Title	**Collection Kit Number**
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

H.3 Other resources

These publications are also relevant as further information sources:

- *IBM Communications Server for OS/390 IP Migration*, SC31-8512
- *IBM Communications Server for OS/390 IP Configuration Guide*, SC31-8725
- *IBM Communications Server for OS/390 IP Configuration Reference*, SC31-8726
- *IBM Communications Server for OS/390 IP User's Guide*, GC31-8514
- *IBM Communications Server for OS/390 IP Application Programming Interface Guide*, SC31-8516
- *OS/390 IBM Communications Server IP Messages: Volume 1 (EZA)*, SC31-8517
- *IBM Communications Server for OS/390 IP Messages: Volume 2 (EZB)*, SC31-8570
- *IBM Communications Server for OS/390 IP Messages: Volume 3 (EZY-EZZ-SNM)*, SC31-8674
- *IBM Communications Server for OS/390 IP and SNA Codes*, SC31-8571
- *IBM Communications Server for OS/390 IP Diagnosis Guide*, SC31-8521
- *IBM Communications Server for OS/390 Quick Reference*, SX75-0121
- *IBM TCP/IP Performance Tuning Guide,* SC31-7188
- *OS/390 V2R10.0 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835
- *SecureWay Security Server Network Authentication and Privacy Service Administration*, SC24-5896
- *SecureWay Security Server Network Authentication and Privacy Service Programming*, SC24-5897
- *OS/390 Network File System Customization and Operation*, SC26-7253
- *OS/390 Network File System User's Guide*, SC26-7254
- *OS/390 UNIX System Services Planning*, SC28-1890
- *OS/390 UNIX System Services User's Guide*, SC28-1891
- *OS/390 UNIX System Services Command Reference*, SC28-1892
- *UNIX System Services File System Interface Reference*, SC28-1909
- *S/390 Open Systems Adapter-Express Customer's Guide and Reference*, SA22-7403

- *Planning for the S/390 Open Systems Adapter (OSA-1, OSA-2) Feature*, GC23-3870
- *OS/390 Security Server (RACF) Command Language Reference*, SC28-1919
- *OS/390 Security Server for OS/390 RACF Security Administrator's Guide*, SC28-1915
- *OS/390 System Secure Sockets Layer Programming Guide and Reference*, SC24-5877
- *OS/390 Open Cryptographic Services Facility Application Developer's Guide and Reference*, SC24-5875
- *OS/390 Security Server Open Cryptographic Enhanced Plug-ins Guide and Reference*, SA22-7429
- *OS/390 Security Server LDAP Server Administration and Usage Guide*, SC24-5861
- *OS/390 MVS Programming Workload Management Services*, GC28-1773
- *DB2 for OS/390 V5 Installation Guide*, GC26-8970
- DB2 for OS/390 V5 Call Level Interface Guide and Reference, SC26-8959
- David Zeltserman, *A Practical Guide to SNMPv3 and Network Management*, Prentice Hall, SR23-9114, ISBN 0-13-021453-1
- W. Richard Stevens, *TCP/IP Illustrated Volume 1 - The Protocols,* Addison-Wesley, SR28-5586, ISBN 0-201-63346-9
- W. Richard Stevens, *Advanced Programming in the UNIX Environment*, Addison-Wesley, ISBN: 0-201-56317-7
- Evi Nemehn, Garth Snyder, Scott Seebass, and Trent Hein, *UNIX Systems Administration Handbook, 2nd Edition,* Prentice Hall, 1995, ISBN 0131510517
- *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design,* by Electronic Frontier Foundation, John Gilmore (Editor), 1988
- Hans Dobbertin, *Cryptanalysis of MD5 Compress*, 1996, available at:

  `http://www.cs.ucsd.edu/users/bsy/dobbertin.ps`

H.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- IBM Enterprise Servers home page:

  `http://www.ibm.com/servers/eserver/zseries/`
- IBM zSeries networking home page:

  `http://www.ibm.com/servers/eserver/zseries/networking/`
- z/OS UNIX System Services performance home page:

  `http://www.ibm.com/servers/eserver/zseries/zos/unix/bpxa1tun.html`
- z/OS Internet Library home page:

  `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`
- The RACF PKISERV sample application:

  `http://www-1.ibm.com/servers/eserver/zseries/zos/racf/webca.html`

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** ibm.com/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  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.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax 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.

Abbreviations and acronyms

ACL	access control lists
AH	Authentication Header
APAR	authorized program analysis report
API	application programming interface
APPN	Advanced Peer-to-Peer Networking
ARM	automatic restart management
AS	authentication server
ATM	Asynchronous Transfer Mode
BINL	Bind Image Negotiation Layer
BSDS	bootstrap data set
CA	Certificate Authority
CDLC	Channel Data Link Control
CDMF	Commercial Data Masking Facility algorithm
CEC	Central Electronics Complex
CLAW	Common Link Access to Workstation
CLI	call level interface
CRC	cyclical redundancy check
CRL	Certificate Revocation List
CSM	communication storage manager
CTC	channel-to-channel
DCAR	Digital Certificate Access Requester
DCAS	Digital Certificate Access Server
DDF	distributed data facility
DES	Data Encryption Standard
DFSMS	Data Facility Storage Management Subsystem
DHCP	Dynamic Host Configuration Protocol
DLC	data link control
DLL	dynamic link library
DMZ	demilitarized zone
DN	distinguished name
DNS	Domain Name System
DoS	denial of service
DPI	Distributed Programming Interface
DSCP	differentiated services codepoint
ECSA	Extended Common Service Area
EGP	External Gateway Protocol
ELF	Express Logon Feature
ESM	External Security Manager
ESP	Encapsulating Security Payload
FEP	front-end processor
FRCA	Fast Response Cache Accelerator
FTP	File Transfer Protocol
GbE	Gigabit Ethernet
GUI	graphical user interface
HFS	Hierarchical File System
HMAC	Hashed-Based Message Authentication Code
HOD	Host On-Demand
HPDT	high-performance data transfer
HPNS	High-Performance Native Sockets
HSAS	High Speed Access Services
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IBM	International Business Machines Corporation
ICCP	IP Channel Communications Program
ICMP	Internet control message protocol
ICP	Interconnect Controller Program
ICRF	Integrated Cryptographic Feature
ICSF	Integrated Cryptographic Service Facility
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force

IKE	Internet Key Exchange
I/O	input/output
IP	Internet Protocol
IPA	IP assist
IPL	initial program load
IPSec	IP security architecture
iQDIO	Internal Queued Direct I/O
IS	integrated service
ISAKMP	Internet Security Association Key Management Protocol
ISPF	Interactive Structured Programming Facility
ITSO	International Technical Support Organization
IUCV	inter-user communication vehicle
IVP	installation verification procedure
KDC	Key Distribution Center
LAN	local area network
LCS	LAN channel station
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Directory Interchange Format
LFS	logical file system
LPD	line printer daemon
LPR	line printer requester
LU	logical unit
MAC	Message Authentication Code
MAE	Multiaccess Enclosure
MDA	Mail Delivery Agent
MIB	Management Information Base
MTA	Mail Transfer Agent
MUA	Mail User Agent
MVS	multiple virtual storage
NFS	Network File System
MPC	multipath channel
NAT	Network Address Translation
NCP	Network Control Program
NCS	Network Computing System
NFS	Network File System
NIST	National Institute for Standards and Technology
NPF	Network Print Facility
NQN	Network Qualified Name
NSA	National Security Agency
ODBC	Open Database Connectivity
OCSF	Open Cryptographic Services Facility
OSA	Open Systems Adapter
OSPF	Open Shortest Path First
PCLS	Policy Core LDAP Schema
PDU	protocol data units
PDS	Partition Data Set
PDSE	Partition Data Set Extended
PFS	Perfect Forward Secrecy
PFS	physical file system
PHB	per-hop behavior
PKDS	Public Key Data Set
PKI	Public Key Infrastructure
POP3	Post Office Protocol
POSIX	portable operating system interface for computer environments
PPP	Point-to-Point Protocol
PU	physical unit
PXE	Pre-Boot Execution
QDIO	Queued Direct I/O
RACF	Resource Access Control Facility
RAS	reliability, availability, and serviceability
RFC	Request for Comments
RIP	Routing Information Protocol
RSVP	Resource Reservation Protocol
RTT	round trip time
SA	security association
SAF	Security Access Facilities
SD	Sysplex Distributor
SHS	Secure Hash Standard
SLA	service level agreement
SLAPM	service level agreement performance monitor

SMS	Storage Management Subsystem
SMSG	special message (VM)
SMTP	Simple Mail Transfer Protocol
SNA	Systems Network Architecture
SNMP	Simple Network Management Protocol
SOA	start of authority
SPI	Security Parameter Index
SPUFI	SQL Processor Using File Input
SQE	SNMP Query Engine
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TFTP	Trivial file transfer protocol
TGS	ticket-granting server
TNF	termination notification facility
TTL	time to live
TOS	type of service
TRM	traffic regulation and management
UDP	User Datagram Protocol
UCS	Universal Character Set
UNDI	universal network driver interface
USM	user security model
UTF	UCS transformation format
UUID	Client User/Group ID
VIPA	Virtual IP address
VMCF	virtual machine communication facility
VPN	virtual private network
VTAM	Virtual Telecommunications Access Method
WAN	wide area network
WAS	WebSphere Application Server
WLM	workload manager
X.25	ISO standard for interface to packet switched communications services
XCF	cross-system coupling facility
XPG	X/OPEN portability guide

Index

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 845 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-5383-01 Secure e-business in TCP/IP Networks on OS/390 and z/OS
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	 O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

IBM

Redbooks

Secure e-business in TCP/IP Networks on OS/390 and z/OS

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Secure e-business in TCP/IP Networks on OS/390 and z/OS

Guide to the secure e-business environment on OS/390 and z/OS

Understand TCP/IP security technologies

IP network and application security using OS/390 and z/OS

The Internet has changed the way of doing business all over the world. It enables consumers to perform many activities, such as banking, investment, shopping, and researching, from virtually anywhere they can get online. As the volume of the business activities over the Internet increases by leaps and bounds, companies that provide services require a more powerful, secure, and reliable platform, because they are expected to serve customers all over the world 24 hours a day, seven days a week.

However, as the volume of business grows, so does the risk. There are some people with malicious intentions trying to prevent those services from being performed and to impose heavy losses on businesses. OS/390 (and now z/OS) offers functionality, especially for networking and security, that has made it the ideal e-business platform with huge computing capacity, reliability, extensibility, and safety.

This redbook shows you how to secure your network access and TCP/IP applications on an OS/390 or z/OS platform, with practical examples of various configurations and implementations. The focus of this redbook, however, is not on detailed implementation information, but on helping you to establish a secure e-business environment using OS/390 or z/OS.

SG24-5383-01 ISBN 0738421561